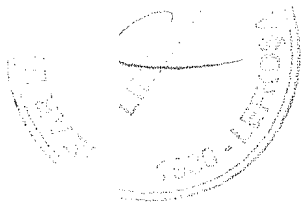


NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

MICROSOFT SHAREPOINT PRODUCTS AND TECHNOLOGIES

Graduation Project
COM 400

Student: Filiz Akosman(991124)

Supervisor: Mr Kaan Uyar

Nicosia 2005

ACKNOWLEDGEMENTS

First, I would like to thank my superviser Mr. Kaan Uyar for his invaluable advice and belief in my work and myself over the course of this Graduation Project.

Than, I thank my family for their constant encouragement and support during the preparation of this project.

Finally, I would also like to thank all my friends especially Mr. Erkan Hakverdi for their advice and support.

ABSTRACT

More and more organizations are taking advantage of the opportunity to share information with partners and customers. Microsoft SharePointPortal Server c:ar be use to provide external users with access .to documents, data, information, and applications. This can help organizations take advantage of many business-to-business opportunities. For example, with a SharePoint Portal Server, .iL8a,npE!~mit customers to place orders, view the status of their orders, or check the availability of a specific item.

SharePoint Portal Server is a powerful portal solution that provides a single point of access to people, teams, knowledge, and applications. It helps you put information to work, connect people and spaces, and target and personalize information. SharePoint Portal Server builds on and enhapöE!s•.. 'y'vindows SharePoint Services by providing enterprise features and services,· such as global search, personalized portals, information delivery, and support for enterprise application integration and single sign-on.

SharePoint Portal Server is a reliable, scalable, and easy-to-deploy and manage platform for developing customized portal solutions.. Wit? phFrnfoint Portal Server, that can tie together disconnected islanqs Of)9.FİFY<imt.E!ig.rate technology with business processes, and target and./pE!rs9rıalifE!iipfq.rrnation for groups and individual users, so users can be more pro9uftiVE! cı.ndteffective in their work. SharePoint Portal Server achieves mi~ yvhHe tcı~ing ad.vantage of .exlstinq infrastructure investments and keeping the total cost of ownership low.

TABLE OF CONTENTS

INTRODUCTION

Collaboration, communication, and portal technologies have grown dramatically in the last decade. Bee of the availability and affordability of fast network connections, the Internet, and local area networks (LP and intranets, an individual cart connect to people and information that are located down the hall or or other side of the globe. Emerging technologies such as Extensible Markup Language (XML), Simple 01 Access Protocol (SOAP), and the Microsoft® .NET Framework create new opportunities for integration automation. Organizations need to take advantage of these opportunities to help their employees be r productive, and to work better with partners and customers. Additionally, organizatiorıs need these emer technologies to work well with their existing investment in applications and infrastructure. They need solutions that make managing and maintaining their organization easier instead of harder. Microsoft O SharePoint™ Portal Server is designed to help organizations meet these goals.

Teams need to communicate, share tasks, and work together in order to get their workdôrie. Addition team members often need to work with people from other teams and divisions to complete tasks. This is more difficult when people are located in different locations. SharePoint Portal Server helps them communicate, and collaborate with people no matter where they are located.

SharePoint Portal Server and Windows SharePoint Services provide teams with the t??Is ~n1i~.~f"!ce~ need to work together, both in the ongoing context of a team and in the context of ad-hoc tasks ~~d proj SharePoint Portal Server makes creating and managing collaborative sites and workspaces easy by facilit: communication throughout the organization. The searching and indexing features in SharePoifitPortal S< can help the person find the people and teams who have the knowledge and information needed. The result is. an organization that is more efficient and effective in its work, which translates into l: productivity and higher employee satisfaction.

CHAPTER 1

1.1 Microsoft SharePoint Products and Technologies Overview

Based on customer feedback and the experience of developing the first version of SharePoint Products and Technologies, Microsoft designed the next generation of Microsoft SharePoint Products and Technologies to use a common set of Microsoft Windows Server 2003 services named Windows SharePoint Services. This set of services takes advantage of the performance, stability, and security features of the Microsoft .NET Framework. The SharePoint Portal Server 2003 application is based on Windows SharePoint Services. Windows SharePoint Services is a set of services that you can use to create and maintain many team sites, and SharePoint Portal Server 2003 is a server product that adds features that individuals can use to build and manage integrated, large-scale portal solutions.

1.2 Architecture and Design

SharePoint Team Servic.es by rvlicrosôff and Micrôsb~ ShfrePoint Portal Server 2001 were the original versionsof Microsoft Products and Technologies, released in 2001.SharePoint Team Services addressed the Web-based sharing and communication needs of teams 'and team Web sites, and SharePoint Portal Server 2001 addressed the document management, Web portal, and enterprise search requirements of a portal solution. To deliver these features, Microsoft used the best available (but different) technology platforms to build Windows SharePoint Team Services and SharePoint Portal Server 2001.

To determine the main design goalsofthelatest versi~n.}fSlifrePointProducts and.Technolôgies, Microsoft used feedback from customers and partners, the lessons learned from its large-scale internal deployment of SharePoint sites, and the experience gained from creating the first version of SharePoint Products and Technologies.

1.3 Customer Feedback

The Micros.oft Sh.arePoint Products and Technologies tefrn Was fortunate to receive· early feedback· from many types of customers and partners and to use that fee.dback to drive the design goals for the latest version of SharePdirtt Products and Technologies.

The feedback focused on.three topics:

- Enterprise abilities - Greatly improve scalability, manageability, security, and performance while providing the greatest possible cost effectiveness for a wide range of customers, from the smallest project teams up to the largest organizations.
- Single, integrated platform - Base the next version of SharePoint Products and Technologies on a single; integrated set of technologies, so end users, network. administrators, and developers can increase their productivity and reduce their costs.
- Advanced collaboration features - Improve the features and capabilities of SharePoint Products and Technologies to deliver the Microsoft vision of smart, connected spaces for people, meetings, teams, projects, divisions, and organizations .. Thisjncludes more powerful information organization and searching capabilities, comprehensive support for enterprise application integration using Microsoft BizTalk Server, Web site personalization, and deeper integration with Microsoft Office to provide rich support for data analysis and work sharing.

1.4 Main Design Goals

The overall design goal for SharePoint Products and Technologies was to unify and integrate Wi SharePoint Services as a technology platform on which to build products such as SharePöint Portal 2003. This overall goal of unification and integration was divided into four areas:

- Consistent SharePoint Products and Technologies experience for users, developers, a professionals
- Consistency and integration with the Microsoft .NET Framework
- Integrated storage strategy
- Trustworthy Computing: security and reliability

1.4.l\Consistent Experience for Users, Developers, and Network Administrators

The most obvious change and improvement in the latest .yersio.n.of.Share~o.inf,ı>r§9Hst~:ıpq(f~f,£11.o.10 the creation of a single, integrated technology platform named Vv'ind.ows Sh:ı.re.I)o.int Ş$f}'jf,fS.ı.)~b~! Portal Server 2003 is a server product pased .on Windows Sharef'oint Services. ~harePo.intProduc Technologies provides a. consistent experience for users as they work with both Windows Shaı Services-basedsites and SharePoint Portal Server 2003-basecl portal sites.

Solution developers for SharePoint Products and Technologies now need to know only one user in technology (ASP.NET Web pages and controls) andone SharePoint J;>ı;e>ducts andTec:h11e>logiesobject to create advanced sharing solutions with .Sharel'ointProdµc:Js and Tech11ologies.

1.4.2 Consistency and Integration with the Microsoft .NET Framewor

The strategy of using the Microsoft .NET Framework, Web services, .Microsoft:Y"isı.falStüdioNE' Windows Server 2003 to build SharePoint Products and Technologies is part. of Microsoft's techı strategy for connecting people, inforınation, applications,„ı11cl cle\jc:es0 *'Thf* :fif~t ag..vantage of• usi Microsoft .NET Framework:.lS that. it is the. most .scalablf, fle:x.ibl.f, .ınd şec:ury foundati.on for bu deploying, and managing enterprise Web applications, includingSharel'oint Products and Technologie sharing sites and portal sites.

Additionally, integrating SharePoint Products and Technologies with information from virtuall enterpriseapplication is easy with the support for Web services}nclu.d.edh the Microsoft .NET Frame"

1.4.3 Integrated Storage Strategy

The long-term strategy for storage technology at Microsoft if to lake advantage of SQL Server rel database technology and XML data storage technology wherever possible, and SharePoint Produc Technologies is a showcase example. -Except for the full-text search indices created by Microsoft technologies, all content, configuration informa~orl, and other.SharePoirıfProducts and Technologies stored in SQL Server databases. Using a single, consistent, integrated data storage platform creates sigr advantages for IT professionals and developers by increasingtheir productivity andreducing their day- development, deployment, and management costs.

1.4.4 Trustworthy Computing: Security and Reliability

The secure, reliable operation of everyday computer systems is at the heart of the Trustworthy Corr initiative at Microsoft. All new Microsoft products, including SharePoint Products and Technologies,

to these Trustworthy Computing principles and take advantagtof the security and reliability engineering that is part of Windows Server 2003 and the Microsoft .NET framework. Each developer at Microsoft takes extensive software security training and applies their knowledge by performing security audits of each Microsoft software component that they are responsible for.

The development teams for SharePoint Products and Technologies groups took advantage of the built-in security and reliability of Windows Server 2003 and the Microsoft .NET Framework when they designed, implemented and tested the new version ofSharePoint Products and Technologies.

1.5 Architecture and Design Decisions

1.fül Integrated Storage

Shar~Point Tearrı Services used a hybrid model of Web server, file system, Windows registry, and SQL Server-based storage to manage documents, lists, vi~\Vs, and configuration information. SharePoint Portal Server 2001 used a document store based on the Microsoft Web Storage System (the şamestorage technology used by Microsoft Exchange Server) for most data storage requirements. Both of these solutions required content to be stored on the same server that hosted the Web portal. This requirement limited the range of deployment scenarios and scalability.

Each storage solution'served the original version of its product well. However, these storage solutions did not support the additional requirements for administration, management, performance, scalability, and functionality in the next generation of SharePoint Products and Technologies. For example, backup and restore operations are .difficult to implement and manage when the relevant data is spread out among many differentstorage systems on the server. t

For ex.aınple,backup and restore operations are difficult to iınplement and manage when the relevant data is spread out among many different storage systems on the server. Windows SharePoint Services stores all documents, lists, views, and configuration information in. SQL Server content stores. Because of this, Windows SharePoint Services offers true enterprise scalability and personalized Web portal experiences.

SharePoint PôrtafServer 2003 uses Windows SharePtunfServfo~sto take advantage of the same SQLServer contents.tore archit~ctuı:e. SharePoi~t Portal Serv~r .903 als.o Bupports the option of installing.backward-compatible document libraries (Web Storage System•based)for document storage. The backward-compatible documenr libraries are. l)()rripatibl~ with SharePoint Rfrtal' s.erver 2001 document approval and routing, and they support multiple •docuınentprofiles for each. docu111ent library folder. With the backward-compatible document libraries, you can use a phased strategy to migrate to SharePoint Portal Server 2003.

SharePÔfôt Portal Server 2003 still requires a SQLServer content store for managing ASP.NET portal Web pages, lists, views, and configuration information: y.!indowsSharePoint .Services and Sharef'oint Portal Server 2003 both use SQL Server content stores and require Microsoft SQL Server or Microsoft SQL Server Data Engine (MSDE). Microsoft SQL. Server is a separately Jicensed product that is not included with Windows SharePoint Services or SharePoint Portal Server. MSDE is included with Windows SharePoint Services and with SharePoint Portal Server 2003.

1.5.2 ASP.NET Web Part Pages and Web Parts

SharePointProducts and Technologies now uses Web Part Pages and Web Parts based on the .NET Framework and ASP.NET.

SharePoint Team Services and SharePoint Portal Server 2001 used separate technologies to create and display (render) SharePoint sites in a Web browser. Web pages in SharePoint Team Services were based on Microsoft

FrontPage and Office Web Server technologies, and Web portal pages in SharePoint Portal Server 2001 were based on Web Storage System, dashboard, and Web Part technologies.

The next generation of SharePoint Products and Technologies uses Microsoft ASP .NET Web Part Pages to create and display ShareP?int sites ..in a Web brovıser. Web Part Pages are ASP .NET pages that take advantage of the. performance, stability, and security of the .NET Framework and Windows Server 2003. You can easily integrate Web Parts with Web services, Microsoft Office, and Microsoft BizTalk Server to provide powerful, flexible, and cost effective solutions for work sharing, enterprise applications, and portal sites.

1.5.3 Common Document Management Services

SharePoint Portal Server 2001 was designed to provide document version tracking and check-in/check-out document management functions. However, SharePoint Team Services was the solution that large numbers of teams used every day to create, review, approve, and manage their Office documents, to plan and hold meetings, and to track project tasks. Document version tracking and document. check-in/check-out are now included in Windows SharePoint Services, where users need them the most and \\!hery all SharePoint Products and Technologies solutions can take advantage .of these document management fünctio11s.'.

Document management is one of the most valuable end-user features in Windows SlıarePointServices,.ıınd.i is the area in which Microsoft made the most changes. The following list shows the main differences betwee: the document management functions in SharePoint Portal Server 2001 and the document managemer functions in Windows SharePoint Services:

- In Sharel'oint Portal Server 2001, the document libraries based. on the 'lv'el:ı St()ı;ıgY.>Systyı supported multiple document profiles for each document library .folder. In. ~il}4p'ıVs . .Ş.J:ı.ıref~i Services, the SQL Server content store supports one set of properties (the equivalent of oı document profile) for each document· library. Beca.use. pftpişo..¥?:HiJPlly.·'ıVatltrtı,ıt?re füf Brop~rti of the most common document profile .in the area where it wo~m.nsıwıllyi~flp11g, an~then store t documents that use secondary documen.t profiles in a subarea·,.· U.J.iB.·&· ... secondary document profile.
- SharePoint Portal Server 2001 provided both serial and parallel routing and• approval process Windows SharePoint Services now provides a simpler, .one-st~ptnpd~ratorapproval process.
- SharePoint Portal. Server 2001 provided support for docurrıept y~r~ion tracking usipg 111~}orfi minor version numbers. Windows SharePoint Services and SharePoint Portal Server 2003 use ma version numbers only.
- SharePoint Portal Server 2001 only supported access. control atthe folder leyel ~nd the subfol level (and user and group exclusions at the file level). Windows SharePoint Servj?es ~nd ShareP< Portal Server 2003 support access control at the site level and the document library level.

1.5.4 Site Creation and Management Services

The original versions of Sharel'oint Portal Server 2001 and ShatePoint Team Services were complementary. SharePoint Team Services sites were easily deployed ill large· numbers within organization, and SharePoint Portal Server 2001 provided enterpiise search and Lirıks Web Parts to aggre team sites into one or more portal sites. To improve support for very large numbers of SharePoint Windows SharePoint Services now provides common site creation and management features, such as templates and self-service creation of SharePoint sites. SharePoint Portal Server 2003 adds the folio deployment and management features for the large organization: a site directory that provides an easy tı site registration system; preconfigured enterprise application integration (EAI) solutions with single sif support for third party applications; dynamically configurable site maps; large-scale server topı management; and the abilityto share multiple index and search servers.

1.5.5 Integrated Search Solution

For full-text content indexing and searching, SharePoint T~arn Servicesused the Windows Indexing service, an early version of Microsoft Search technologies included in Microsoft Windows 2000. When SharePoint Porta] Server 2001 was released, it included an updated enterprise version of Microsoft Search technologies.

Windows SharePoint Services uses the full-text searching features from the latest version of Microsoft SQL Server SharePoint Portal Server 2003 uses the latest version of Microsoft .NET Search services in addition to SQL Server full-text search.

1.5.6 Personalization and Audience-Targeted Information and Appllcations

To help individual users find and use the information and tools they need, a portal solution must support targeted delivery of content, information, and application functionality. This includes targeting information and applications to individuals, teams, divisions, and entire organizations. This also includes effective support for personalized content and support for group-basedportal page content.

Personalization is a service inclıided inWin.dows ShatePôlnt Services. SharePoint Portal Server 2003 uses Audiences to exteri~ thisSel'\'ice. A~djencesare dynamic.groups of users that share one or more common properties (for example, business function, department, or team membership). The properties that determine Audience membership can reside in an enterprise directory such as Microsoft Active Directory or any other SQL Server-based database. Audiences are used to determine which Web Parts appear on a particular Web page, and they can also act asa filter for the infonnatiprtd.i~played in those Web Parts. SharePoint Portal Server 2003 also supports creating and managing individual portal pages for each user.

1.5.7 Subscriptions and Alerts

In SharePoint Team Services and SharePoint Portal.Server 200.1, users could use Subscriptions to receive messages when your shared documents were changed, .The name of the Subscriptions feature has now changed to Alerts. Windows SharePoint Services and SharePoint Portal Server.2003 both continue to support Alerts.

1.6 Simple Single Server Configurations and Highly Scalable Server Farm Configurations

SharePoint Team Services and SharePoint Portal Server 2001 were deployed as single server solutions or as groups of servers, but little support was available for creating and deploying highly scalable server farms.

In the next generation of Microsoft SharePoint Products arid Technologies, Windows SharePoint Services is specifically designed to vastly improve performance for each server and to support deployment in highly scalable server farms using multiple stateless front-end servers connected to one or more back-end content servers.

1.7 Important Features and Terminology Used in SbarePoint Products and Technologies

In addition to understanding SharePoint Products and Technologies technical design, understanding the new and changed terminology introduced in the latest version of SharePoint Products and Technologies is also important.

1.8 SharePoint Sites and Site Collections

The terminology used to name the components in Sharel'oint Products and Technologies varies dependini whether you are an end user, a developer, a Windows administrator, or a SharePoint Products Technologies administrator.

For Sharef'oint Products and Technologies end users, the terminology used in the Sharel'oint Products Technologies Web interface is consistent with tlie long-term goals and direction for SharePoint Products Technologies.

Some of the terminology for developers is more consistent with the older Sharel'ointTeam Services ol model. The terminology for Windows administrators and Sharel'oint Products and Technok administrators is a mixture of SharePoint Team Services terminology and Windows SharePoint Ser terminology. Microsoft plans to increase the consistency of the terminology for developers and administr in future versions ofSharePoint Products and Technologies.

In SharePoint Team Services, the term for the top-level content directory ofa We::bserveri.s ."root Web.: In a multi-hosting environment, each virtual Web server that is configured on the Web server contains top-level (rootWeb) site. Additionally, the term for a site withinarnot '.Vebsite,iıı SharePQipf'J'earriSer is "subweb.11 "Subweb" is a term ad.optçdfrom Microsoft FrontPage'.Ve:t.~ites (the original techrıolog which SharePoint Team Services was built). You can create multiple subwebs in a root Web site, and yoı create subwebs within other subwebs.

In SharePoint Portal Server.2001,thetermfor the top-level content directoryçfaWebsçrv~tii~YWM~Sf The term for a site within a workspaceis "subdashboard." Users can create.additiqnal suhdashhoard personal dashboards for subprojects and individual users.

In the latest version of SharePoint Prnducts and Technologies; the tefül>'iiit~)fm'l~t~f }hf *ptfvi6uis* t There are two types of SharePoint ~.ites that you can us~ to di.vide ~it~ ~?ntent *int?* distinct, sepa manageable sites: top-level sitesan1suR~ites. A top-level Sh.~~e~?intsite. is t~eyaryntsity of all sites in collection. Top-level sites can ~?nt~i~ tl'lUltiple subsites, and si.ıbsites can also·contain·multiple sul continuing for as many levels as your users require.

Individuals can use this hierarchy to create a main subsite for their entire team and create individual su or shared sites for side projects. TopLJevelWeb sites and subsites pemıit different levels of control ov features and settings.

1.8.1 Single Server ScenaI"i()

SharePoint Products and Technologies siıppôrts both vertical, single server solutions and horizontal, farm solutions. This section describes a single server configuration:

One of the minimum system requirements for Windows SharePoint Services is. the Windows Serve operating system. Additionally, you must install and configure Internet Information Services {Il: ASP.NET before you install Windows SharePoint Services. When you install Windows SharePoint Se it creates and configures a virtual.~e.rverp.amedSharyPointCe.~traJAdmi.nistrationA,dditionally,if you Windows SharePoint Services in a single-server configuration, it automatically extends the existing Web site that was created when you installed IIS.

Windows IIS Web sites are alsoreferred to as IIS virtual Webservers, virtual servers, or v-servers. In a to the two default IIS virtual servers that are created when you install Windows SharePoint Services,) configure as many as nine end-user virtual servers with separate application pools or 99 end-user

servers with a shared application pool on a single Windows Server 2003-based computer. Each IIS virtual server can host multiple SQL Server content stores.

Each SQL Server content store for SharePoint Portal Server 2003 can only contain one portal site collection. Each SQL Server content store for Windows SharePoint Services can contain as many as 50,000 site collections, even if the site collection is hosted under a SharePoint Portal Server 2003 portal site.

A site collection is a set of Web sites that have the same owner and administration settings and that reside on the same virtual server. Each site collection contains a top-level Web site and can contain one or more subsites. A site collection serves as the administrative unit for assigning users to site groups and for granting security rights to site groups. For more information about site administration, see the next section. For a complete list of SharePoint Products and Technologies security rights.

Users can nest a site within another site. The term for the nested site is "subsite" or "child site." Each site (and subsite) uses the same SQL server content database as its parent site.

In SharePoint Portal Server 2003, the. term "site collection" is the equivalent of the older term "site" in SharePoint Team Services, and in the new version of SharePoint Products and Technologies, the term "site" or "SharePoint site" is the equivalent of the older term "subweb" in SharePoint Team Services. Be careful to avoid confusing the overlapping terminology. For more information about terminology in SharePoint Products and Technologies, seethe following table.

Table 1.1

SharePoint Products and Technologies terms	SharePoint Products and Technologies Object Model terms	SharePoint Team Services terms
Windows IIS Web site	SPVirtualServer	Windows IIS virtual server
Site collection	SPSite	Site
Top-level site	SPWeb	Root Web site
Child site	SPWeb	Child Web site
Site (including top-level sites and child sites	SPWeb	Subweb site (including root Web sites and child Web sites
Child site collection	SPWebCollection	Subweb collection
Site group	SPRole	Role
Cross-site group	SPGroup	n/a
Cross-site group collection	SPGroupCollection	n/a
Rights mapping for a principle	SPPermission	n/a
Access control list	SPPermissionCollection	Access control list
Security principle	SPUser	n/a
Security principle collection	SPUserCollection	n/a
Area	Area	Category (SharePoint Portal Server 2001)

If you are a developer, the SharePoint Products and Technologies object model uses the (sometimes overlapping) names from SharePoint Team Services for many of the new SharePoint Products and Technologies components. Use the preceding table to cross-reference terms from the old and new naming conventions.

1.8.2 Server Farm Scenario

In a server farm scenario, the terminology remains the same, and the requirement thatalFSharePoint sites the same SQL Server content database as the top-level site remains the same. However; you can use mul stateless front-end Web servers to support a large number of user connections and to render ASP.NET pages. Logically located behind these front-end Web servers, a SharePoint Products and Technok configuration can use many back-end database servers.

You can use multiple content database servers to support multiple site collections and to provide fault-tol fail-over. Optionally, you can use separate database servers to store the configuration database for server configuration maps and site collection-to-content database maps.

1.8.3 Portal Sites

One of the minimum system requirements for SharePoint Portal Server2003 is Wip?o\\\'s Shar.ePointSer When you install SharePoint Portal Server 2003, Setup automatically installs Windows SharePoint Serv it is not already installed.

You can configure only one portal site for each end-user IIS virtual Web server.The portal site correspc the SharePoint Products and Technologies top-level site for both the virtual server and the site coll rooted at the virtual server.

1.9 Security

1.9.1 Site Groups and Rights

SharePoiritProducts and Technologies uses a security model based on site groups .andrights .. Site gro groups of users with related security requirements. Security rights are assig11~d to each security groı can customize the rights assigned to. these site groups or add new site groups to conıbine.different rights. By default, Windows SharePoint Services includes five site groups: Administrator, · Web D Contributor, Reader, and Guest.

Members of the Administrator site group have complete control over a Web site. They can configure manage users and. site groups, and view usage analysis data.

Members of the Web Designer site group can use a SharePoint Products and Technologies-compatil page editor such as Microsoft Office FrontPage 2003 to customize the Web site.

Members of the Contributor site group can interact with Web Parts, lists, and document Additionally, they can create and manage personal views and cross-site groups: and personalize \ pages.

Members of the Reader site group can view items in lists and document libraries, view pages in the create a site using Self-Service Site Creation.

The Guest site group is designed to be combined with specific permissions for specifics lists, so gı can have access to a specific list without having access to the entire site. You cannot customize or Guest site group.

If you use one of the SharePoint Products and Technologies upgrade tools to create the Sharel'oinı and Technologies installation, the upgrade tool inspects the permissions granted to each role in S

Team Services or SharePoint Portal Server 2001 and uses the permissions granted to each role to assign users to corresponding SharePoint Products and Technologies site groups.

1.9.2 Cross-Site Groups, Local Groups, and Domain Groups

Cross-site groups, local groups, and domain groups can be members of site groups.

Cross-site groups are collections of users who can be managed as a single group across multiple SharePoint Products and Technologies sites. Cross-site groups are configured in SharePoint Products and Technologies, and they can be members of a site group.

Domain groups and local groups can also be members of site groups. However, cross-site groups and site groups cannot be members of local groups or domain groups.

1.9.3 Authentication

Authentication is the process of verifying the identity of user or a process. Microsoft Internet Information Services (IIS) handles authentication for SharePoint Products and Technologies. To be authenticated, you need a local user account or a domain user account (if in a networked domain). In most cases, a domain account is a better choice than a local account.

Users can configure authentication to operate in either pre-existing account mode or account creation mode. In pre-existing account mode, SharePoint Products and Techpologies does not automatically create new user accounts. In account creation mode, SharePoint Products and }'echnologies can ~utomatically create new user accounts in Active Directory. Account creation mode js a feature that you must select when you install Windows SharePoint Services or SharePoint Portal Server 2003.

If you use account creation mode, make sure that IIS is configured to use basic authentication. SharePoint Products and Technologies no longer supports IIS digest authentication.

SharePoint Products and Technologies does not fully support Microsoft Passport authentication.

1.9.4 Authorization

SharePoint Products and Technologies stores all securityrnetadata (groups and rights) in SQL Server content stores. User security metadata for Sharel'oint Products and Technologies is not stored in IIS or anywhere else in Windows.

After US uses a local computer account or an Active Directory account to authenticate a user, SharePoint Products and Technologies compares the rights assigned to the user by IIS with the access control information for the SharePoint site to determine which SharePoint site resources the user is permitted to use.

Active Directory is not required for Windows Sharel'oint .Services or SharePoint Portal Server 2003. However, without Active Directory, SharePoint Portal Server you cannot pre-populate and synchronize the SharePoint Portal Server profile database with the list of users from Active Directory, and users' personal sites are not registered for cross-farm synchronization in a multi-server configuration. For best results, deploy SharePoint Products and Technologies in an Active Directoıy environment.

1.9.5 Site Administration

Members of the Administrator site group for a top-level Web site can control settings and features for th, level Web site and any subsites. For example, an administrator of a top-level Web site can:

- Add, delete, or change user permissions
- View usage statistics
- Change regional settings
- Manage Web Part catalogs and template catalogs
- Manage Web document discussions and alerts
- Change the name, description, theme, and home page organization of the site
- Configure settings (for example, regional settings) for the top-level Web site and all subsites
- Update e-mail settings for the top-level Web site and all subsites
- Configure Web Parts settings for the top-level Web site and all subsites

A member of the Administrator site group for a subsite can control settings a~d fei:ıtures only foı particular subsite, and the administrator of a site under that subsite can control settings and features for that particular secorıd-level subsite. For example, an administrator of a subsite can:

- Add, delete, or change user permissions
- View usage statistics
- Change regional settings
- Manage Web Part catalogs and template catalogs
- Manage Web document discussions and subscriptions
- Change the name, description, theme, and home page organization for the

1.9.6 Document and Content Storage

SharePoint Products and Technologies supports two types of content stores. The priınafy store is the Server content store. Based on Microsoft SQL Server technology, the SQL Server content store provii single, consistent data storage solution for document content, list content, and metadata. You can use con Windows and SQL Server management tools and development tools to easily manage, tune, back up enhance SQL Server content stores.

When you install Sharel'oint Portal Server 2003, you have the ofti(jn of installing the backward-comps document store. The backward-compatible document store is an µpdated version of the Web Storage Sys based document store used in SharePoint Portal Server 2001. The backward-compatible document stc provided for users who require features from SharePoint Portal Server 2001, such as complex docu routing and approval, folder-level secuıity, minor-level version numbers, and multiple document profile each folder.

The primary interfaces for the document management features in SharePoint Products and Technologie document libraries, which you can add to any SharePoint site. A· document library consists of the vı folder where the files are stored," the files themselves, and the user-definable descriptive inform (metadata) associated with each item in the document library.

1.9.7 Organizing Documents and Other Content

You can configure each SharePoint site with a document library and a corresponding list component tha display customizable views of the metadata for each document.

With SharePoint Portal Server 2003, you can associate documents and other content in a site with one or more hierarchical areas. Areas provide an alternative way to navigate and search content in a SharePoint Portal Server 2003 portal site. Areas are similar to Categories in SharePoint Portal Server 2001.

In a SharePoint Products and Technologies configuration that uses only SQL Server content stores, major Version numbers are used to track document revisions (minor version numbers are not supported), and a single-step moderator approval mechanism is used to approve documents .

1.9.8 Search Configuration and Usage

Windows. SharePoint Services keeps all content in SQL Server content stores, and it uses the Microsoft Search full-text indexing and searching technology from Microsoft SQL Server. Because of this, Windows SharePoint Services can only index and search content in SQL Server content stores.

Sharel'oint Portal Server 2003 uses the latest version of Microsoft Search technology to index both local and external document collections and Web sites. It supports all of the advanced indexing and searching features from SharePoint Portal Server 2001, with improved performance, scalability, and extensibility. SharePoint Portal Server 2003 can now index and search approximately 20 million documents (a five-fold improvement)' and support load-balanced queries across multiple catalog servers.

Windows SharePoint.Sc:ryicc:sandSharePoint Portal Server 2003 provide easy-to-use sharing tools for your organization. You can use Windows SharePoint Services to create and maintain many team sites, and you can use SharePoint Portal Server 2003 to build and manage integrated, large-scale portal solutions.

To achieve this significant increase in capability, perf91111ance, stability, and security, the overall architecture of Microsoft SharePoint Products and Technologies includes many significant changes. The most important of these changes are the use of the .NET Framework, Windows Server 2003, and Microsoft SQL Server for content storage.

1.10 What's New in Microsoft Office SharePointPortal Server 2003

Microsoft Office SharePoint Portal Server 2003 is an update to Microsoft SharePoint Portal Server 2001, and offers a number of improvements, many of which are described in the following sections.

The following are a just few of the new features for users ofSharePoint Portal Server:

Areas You can organize information on the portal site by using areas. If you find a useful listing missing fiom an area, add a listing for a content manager to approve. You can add a listing to more than one area on the portal site.

News SharePoint Portal Server enables you to highlight information, such as announcements and other key company information, by adding listings to the News area. A news listing can be either text-based content or a link to an existing news item, such as a press release or an article on a news service.

Personal sites My Site is a personal SharePoint site that provides personalized and customized information for you. In addition, My Site provides quick access to things you need to do your work such as links to

documents, people, or Web sites as well as alerts to track changes.to content within the portal site and organization. From My Site, you can also update your user profile and share links with other portal site u

User profiles Easily find information about people, their documents, and their sh~r~d links.

Alerts Get alert results sent to you in e-mail immediately or in daily or weekly summaries for porta content. You can now add alerts for people, lists, list items, and the Site Directory in addition to news, topics, search queries, documents, and backward-compatible document libraries. Alert results are shown easy-to-read HTML format and now identify whether the alert result is sent because content changed o added. You can manage all of your alerts from the My Alerts page.

Lists and views Because SharePointPortal Server is built on Microsoft Windows SharePoint Services can add both predesigned and custom lists.to all SharePoint sites. For example, you. can Create. a pi library to share a collection of digital pictures or create an issue tracking list to maintain a history specific issue. You can also use calendarviews for any SharePoint list that has a date and time colurr addition, you can add attachments to list items, including HTML pages, documents, and images.

Simple site creation and page customization By using Self-Service Site Çr~atiori, You. can c SharePoint sites - such as team sites or Meeting Workspace sites - on defuaridiwithoutjnvolving tl department if you have the necessary permissions. In addition, you can customize a page by changii adding Web Parts. Each list and library on a portal site is a Web Part, enabling easy custornizatior personalization using the browser.

Search Faster results and improved relevancy ranking enable you to find the information you need e Search results now include people, picture libraries, list items, and user profiles. If you search for an ir you'll see a thumbnail view of the image; if you search for a person, you'll see his or her personal profile. can also group search results in different ways, such as by author, site, date, or area. From the search n page, you now caii save a useful search to the My Links Web Part ori your personal site.

Site Directory The Site Directory provides a central location from which to view and access all Web associated with a specific portal site. You can also create sites based on Windows SharePoint Services o links to existing sites. In addition, adding a site to the Site Directory is a quick and easy way to im content in search results.

The following are a just few of the new features for content managers:

Lists and views Because SharePoint Portal Server is built on Microsoft Windows SharePoint Services can add predesigned and custom lists to all SharePoint sites. List managers can approve or reject items

documents, people, or Web sites as well as alerts to track changes to content within the portal site and yo organization. From My Site, you can also update your user profile and share links with other portal site user

User profiles Easily find information about people, their documents, and their shared links.

Alerts Get alert results sent to you in e-mail immediately or in daily or weekly summaries for portal content. You can now add alerts for people, lists, list items, and the Site Directory in addition to news, an topics, search queries, documents, and backward-compatible document libraries. Alert results are shown ir easy-to-read HTML format and now identify whether the alert result is sent because content changed or added. You can manage all of your alerts from the My Alerts page.

Lists and views Because SharePoint Portal. Server is built on Microsoft Windows SharePoint Services, can add both predesigned and custom lists to all SharePoint sites. For example, you. can 8ı:~atea pie library to share a collection of digital pictures or create an issue tracking list to maintain a history specific issue. You can also use calendar views for any SharePoint list that has a date and time colum addition, you can add attachments to list items, including HTML pages, documents, and images.

Simple site creation and page customization By using Self-Service Site Çr~atiğn,)'oı.ı can c SharePoint sites - such as team sites or Meeting Workspace sites - orı clefuaricl withôuffüvolvi.rtgt' department if you have the necessary permissions. In addition, you can customize a page by charigi adding Web Parts. Each list and library on a portal site is aJVeb Patt,.i~t1aplfpgeasy p.ustomizatio: personalization using the browser.

Search Faster results and improved relevancy ranking enable you to find the information you need Search results now include people, picture libraries, list items, and user profiles. If you search for an i you'll see a thumbnail view of the image; if you search for a person, you'll see his or her personal profil can also group search results in different ways, such as by authôı:,site, date, or area. From the search page, you now carı save a useful search to the My Links Web Part di'i'yourpersonal site.

Site Directory The Site Directory provides a central location from which to view and access all Wt associated with a specific portal site. You can also create sites based on Windows SharePoint Services links to existing sites. In addition, adding a site to the Site Directory is a quick arid easy way to content in search results.

The following are a just few of the new features for contentmanagers:

Lists and views Because SharePoint Portal Server is built on Microsoft Windows SharePoint Servi can add predesigned and custom lists to all SharePoint sites. List managers can approve or reject it

Shared services Deliver shared services to multiple portal sitetfrom a centrally tr1a.tıaged and confij server farm. Shared services can include creating indexes and search, user profiles, audiences, alerts personal sites.

Communicate with external partners by using an extranet If you work with external partners, or i have users who need to access data from outside of your organization's firewall, you can use Shan Portal Server in an intranet/extranet environment. In this configuration, internal and external users caıı and interact with the same content and data. You can also employ the antivirus protection and blockı extension features to help protect your server integrity.

1.10.1.1 International

Support for multiple language sites Multiple language sites can be hosted on a single server or serve running SharePoint Portal Server. Note-that'site language is independent from server language.

Regional settings for each site Each site can have its own regional settings, such as time zone.

New word breakers Word breakers.for Czech, Finnish, Hungarian, and Portugıese are available, as the original set of SharePoint Portal Server 2001 word breakers for English, French, Spanish, Japanesı Korean, Chinese Traditional, and Chinese Simplified.

1.10.1.2 Management

Alerts The portal site now automatically identifies and optimizes alerts that have the potential for ger large numbers of results; it will deactivate any alert that generates an excessive number of Administrators can deactivate or delete any user's alerts and alert results. Misdirected e-mail message. prevented by locking e-mail address fields to use only user profile data. You can also customize the fc the alert results e-mail messages by using an .xsl file.

Single sign-on Single sign-on allows you to store and map account credentials so that users don't sign on again when portal-based applications retrieve information from enterprise applications.

Securely integrate enterprise applications Tight integration with Microsoft Biz'Talk Server 2002 rich and secure enterprise application integration using single sign-on. Connectors from Actiona integration with PeopleSoft, SAP, and Siebel.

Full-text searching The portal site delivers a scalable, high performance index creation and query infrastructure. By using a multiserver topology, you can manage your resources by propagating indexes from the index management server to multiple dedicated search servers. Creating indexes o

protocol enables crawling of Web sites over SSL. In addition, protocol handlers for Windows SharePoint Services sites enable the portal site to crawl information in site pages, document libraries, lists, and list items. Ifilters now provide the ability to full-text search files created by Microsoft Office Publisher (.pub) and Microsoft Office Visio (.vsd) in addition to the existing capability to search files in Microsoft Office Word (.doc), Microsoft Office Excel (.xls), Microsoft Office PowerPoint (.ppt), MIME, XML, and HTML formats.

Audiences Audiences allow organizations to target content to users based on their job role qr task. Target Web Parts, news, lists, and list items to one ôr more specific audiences. Use your investment in Microsoft Active Directory directory service to easily create Audiences from existing distribution lists and security groups.

Backup and restore Improved backup and restore enables flexible site recovery. Each site in a server farm can be individually backed up and restored. This feature can also be used for archiving inactive sites prior to deleting them.

User **profiles** Easily create user profiles by importing properties and user data from Active Directory. User profiles make it easy to find people and enable content managers to target information by using audiences. Add properties to the flexible user profile for use by integrated applications or to enable portal site users to find people more easily.

Inactive site management Site owners are periodically asked to confirm that their sites are in use or delete them. If multiple notices are sent to a site owner without any response, the administrator can specify that the site be automatically deleted.

1.10.1.3 Security

Standard Windows authentication and security methods You can use SharePoint Portal Server with any Microsoft Internet Information Services (IIS) 6.0 authentication method, connect to the database by using Microsoft Windows authentication or Microsoft SQL Server authentication, and integrate SharePoint Portal Server with Active Directory.

SharePoint administrators group Allow members of a domain group to perform central administration tasks without granting them administrator rights to the local server computer.

Manage users from SharePoint Central Administration Use the SharePoint Central Administration pages to add or delete users on all sites and assign site owners.

Domain group support Use domain groups to control access to your site.

Blocked file extensions Server administrators can block the upload of specific file types (for example, .r or .exe files).

For a complete list of new administrator features, and information about using these features, see Administrator's Guide for SharePoint Portal Server.

1.10.2 What's New in Microsoft Windows SharePoint Services 2.0

Microsoft Windows SharePoint Services 2.0 makes it easier to create sites to share information with ot with new types of sites, lists, and libraries. Managing your sites is also much easier with an enhanced secı model and tools for monitoring and controlling your sites.

The following features are just some of the additions and improvements for users of Windows Sharel Services.

Alerts Alerts improve on the subscription notifications in SharePoint Team. Services l.O ftorrı Mictc Windows SharePoint Services uses alerts to tell you through e-mail about additions, deletions, and chang lists, list items, libraries, and other parts of sites. You can receive alert results immediately, or request c or weekly alert results summaries.

Lists Issue lists, calendar views, group-by views, personal views, and rich text.expahdthe .possihiliti Windows SharePoint Services lists. You can use formulas and functions in lists to create calculated col and views. Creating a list is even easier from the one-stop Create page. Lists can be set up to require tl owner's approvalbefore new items appear.

Document libraries Document libraries now support versions-s-or creating a backup copy of a file whe you save a file to the library-as well as check in and check out, and subfolders, Windows Shan Services includes viewers for files that enable you to view documents from programs such as Mic Office 2003 even if you don't.have the program installed.

Picture libraries You can store photos and graphics in new picture libraries. View pictures as thuml filmstrips, or in a standard files list.

Meeting Workspace sites New Meeting Workspace sites deliver a place for managing meetings anc attendees, agendas, documents, decisions, and action items. Users can contribute to a Meeting Workspa using a browser. You can create a Meeting Workspace site from an e-mail program compatible with Wi Sharel'oint Services, such as Office Outlook 2003 or by using the browser from an events list.

Document Workspace sites New Document Workspace sites deliver sites that are centered around more documents. You can can easily work together with coworkers on a document - either by v.

directly on the copy located on the Document Workspace site or by working on your own copy, which you can update periodically with changes that are saved to the copy located on the Document Workspace site. You can create a Document Workspace site from a word processing program compatible with Windows SharePoint Services. For example, you can create a Document Workspace from Microsoft Office Word 2003, Office Excel 2003, Office PowerPoint 2003, as a Shared Attachment in Office Outlook 2003, or by using the browser from a document library.

Integration with other programs You can open files from, save files to, and upload multiple files to document libraries from productivity programs that are compatible with Windows SharePoint Services, such as Office 2003. You can link events and contacts information between Windows SharePoint Services and compatible calendar and contacts programs. Document Workspace sites integrate with Windows SharePoint Services-compatible e-mail, spreadsheet, presentation, and word processing programs to enable you to send documents in a Document Workspace site as attachments. Windows SharePoint Services-compatible spreadsheet and database programs enhance SharePoint lists in Datasheet views.

Online presence integration User presence is indicated everywhere a member name appears in a site. The presence menu integrates with Microsoft Active Directory directory service, Microsoft Exchange, and Microsoft Windows Messenger to offer information such as free/busy status, office location, and manager.

The following are just some of the new and improved features for people who manage a site or sites.

!Lists and libraries Windows SharePoint Services offers site administrators that ability to manage lists and libraries by doing the following:

Blocking specific file types from document libraries to prevent suspicious file types from being uploaded to the server

Linking a document library with a public folder based on Microsoft Exchange 2000 or later to store documents attached to e-mail messages

Specifying list permissions to allow only specific users to change a list

Customizing sites You can enable users to update the home page content and layout by using the Web Parts tool pane. Also, each list in a site is a Web Part that allows easy customization and personalization by using the browser. You can also manage themes and templates to make customization even easier.

Templates Templates can apply to entire sites, or individual lists and libraries. You can save lists as templates, and then reuse them or distribute them to other sites. You can save sites as templates to capture best practices or to define a consistent look and feel. You can store Web Parts, list templates, and site templates in libraries for use by all sites in the site collection.

International settings Windows SharePoint Services offers support for regional settings such as language, time zone, currency type, international calendar formats, and other locale settings on a per-site basis.

Site users You can use your organizational address book to choose the users to add to a site, and use domain groups to control access to your site. Windows SharePoint Services supports Microsoft Windows

authentication support for anonymous, basic, integrated Windows authentication, and certificati authentication.

Site creation You can enable self-service site creation to allow users to create sites on demand witho involving the IT department.

Tracking site use You can use usage information to determine .how many users visit each site. Sites owni are automatically notified if their site has been inactive for a specified period of time. If multiple notices sent to the site owner and the site remains inactive, the administrator can specify that the site be automatica deleted.

The following are some of the new and improved features for administrators of servers running Windc SharePoint Services.

Site collection management Top-level Web sites and subsites can divide site content into distinct separately manageable sites. These site collections support customized management tools and simpli management from a single list of all sites. Each site in a server farm can be individually backed up restored. You can set quotas for site storage size and generate automatic notifications for the site owner Vi a site reaches its size limit. You can use the SharePoint Central Administration pages to manage users 01 sites and create cross-site groups that can be used within all sites in a site collection.

Server topology You can install and configure Windows SharePoint Services to allow your server fan host several sites with the same Internet Protocol (IP) address, with multiple site names and separate con You can add multiple servers running Microsoft SQL Server to the server farm to suppoıt server clusters. can load balance your front-end Web servers by using a single back-end SQL Server c:oirıpıiter.

Site migration You can use the Microsoft SharePoint Migration Tool (Smigrate.exe) to move exi Windows SharePoint Services sites to another server. SharePoint Team Services vl .0 sites can be mov SharePoint sites as well.

Extensibility Sites are built on Microsoft ASP.NET technology and are extensible by using the Mier .NET Framework. You can use the object model to create custom solutions for SharePoint Product/ Technologies. You can use events in document libraries to build custom actions and send documents document library by using e-mail.

SQL Server integration When using Windows SharePoint Services with Microsoft SQL Server, ful indexing provides site-wide search. To take advantage of SQL Server backup and restore capabilities, Server is used to store all site data including documents.

International settings Windows SharePoint Services languages supports more languages than Shan Team Services vl .0. Multiple language sites can be hosted on a single server or server farın running Wit SharePoint Services. Site language is independent from server language.

1.10.3 About Index and Search Services

With the poweıful index and search services in Microsoft Office SharePoint Portal Server 2003, you can search for people, sites, documents, or other information stored in many different locations and in a variety of formats, SharePoint Portal Server makes searches faster and more successful by using these features:

A single location to search for information stored in many different places and in many different formats
Keyword searches that find people, sites, documents, images, or other types of information on the portal sitete
Tepic areas that make it easy for users who are unfamiliar with other areas of the portal site to find what they need
Automatic categorization of content
Best Bet tagging for items that are highly relevant to a search
Alerts that notify you of search results
User profiles that can be searched to find people by title, location, and other details they publish
Whether you are searching for specific information or just want to browse through a group of related items, SharePoint Portal Server makes finding information easy.

l.10.4 About Integrated Enterprise

Microsoft Office Sharel'oint-Portal Server 2003 provides an architecture that addresses the most demanding performance needs. By using the latest technologies, SharePoint Portal Server provides a centralized, unified interface for enterprise users and highly flexible deployment options.

Microsoft .NET enterprise, scalable, distributed architecture SharePoint Portal Server is built on the .NET Framework, which is fast and scalable and uses ASP.NET, common language runtime, Web Forms and Web Part Pages, and a secure infrastructure to deliver better performance and greater integration. Through its flexible deployment options - from single server to server farm configurations - SharePoint Portal Serveris designed to have high availability and manageability.

Business applications SharePoint Portal Server can present specific applications and customized content based on the user's functional group and organizational role,

Delegated administration The portal site administrator can assign different managers for areas in the portal site. The content manager can then control what content appears in the area and who has access to the area.

Single sign-on The single sign-on service is an authentication process that permits a user to enter one name and password to access multiple applications.

Index and search services These services help users find people, sites, documents, and other types of information from different sources.

Alerts Alerts notify users of updates to relevant information.

1.10.5 About Integrated Enterprise

Microsoft Office SharePoint Portal Server 2003 provides an architecture that addresses the most demand performance needs. By using the. latest technologies, SharePoint Portal Server provides a centralized, uni: interface for enterprise users and highly flexible deployment optiops.

Microsoft .NET **enterprise, scalable, distributed architecture** SharePoint Portal Server is .built on .NET Framework, which is fast and scalable and uses ASP.NET, common language runtime, Web Forms Web Part Pages, and a secure infrastructure to deliver better performance and greater integration. Throug flexible deployment options - from single server to server farm configurations - SharePoint Portal Ser designed to have high availability and manageability.

Business applications SharePoint Portal Server can present specific applications and customized co based on the user's functional group and organizational role.

Delegated administration The portal site administrator can assign different managers for areas in the l site. The content manager can then control what content appears in the area and who has access to the art

Single sign-on The single sign-on service is an authentication process that permits a user to enter one and password to access multiple applications.

Index and search services These services help users find people, sites, documents, and other tYJ information from different sources.

Alerts Alerts notify users of updates to relevant information.

1.10.6 About Collaboration

Microsoft Office SharePoint Portal Server 2003 facilitates easy, connected collaboration across an en' organization. It enables people to work together on documents, projects, and tasks and to levera practices by using the combined collaboration features of Microsoft SharePoint Products and Techru Index and search services, as well as newly introduced people services, allow you to increase efficii finding relevant people, teams, sites, and other information.

- Finding and organizing sites, people, and other information
- Index and search

Search SharePoint Portal Server provides a search feature that finds all types of content - documents, sites, and other items on the portal site - based on keywords you enter. Search can fine stored in different sources, such as Web sites, file systems, mail servers, and databases. The result organized in different ways, such as by site, area, or date. For a more specific search, you can

advanced search option to search by properties of items, to sort results by factors other than relevance; and to set other advanced search options.

Best Bets Best Bets enhance search efficiency and provide guidance to users by directing them to people, sites, documents, or other items considered particularly relevant to their search. SharePoint Portal Server displays Best Bets at the top of a search results list.

Site Directory The Site Directory is the easiest way to add content to the portal, site for searching. When a user adds a site, they have the option to include its contents in search results. A search administrator can have sites automatically approved for sear9hing or can manage approval for each site. After approval, a site is indexed and its contents appears in search results.

Alerts You can ask to be alerted when changes occur.to the results for a specific search.

To make it easy for users to navigate, browse, and find what they need, you can divide portal site content into areas. Areas let you organize content -- from documents to people to sites - into sets of related information even though the content can be stored in different sources and formats. To control all of the content in an area, the portal site administrator can assign a manager for the area: The manager can then control what content appears in the area and who has access to it.

User profiles User profiles allow you to search for and Connect with people within your organization based on information they publish about themselves. Index and search services use the profile information to improve search results.

My Site My Site is a personal SharePoint site created in the portal site that provides personalized and customized information for you, including content targeted to you based on your membership in a particular audience.

Audiences You can target content to a specific audience.based on a user's job or task. For example, in an area called Human Resources, a site administrator or the manager of that area can choose to add a news item targeted to all new employees that directs them to the New Employee Benefits site.

SharePoint Portal Server enables you to easily share information, leverage best practices, and work together with others on documents, projects, and other efforts. Some of the ways you can do that are by:
Win<lows SharePoint Services allows you to create a Web site by selecting a template that best suits the project. SharePoint Portal Server includes a diverse collection of templates to meet business needs. You can create sites to facilitate meetings, organization, teams, or projects. By default, each site template features a custom set of collaboration features from Windows SharePoint Services. If you work with external customers or partners, or if you have users who need to access data from outside of your organization's firewall, SharePoint Portal Server allows both internal and external users to view and interact with the same content

and data. After the sites are created, SharePoint Portal Server can search these sites the same way it search other content on the portal.

SharePoint Portal Server offers a number of features that make it easy to find, organize, and work together documents. A document library offers a central place to store documents and track changes, My Site is personal site where you can add links to documents you work with often, the Topics area lets you organ- documents under different topic headings, and Microsoft Windows SharePoint Services Document Workspaces offer a place for collaborating on documents with others.

1.10.7 About Document Workspace Sites

A Document Workspace site is a Microsoft Windows SharePoint Services site that is centered around one more documents. Colleagues can easily work together on a document — either by working directly on Document Workspace copy or by working on their own copy, which they can update periodically , changes that have been saved to the Document Workspace copy.

1.10.8 Creating a Document Workspace site

On a Windows SharePoint Services site, you can create a Document Workspace site either by going to Create page or by clicking the arrow next to a document on a document library page and then clicking Cr **a Document Workspace.**

When you create a Document Workspace site that's based on a document in a document library Document Workspace site carries the same name as the document on which it is based. The document is stored in a separate document library in the new Document Workspace site. This document can be publ back to its source location, in the original document library, from the Document Workspace site.

The ability to publish a document back to its original location is available only in a Document Workspac created from a document that is already stored in a document library. This feature is not availai Document Workspace sites created by using the Create page on a Windows SharePoint Services site

1.10.9 Document Workspace site features

Because a Document Workspace site is a SharePoint site, members can use Windows SharePoint Se features, such as using the Tasks list to assign each other to-do items, using the Links list to create hype to resource material, and storing related or supporting documents in the document library.

1.10.10 About Meeting Workspace Sites

The Meeting Workspace site is a special type of SharePoint subsite under a parent SharePoint site. H the Meeting Workspace site, as well as general Windows SharePoint Services Help, is available fr, Meeting Workspace site.

The Meeting Workspace site is made up of one or more pages that contain meeting details and information that are common when planning, conducting, or following up on a meeting. Typical lis Meeting Workspace site include Objectives, Agenda, Attendees, Decisions, and Tasks. In addition you can add a document library and a picture library where users can store materials related to the n The lists and libraries that appear by default on the home page depend on the template you choose wl create the Meeting Workspace site.

On the home page (or any new pages you add), the information is divided into parts called Web Parts. Web Parts exist for each type of list or library and for other types of information you can add to the Meeting Workspace site.

If the Meeting Workspace site doesn't contain all the information you need or you don't like the layout or look of the Meeting Workspace site you created, or if you need to automate certain actions on the Meeting Workspace site, you can make the changes you need programmatically. See Microsoft.SharePoint.Meetings Namespace, Meetings Web Service, and the programming tasks for help on working with Meeting Workspace sites programmatically .

1.10.11 About Picture Libraries

Picture libraries provide a simple way to share and organize digital pictures in a corporate server environment. For example, an organization could create a picture library for marketing graphics - providing a single location for team members to view, share, edit, and download corporate logos or other marketing material.

Although pictures can be stored in other types of lists in. Microsoft Windows® SharePoint™ Services, there are many advantages to using picture libraries. These advantages include:
Viewing pictures with one of three unique display styles.
Sharing pictures by using slide shows or by sending pictures directly to Microsoft Office 2003 programs.
Downloading pictures directly to a computer.
Editing pictures with Windows SharePoint Services-compatible image editors, such as Microsoft Office Picture Manager.

If you are familiar with Windows Share.PointServices doC:umeritlibraries, you will be familiar with picture libaries. The picture library is a special type of library ~11der1parent SharePoint site that contains pictures. Help for the picture library is available from the Windows SharePoint Services site.

You can also work programmatically with picture libaries. F.ofmore information, see Imaging Web Service and OISClientLauncherControl.

1.11 Security Overview for Microsoft SharePôiıt,Pfoducts and Technologies

1.11.1 Code Access Security

The Microsoft® SharePoint™ Products and Technologies2003 class libraries use code access security to help protect certain resources and operations by specifying the permissions they require in order to run. The menhers of the SharePoint class libraries demand SharePointPermission with the ObjectModel property set to true. You should make sure that assemblies you create that are clients of the classes in the SharePoint class lib:raries have this permission.
There are several ways to make sure your code has the required permissions to access the SharePoint class libraries, Options include the following:

Install the assembly in the Global Assembly Cache (GAC), because code in the GAC always has Full trust.

Raise the trust level for the virtual server extended with Microsoft Windows® SharePoint™ Services.

Create a custom security policy and assign the SharePoiıttPermission with the ObjectModel property set to true to the specific assembly or set of assemblies.

Whatever option you choose, make sure that you fully understand the security implications so that you do not expose a security weakness. For more information, see Code Access Security for Administrators.

If your code is granted the permissions demanded by the SharePoint class library, your code will be allo to access the library. Note that any assembly that calls your assembly that calls one or more of the ShareF assemblies will also need to be granted the same permission. All clients in the chain of callers leading u SharePoint class libraries need the **SharePointPermission** set as explained earlier; if your code does not l the appropriate permissions, it will not be allowed to access the class library. If other code uses your coc indirectly access the resource, it too must have the appropriate permissions to access the class library, or code will not be allowed to run. This helps prevent malicious code from trying to use your code to acces resource without permission.

1.11.2 Other permissions your code may require

If your code tries to write data to a database on an HTTP Get or if the code needs to enable Web Part-to-Part connections, your code might also need two other permissions specific to Microsoft Office Share Products and Technologies. They are: **SharePointPermission.UnsafeSaveOnGet** **WebPartPermission.Connections.**

Important For Microsoft Office SharePoint Portal Server only, if you want to use the classes and mer in the **Microsoft.SharePoint.Portal.SingleSignon** namespace, your code will need an additional permi called **SingleSignonPermission.Access.**

Your code mightalso need 011e or mor~ of the.default ASP.NET permissions if it tries.to perform an act access a resource that is protected by the common language runtime.

Code access..security does. not elimi11~!e the possibility of human .error in writing .fgclf h°\Ve., applications use secure class Jil.:ıraric~ to access protected resources, the seciırity.psk for.application.co< be decreased because class libraries are closely scrutinized for potential security problen:ıs.

For a complete understanding (>L code access security and how it is used in SharePoint Produc1 Technologies, see Code **Access Security for Developers.**

1.11.3 Code Access Security for Administrators

To help protect computer systems from malicious mobile code, to allow code from unknown origins safely, and to protect trusted code from intentionally or accidentally compromising security, the Mi, .NET. Framework provide~.a security mechanism called code access security. Microsoft Wint SharePoint™ Services inherits this feature from ASP.NET and uses it to control access to protected res and operations.

The ASP.NET.code access security feature lets you assign to a Microsoft® SharePointTM Produc Technologies application a configurable level of trust that corresponds to a predefined set of permissic default (unless you explicitly alter the configuration),applications receive a level of trust commensura the evidence they present. For example, local applications must run in the MyComputer zone with ti trust permission set, and applications located on a Universal Naming Convention (UNÇ).share must rui Intranet zone with the **Locallntranet** restricted permission set.

To allow the administrator to easily switch levels of trust assigned to an application, Windows Sha Services includes the ASP.Net default security policy files and a couple of policy files of its own. Code security can be easily handled for an individual assembly by strongly naming it and adding policy assembly. The administrator can associate the appropriate level of trust to the application wit web.config file of that application.

Code access security allows code to be trusted to varying degrees, depending on where the code or and on other aspects of the code's identity. Code access security also enforces the varying levels of

code, which minimizes the amount of code that must be fully trusted ill order to run. Using code access security can reduce the likelihood that your code can be misused by malicious or error-filled code. It can reduce your liability because you can specify the set of operations your code should be allowed to perform as well as the operations your code should never be allowed to perform. Code access security can also help minimize the damage that can result from security vulnerabilities in your code.

Code access security is a mechanism that controls the access to protected resources and operations through code. In Windows SharePoint Services, code access security performs the following functions:

Defines permissions and permission sets that represent the right to access various system resources.

Enables administrators to configure security policy by associating sets of permissions with groups of code (code groups).

Enables code to request the permissions it requires in order to run, as well as the permissions that would be useful to have, and specifies which permissions the code must never have.

Grants permissions to each assembly that is loaded, based on the permissions requested by the code and on the operations permitted by security policy.

Enables code to demand that its callers have specific permissions.

Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particular organization or site to call the protected code.

Enforces restrictions on code at run time by comparing the granted permissions of every caller on the call stack to the permissions that callers must have.

To determine whether code is authorized to access a resource or perform an operation, the runtime's security system walks the call stack, comparing the granted permissions of each caller to the permission being demanded. If any caller in the call stack does not have the demanded permission, a security exception is thrown and access is refused. The stack walk is designed to prevent luring attacks, in which less-trusted code calls highly trusted code and uses it to perform unauthorized actions. Demanding permissions of all callers at run time affects performance, but is essential to protect code from luring attacks by less-trusted code. To optimize performance, you can have your code perform fewer stack walks; however, you must be sure that you do not expose a security weakness whenever you do this.

Windows SharePoint Services includes two security policy files of its own, Windows SharePoint Services Minimal **(WSS_Minimal)** and Windows SharePoint Services Medium **(WSS_Medium).** When a virtual server is extended to host Windows SharePoint Services, the **WSS_Minimal** policy is applied to it. Both these policies extend from ASP.NET default policy files. These policies grant assemblies and pages in the Global Assembly Cache (GAC), $CodeGen, and _layouts and _vti_bin roots **Full** trust but only partially trust the assemblies installed in the Bin directory of the virtual server.

Table 1.2

Permission	Full	WSS_Medium	WSS_Minimal
AspNetHostingPermission	Full	Medium	Minimal
Environment	Unrestricted or all subpermissions granted	Read: TEMP, TMP, USERNAME, COMPUTERNAME	OS,

IFile!o	
IsolatedStorage	ll~restricted or llsubpennissions granted AssemblylsolationI3yUser, UserQuota specified
Reflection	~restricted 0, subpermıssıons granted
IRe~stry	~;~:on:1
Security	Unrestricted or all subpermıssıons granted Execution, Assertıon,ı ControlPrincipal, ControlThread,IIExecution RemotingConfiguration
Socket	Unrestricted or all subpermıssıons granted
WebPermission	Unrestricted or all subpermissions granted Connect to origin host configured)
DNS	~~restri~te~ orj11unrestricted. or all subpermissions subpermıssıons granted granted
Printing	~~brestricted or Default printing subpermıssıons granted
OleDBPennission	~1'°,tn'.'ed subpermıssıons granted
SqlC!ientPermission	IIUnrestricted or all subpermissions granted Unrestricted or all subpermissions granted
Eventl.og	Unrestrıcted or ~~~permissions granted
Message Queue	~r"tn'.'ed 0, subpermısswrı_s_----- -------- -- -- -- -------

Service Controller	[granted Unrestricted or ~,";;issions		
Performance Counters	Unrestricted o, ;;;\',";;ssions		
Directory Service	~r.~mct,d o, subpermissions granted		
SharePointPermission	Unrestricted or sub permissions granted	SharePointPermission.ObjectModel	
WebPartPermission	Unrestricted or allb permissions granted	IIWebPartPermission.Connections	WebPartPermission.Connections

1.11.4 Microsoft® SharePoint™ Products and Technologies Permissions

In addition to the default set of permissions defined by ASP.NET and the common language runtime, Microsoft® SharePoint™ Products and Technologies has defined two important custom permissions, **SharePoiiitPermission** and **WebPartPermission.**

All code J~at tries to access the Microsoft® SharePoint™ Products and Technologies class libraries need the **SharePointPermission** with the ObjectModel property set to **true.**

If your code tries to write data to a database on an HTTP Get or if the code needs to enable Web Part to Web Part conftections,your code might also need two other permissions specific to Microsoft SharePoint Products and Technologies. They are: **SharePointPermission.UnsafeSaveOnGet** and **WebPartPermission.Connections.**

For Microsoft Office SharePoint Portal Server only, if you want to use the classes and members in the **Microsoft.SharePoint.Portal.SingleSignon** namespace, your code will need an additional permission called **SingleSignonPermission.Access.**

Your code might also need one or more of the default ASP.NET permissions ifit tries to perform an action or access a resource that is protected by the common language runtime CLR.

If your code requests and is granted the permissions required by the class library, your code will be allowed to access the library and the resource will be protected from unauthorized access; if your code does not have the appropriate permissions, it will not be allowed to access the class library and malicious code will not be able to use your code to indirectly access the resource. Even if your code has permission to access a library, it will not be allc,wed to run if code that calls your code does not also have permission to access the library.

There are several ways to make sure your code has the required permissions to access the Microso SharePoint™ Products and Technologies class libraries. One option is to consider installing the assembl: the Global Assembly Cache as the code in it always has **Full** trust. Another option is to consider raising trust level for the virtual server extended with SharePoint. Yet another option is to create a custom sect policy and assign the **SharePointPermission** with the **ObjectModel** property set to **true** to the spe assembly or set of assemblies.

Based on requirement and the trust associated with assemblies installed in the _bin directory of the vi server, administrators may choose to change the permissions associated with .these assemblies. An approach is to change the policy applied to the virtual server, thereby raising or lowering the permisi granted to all the associated assemblies together. This can be done by changing the trust level attribute i web.config file of the specific virtual server. The default levels of trust available to the user are:

Full

High (This is an ASP.NET default level, so the **SharePointPermission** and **WebPartPermission** will granted.)

Medium (This is an ASP.NET default level, so the **SharePointPermission** and **WebPartPermission** v, be granted.)

WSS_Medium

Low (This is an ASP.NET defaırlt-Ievel, so the **SharePointPerıiission** and **WebPartPermission** will granted.)

Minimal (This is an ASP.NET default level, so the **SharePointPerırilssion** and **WebPartPermission** be granted.)

WSS_Minimal

Open the web.config file of the virtual server in a text editor such as Notepad.

Search for: <trust level="WSS_Minimal" originUrl="**/>**.

Change it to: <trust level="WSS_Medium" originUrl=""/>.

Save the web.config file.

Reset Internet Information Services (IlS) by using **iisreset** in the command prompt.

The trust level of the given application has now successfully been changed to **WSS_Medium.**

Changing trust levels for the virtual server is an easy approach to granting or revoking permission also means that administrators need to trust all the assemblies equally. There may be situations in administrator may want to either grant more permissions or reduce the number of permissions specific assemblies. To do this, you must create a custom policy.

For example, let's say that you want only a specific assembly (Name: TestParts.dll, PublicKeyBlob: "XXXX") to have the **SharePointPermission.ObjectModel** permission and the rest to have the WSS_Minimal policy applied.

To retrieve the assembly's public key blob, which is different from its public key token, use the **secutil** tool as follows: secutil.exe -hex -s Testl'arts.dll. To learn more, see the MSDN topic on Secutil Tool.

Make a copy of the WSS_Minimal policy file at "<Local Drive>\My Custom Policies" and call it say "MyCustomPolicy.config".(The WSS_Minimal policy file is located by default at "<Local Drive>:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\config\wss_minimaltrust.config").

Now define a reference to the **SharePointPermission** class under the <SecurityClasses> node;

```
<SecurityClasses>
  â€¦ SecurityClass nodes defined in WSS_Minimal â€¦
<SecurityClass Name="SharePointPermission" Description="Microsoft.SharePoint.Security.SharePointPermission,
  Microsoft.SharePoint.Security, Version=XXXX, Culture=XXXX, PublicKeyToken=XXXX"/>
  </SecurityClasses>
```

Replace **-XXXX** for the version, culture and publickeytoken with the appropriate values.

Now find the "ASP.Net" PermisisonSet and copy it and rename it to "MyCustomPermissions" as shown below.

```
<NamedPermissionSets>
â€¦ PermissionSet nodes defined in WSS_Minimal â€¦
  <PermissionSet
  class="NamedPermissionSet"
  version="1"
  Name="ASP.Net">
â€¦ IPermission nodes defined in WSS_Minimal for the ASP.Net PermissionSetâ€¦
  </PermissionSet>
  <PermissionSet
  class="NamedPermissionSet"
  version="1"
  Name="MyCustomPermissions">
â€¦ IPermission nodes defined in WSS_Minimal for the ASP.Net PermissionSetâ€¦
  </PermissionSet>
</NamedPermissionSets>
```

Now add a reference to **SharePoint.ObjectModel** permission under the "MyCustomPermissions" PermissionSet.

```
<NamedPermissionSets>
```

```
    â‹¯ PermissionSet nodes defined in WSS_Minimal â‹¯
<PermissionSet
    class="NamedPermissionSet"
    version="1"
    Name=" MyCustomPermissions">
â‹¯ IPermission nodes defined in WSS_Minimal for the ASP.Net PermissionSetâ‹¯
        <IPermission class="SharePointPermission"
        version="1"
        ObjectModel="True"
        />
    </PermissionSet>
</NamedPermissionSets>
```

Once the permission set is defined, we need to create a code group that assigns the permission to the s assembly. In the default policy files shipped by ASP.NET, the "AJICode" code group "FirstMatchCodeGroup", so the common language runtime will stop assigning permissions to an asi after the first match to a specific code group. As a result, you must declare the specific code group asi custom permissions to "TestParts" as the first code group within the "AJICode" group. The common la runtime will then assign the "MyCustomPermissions" Permission set and stop at it without proceeding default "$AppDirUrl$/*" code group that is used to assign permissions based on the fact that the asse located in Bin directory.

```
<CodeGroup
class="FirstMatchCodeGroup"
version="1"
PermissionSetName="Nothing">
    <IMembershipCondition
    class="AllMembershipCondition"
    version="1"
                />
          <CodeGroup
          class="UnionCodeGroup"
version="1"

          PermissionSetName="MyCustomPermissions">
            <IMembershipCondition
              class="StrongNameMembershipCondition"
```

```
            version="XXX"
            PublicKeyBlob="XXX"
            Name="TestParts"
            />
        </CodeGroup>
â¦CodeGroup nodes defined in WSS_Minimalâ¦
</CodeGroup>
```

Replace "XXXX" for the version and PublicKeyBlob with the appropriate values.

Save your changes. The policy file is ready to be used.

Once you create the policy file, you must create a trust level that can be used to assign the defined policies. To do this, add the following line in the web.config file of the virtual server.

Search for:

```
securityPolicy>
<trustLevel name="WSS_Medium" policyFile="C:\Program Files\Common Files\Microsoft Shared Debug\Web Server Extensions\60\config\wss_mediumtrust.config" />
<trustLevel name="WSS_Minimal" policyFile="C:\Program Files\Common Files\Microsoft Shared Debug\Web Server Extensions\60\config\wss_minimaltrust.config" />
</securityPolicy>
```

Replace with:

```
<securityPolicy>
<trustLevel name="WSS_Medium" policyFile="C:\Program Files\Common Files\Microsoft Shared Debug\Web Server Extensions\60\config\wss_mediumtrust.config" />
<trustLevel name="WSS_Minimal" policyFile="C:\Program Files\Common Files\Microsoft Shared Debug\Web Server Extensions\60\config\wss_minimaltrust.config" />
<trustLevel name="MyCustomLevel" policyFile="C:\My Custom Policies\MyCustomPolicy.config" />
</securityPolicy>
```

Once the trust level is created, you can assign it by further editing the web.config file.

Search for: <trust level="WSS_Minimal" originUrl="" />

Change it to: <trust level="MyCustomLevel" originUrl="" />

Save the web.config file.

Reset IIS by using "iisreset" at the command prompt.

Your custom policy has now been applied to the given virtual server.

For more information, see **Creating and deploying custom security policies** earlier in this topic.

1.11.5 Code Access Security for Developers

To help protect computer systems from malicious mobile code, to allow code from unknown origins to n safely, and to protect trusted code from intentionally or accidentally compromising security, the Microso .NET Framework provides a security mechanism called code access security. Microsoft Windows SharePoi Services inherits this feature from ASP.NET and uses it to control access to protected resources ar operations.

The ASP.NET code access security feature lets you assign to a Microsoft Windows SharePoint Service application a configurable level of trust that corresponds to a predefined set of permissions. By default (unle you explicitly alter the configuration), applications receive a level of trust commensurate with the evidem they present. For example, local applications must run in the MyComputer zone with the **Full** trust permissio set, and applications located on a Universal Naming Convention {UNC) share must run in the Intranet zor with the **Locallntranet** restricted permission set.

To allow the administrator to easily switch levels of trust assigned to an application, Windows SharePoi Services includes the ASP.Net default security policy files, as well as policy files of its own. Code acce security can be easily handled for an individual assembly by strongly naming it and adding policy for th assembly. The administrator can associate the appropriate level of trust to the application within tl web.config file of that application. See Code Access Security for Administrators for more information.

Code access security allows code to be trusted to varying degrees, depending on where the code originate and onother aspects of the code's identity. Code access security also enforces the. var~ng levels of trust. o code, which minimizes the amount of code that must be fully trusted. in order to run, ysiıng cod.e acces security can reduce the likelihood that. your code can be misused by malicious. or error-filled code. ı.t ca reduce your liability because you can specify the set of operations your code should be allowed to perform ~ well as the operations your code should never be allowed to perform. Code access security can· also he minimize the damage that can result from security vulnerabilities in your code.

Code access security is a mechanism that controls the access to protected resources and operations throug code. In Windows SharePoint Services, code access security performs the following functions:
Defines permissions and permission sets that represent the right to access various system resources.
Enables administrators to configure security policy by associating sets of permissions with groups of cod (code groups).
Enables code to request the permissions it requires in order to run, as well as the permissions that would b useful to have, and specifies which permissions the code must never have.
Grants permissions to each assembly that is loaded, based on the permissions requested 'by the code and o the operations permitted by security policy.
Enables code to demand that its callers have specific permissions.
Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particula organization or site to call the protected code.
Enforces restrictions on code at run time by comparing the granted permissions of every caller on the cal stack to the permissions that callers must have.
To determine whether code is authorized to access a resource or perform an operation, the runtime's securir system walks the call stack, comparing the granted permissions of each caller to the permission bein demanded. If any caller in the call stack does not have the demanded permission, a security exception i thrown and access is refused. The stack walk is designed to prevent luring attacks, in which less-trusted cod calls highly trusted code and uses it to perform unauthorized actions: Demanding permissions of all callers a run time affects performance, but is essential to protect code from luring attacks by less-trusted code. T< optimize performance, you can have your code perform fewer stack walks; however, you must be sure tha you do not expose a security weakness whenever you do this.

The SharePoint Products and Technologies class libraries use code access security to protect certain resource and operations by specifying the permissions they require in order to run. The members of the SharePoir

class libraries demand SharePointPermission with the ObjectModel property set to true. You should take a note of this and ensure that any assemblies you create (that are clients of the classes in the SharePoint class libraries) have this permission.

There are several ways to make sure your code has the required permissions to access the SharePoint class libraries. One option is to consider installing the assembly in the Global Assembly Cache as the code in it always has Full trust. Another option is to consider raising the trust level for the virtual server extended with Sharel'oint, Yet another option is to create a custom security policy and assign the SharePointPermission with the ObjectModel property set to true to the specific assembly or set of assemblies. Whatever option you choose, make sure that you do not expose a security weakness. For more information, see Code Access Security for Administrators.

If the code is granted the permissions demanded by the SharePoint class library, your code will be allowed to access the library. Be aware that any other assembly that calls your assembly to call one or more of the Microsoft SharePoint Products and Technologies assemblies will also need to be granted the same permission. All clients in the chain of callers leading up to SharePoint class libraries will need the SharePointPermission.ObjectModel permission as explained earlier; if your code does not have the appropriate permissions, it will not be allowed to access the class library and malicious code will not be able to use your code to indirectly access the resource. In other words, even if your code has permission to access a library, it will not be allowed to run if code that calls your code does not also have permission to access the library.

Other permissions your code may require:

If your code tries to write data to a database on an HTTP Get *ot* ifthe code needs to enable Web Part to Web Part connections, your code might also need two other permissions Specificto Microsoft SharePoint Products and Technologies. They are: SharePointPermission.UnsafeSaveOnGet and WebPartPermission.Connections.

]Important For Microsoft Office SharePoint Portal Server only, if you want to use the classes and members in the Microsoft.SharePoint.Portal.SingleSignon namespace, your code will need an additional permission called SingleSignonPermission.Access.

Your code might also 11eed one or more ofthe default ASP.N~Tpermissions if it tries to perform an action or access,a resource th.atis protected by the common languageruntime.

Code access security. does not eliminate the possibility of human error in wrıtıng code; however, if applications use secure class libraries to access protected resources, the security risk for application code is decreasedbecause class libraries are closely scrutinized for potential security problems.

Programming errors in class libraries can expose security vulnerabilities because class libraries often access protected resources and unmanaged.code. If you design class libraries, you need to understand code access securityand be careful to secure your class library.

Table 1.3 Table describes the three main elements you need to consider when securing a class library.

Security element	Description
Security demand	Demands are applied on the class and method level as a mechanism for requiring that callers of your code have the permissions that you want them to have. Demands invoke a stack walk, in which all callers that directly or indirectly call your code are checked on the stack when your code is called. Demands are usually used in class libraries to protect resources.

Security override	Overrides are applied on the class and method scope as a way to overrule certain secu decisions made by the runtime. They are invoked when callers use your code. They are u to stop stack walks and limit the access of callers who have already been granted cer permissions.
	Caution Overrides can be dangerous and should be used with care.
Security optimization	A combination of demands and overrides can enhance performance during the interactio your code and the security system.

Web Parts are ASP.NET server controls that are run on a user's computer on demand when the user acces Web site that contains them. They are deployed as assemblies either in the Global Assembly Cache (dA< in the Bin directory of the virtual server that is extended with Windows SharePoint Services. From a access security perspective, there are two types of Web Parts: those that are run underfhe default sec policy and those that require higher trust.

To write w.eb Parts that are intended to run under the default security policy, you only need to .know operations are allowed by .the default security policy for the intranet or the Internet zones. As long as a Part does not require more permission to execute than it receives as a result of its zone of origin, it wil (Keep in mind that an administrator or user might decide not to grant as much permission to code fro Internet or intranet zones.) To execute Web Parts that demand higher trust, the administrator or user adjust the security policy of any computer that will run the code.

High-trust Web Parts are intended to run under a less-restrictive security policy than their origin (intra the Internet) would typically warrant. Most permission demands made by secure libraries, s.uch as Wi SharePoint Services classes, perform stack walks that check .all callers to ensure that they have been g the demanded permission and that Web pages, although not managed code, are treated as callers : purposes of security. Stack walks are performed to prevent less-trusted code from luring highly truste into performing malicious operations.

Because Web Parts hosted in a browser can be manipulated by active script on a Web page, t~at Web considered a caller and checked during a security stack walk to prevent malicious Vleb-page autho1 exploiting more highly trusted code. The consequence of treating a Web page as a caller is that a granted a high level of trust based on its strong name or publisher certificate .and rnn. from a Web page prevented from performing operations not normally allowed to code originating fromfüe s~me zoru Web page itself (intranet or Internet). For more information about deployment considerations, following section. On the face ofit, this w?uld seem to make writing high-trust Web Parts impossi code access security provides for this scenario by enabling you to selectively override the security sta behavior.

High-trust Web Parts must make judicious use of Asserts to short-circuit stack walks for permissi their callers (the Web pages from \.Vhi.ch they were run) would not normally have. When you use Ass must be careful not to expose dangerous APIs that would allow malicious· Web pages to inappropriate operations. For this reason, the level of care and Security consciousness that must writing a high-trust Web Part approaches that required to write a secure class library.

High-trust Web Parts should.always be. strongly nam.e.d or signed with a publisher certificate (X.5 allows policy administrators to grant higher trust to these Web Parts without reducing their seer regard to other intranet/Intemet code. After the asseınbly is signed, the user must create a new cc associated with sufficient permissions, and spe~ify th.at .9nly code signed by the user's. cor organizationbe allowed membership to the code gropp./.A_~~r the security policy is modified in thi the highly trusted control will receive sufficient permisşiorıto execute.

Because security policy must be modified to allow hi!thitfti.St, downloaded Web Parts to function, this type of webpart is much easier on a corporate intranet for which there is typicaJly an

administrator who can deploy the described policy changes to multiple client computers. In order for high-trust Web Parts to be used over the Internet by general users with no common corporate or organizational affiliation, there must be a trust relationship between the control publisher and the user. Ultimately, the user must be comfortable using the publisher's instructions to modify the policy and to allow the high-trust Web Part to execute. Otherwise, the Web Part will not be allowed to run.

1.11.6 Platform

Microsoft® SharePoint™ Products and Technologies SDK contains the Microsoft Windows® SharePoint Services SDK.

By using the information about the languages, protocols, and technologies provided in the Windows SharePoint Services SDK, you can build upon the Windows SharePoint Services platform for Web development.

CHAPTER2

2.1 Microsoft Windows SharePoint Services

The Microsoft® Windows® SharePoint™ Services 2003 Software Development Kit (SDK) prov information about the languages, protocols, and technologies that you can use to customize a Web sil Windows SharePoint Services. This SDK is part of the larger Microsoft® SharePoint Products Technologies 2003 Software Development Kit.

This part of the Windows SharePoint Services SDK contains the following sections:

Concepts and Architecture
Programming Tasks
Reference
Security Practices
Appendix

You can use the Windows SharePoint Services object model in code running on the server to access and with list or Web site data and templates, to customize Web Part views of list data, or to manage a top-site or a subsite. The object model is fully integrated with Microsoft ASP.NET so that you can worl example, within the Microsoft Visual Studio® .NET integrated development environment (IDE) to de· custom Web applications or to customize Web Parts for use in relation to a Windows SharePoint Ser deployment. The object model provides extensive access to list data, list views, administration, and almc other aspects of a SharePoint site. For more information on the object model, see Namespaces in the Win SharePoint Services Object Model. For information on Web Parts, see Web Part Pages and the Wel Infrastructure.

The **Windows SbarePoint Services Web Service** provides interfaces for communicating with the ı running Windows SharePoint Services from a remote computer to work with list and site data. Through interfaces, you can use SOAP messaging to update list data, or you can create a Windows Application Visual Studio® .NET IDE that uses Web service methods to interact with a site. In addition to proı general access to lists and Web sites, the services provide, for example, access to Meeting Workspaces events lists or to Document Workspace sites.

This SDK also provides information about Collaborative Application Markup Language (CAML), whir be used within SOAP messaging and also within C#, Microsoft Visual Basic® .NET, or scripting lang to create, for example, queries or views on list data. CAML is also used in templates and in site definitiı the front-end Web server or servers. See Introduction to Templates and Definitions for information kinds of customization that are possible.

The SDK also documents protocols and various client-site APIs that can be used in the context of Wi SharePoint Services.

2.2 Concepts and Architecture

This section contains the following overviews about programmability in Microsoft Windows Shaı Services:
Security, Users, and Groups Overview
Namespaces in the Windows SharePoint Services Object Model
Programming with the Microsoft.SharePoint and Microsoft.SharePoint.Administration Namespaces

Programming with the Microsoft.HtmlTrans.InterfaceNamespace
Server and Site Architecture Overview
Introduction to Templates and Definitions
Guidelines for Templates and Definitions
Introduction to the Windows SharePoint Services Web Service
Guidelines for Using the Object Model
Major Schema Files

2.2.1 Security, Users, and Groups Overview

In Microsoft Windows® SharePoint™ Services 2.0, access to Web sites is controlled through a membership system by which each user is associated directly or indirectly with a permission that controls the specific actions that the user can perform. Windows SharePoint Services provides the ability to control site access through the following uses of permissions:

Site groups specify which users can perform specific actions in a site. Each user is a member of at least one site group, and each site group possesses corresponding rights. You can edit the rights assigned to a site group, create a site group with a custom set of rights, or delete an unused site group. The rights for the Administrator site group and Guest site group cannot be modified.

Site groups are defined per Web site. Users assigned to the Administrator site group are administrators only for a particular Web site. To perform any administrative tasks that affect settings for all Web sites and virtual servers on the server computer, a user must be an administrator for the server computer (also known as a local administrator) or a member of the SharePoint administrators group, rather than a member of the Administrator site group for the site.

Cross-site groups consist of a group of users and are assigned to a site group on any Web site in a site collection. There are no cross-site groups defined by default in Windows SharePoint Services.

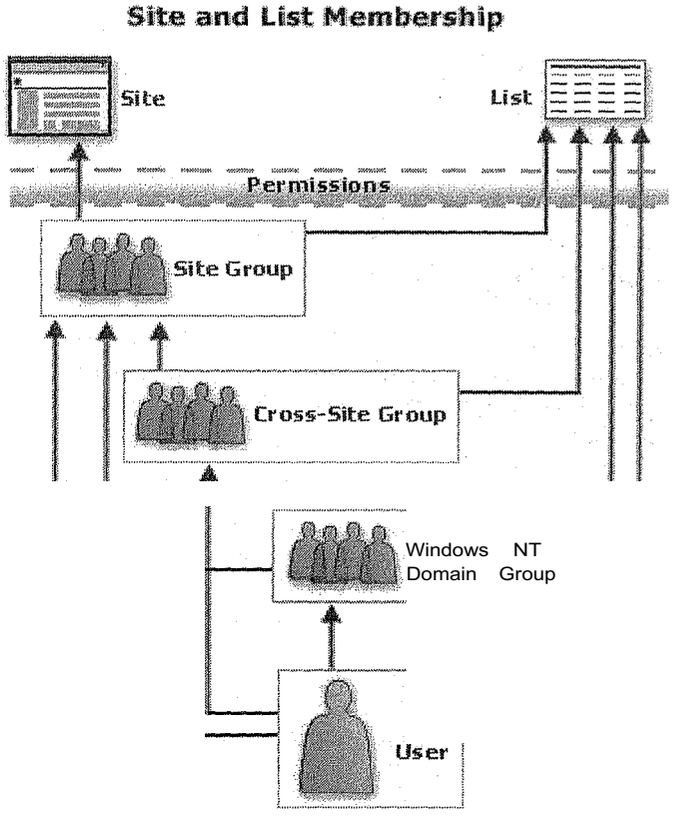
Anonymous access allows users to contribute anonymously to lists and surveys, or to view pages anonymously. You can also grant access to "all authenticated users" to allow all members of your domain to access a Web site without having to enable anonymous access.

Per-list permissions allow finer management of permissions by setting unique permissions on a per-list basis. Unlike for sites, you can add users together with specified permissions directly to a list, in which case the users are automatically assigned to the Guest site group on the current site if the site is unique and does not inherit permissions from a parent site. If the current site inherits permissions, the users are added to the Guest site group on the most recent unique ancestor site.

Subsites can either use the same permissions as the parent Web site (inheriting both the site groups and users available on the parent Web site), or use unique permissions.

Site creation rights **(CreateSSCSite** and **ManageSubwebs)** control whether users can create top-level Web sites, subsites, or workspaces.

For more information about rights and site groups, see the **SPRights** and **SPRoleType** enumerations.
The following diagram shows the means by which users become members of a site or list.



Figüre 2.1

As the diagram illustrates, users become members of a site through direct or indire.ct membership in group. They can be added directly to a site group or added to a cross-site group that is a member of group, or they can be members of a Microsoft Windows NT® Domain Group that is added to a site l The diagram also shows thata user can be directly added to .a list in association with a specified permi Each user, site group, or cross-site group has a unique member **ID.**

The permission for a user or group consists of a single right or set of rights that corresponds to .values **SPRights** enumeration and that forms a *permission mask* For more information, see the **Pfrmissior** property. To run custom code that uses types and members in the SharePoint object model', users and must be assigned the appropriate permissions, just as when interacting with a site or list by using tl interface. However, unlike the user interface, rights are not dependent on other rights in the object Rights can be assigned individually without including dependent rights, and they can be assigned to us groups in any combination. Be careful when customizing permissions through the object model, l: assigningrights inappropriately can lead to an unpleasant user experience.

In addition to using the membership system to control access to sites, Windows SharePoint Servic makes use of the following technologies that affect the security of a site:

User authentication- The process based on Internet Information Services (US) authentication methı is used to validate a user account that attempts to gain access to a Web site or network resource.

SharePoint administrators group - A Microsoft Windows user group authorized to peıform admin tasks for Windows SharePoint Services. To install Windows SharePoint Services, you must be a me the local administrators group on the server computer. However, in addition to the local administrator

you can identify a specific domain group to allow administrative access to Windows SharePoint Services. You can add users to this group rather than to the local administrators group, to separate administrative access to Windows SharePoint Services from administrative access to the local server computer.

Members of the SharePoint administrators group do not have access to the IIS metabase, so they cannot perform the following actions:
Extend virtual servers. They can, however, create top-level Web sites or change settings for a virtual server.
Manage paths.

Change the SharePoint administrators group.
Change the configuration database settings.
Use the Stsadm.exe command-line tool.

Members of both the Shai-ePoint administrators group and the local administrators group have rights to view and manage all sites created on their servers. This means that a server administrator can read documents or list items, change survey settings, delete a site, or perform any action on a site that a user who is a member of the Administrator site group for a site or site collection can perform.
Administrative port security - A means of controlling access to the administrative port for Windows SharePoint Services. You can help secure the administrative port by using Secure Sockets Layer (SSL) security or by configuring the firewall to not allow external access to the administration port, or both.

Microsoft SQL Server™ connection security - A way to help secure data. Use either Windows NT Integrated authentication or SQL Server authentication to connect to the configuration database and content database.

Firewall protection - A firewall helps protect data from exposure to other people and organizations on the Internet. Windows SharePoint Services can work inside or through the firewall of an organization.

For more information about the technologies that affectsecufity in Windows SharePoint Services, see the Administrator's Guide for Windows SharePoint Services.

2.2.2 Namespaces in the Windows SharePôinfServices Object Model

The Windows SharePoint Services object model consi~ts pf ei8ptee~ namespaces in nine assemblies that are used in SharePoint sites on the server that is running Windows SharePoint Services. The following table lists the namespaces, identifies their assemblies, and briefly describes namespaces that can be used to customize a Windows SharePoint Services deployment

Table 2.1

Name	[Assembly	][Description
Microsoft.HtmlTrans.Interface	Microsoft HtmlTrans Interface (in Microsoft.HtmlTrans.Interface.dll)	Provides an interface for custom conversion applications that deliver HTML versions of documents to users who do not have the required. client application or viewer installed on their computers.
Microsoft.SharePoint	Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides types and members for working with a top-level site and its subsites or lists.
Microsoft.SharePoint.Administration	Windows Shar~Point Services (in Microsoft.SharePoint.dll)	Provides types and members for managing a Windows Share:PointServi~~sdeployment.
Microsoft.SharePoint.ApplicationPages	Windows SharePoint Services (in Microsoft.SharePoint.dll)	The types and members of this namespace support Microsoft Windows

		SharePoint Services arıd are n intended to be used directly from yo, code.
Microsoft.SharePoint.Application Runtime	Windows SharePoint Services (in Microsoft.SharePoint.dll)	The types and members of th namespace support Microsoft Windov SharePoint Services and are n intended to be used directly from yo code.
Mtcrosoft.Sharef'eint.Dsp	Microsoft.SharePoint.Dsp (in Microsoft.SharePoint.Dsp.dll)	Provides the base class for the ds retrieval service adapters used Microsoft Windows SharePo· Services.
Microsoft.SharePoint.Dsp.OleDb	Microsoft.SharePoint.DspOleDb (in Microsoft.SharePoint.Dsp.OleDb.dll)	Provides the data retrieval servi adapter for performing queries agaii OLE DB data sources.
Microsoft.Sharef'oint.Dsp.Soapl'T	Microsoft.SharePoint.DspSoapPT (in Microsoft SharePoint.DspSoapPT.dB)	Provides the data retrieval serv adapter for performing pass-throu queries against arbitrary Web service:
Microsoft.SharePoint.Dsp.Sts	Microsoft.SharePoint.Dsp.Sts (in Microsoft.SharePoint.Dsp.Sts.dll)	Provides the data retrieval serv adapter for performing queries agai sites, lists, and document libraries Microsoft Windows SharePc Services.
Microsoft.ShareP oint.Dsp.XmlU rl	Microsoft.SharePoint.Dsp.XmlUri (in Microsoft.SharePoint.Dsp.XmlUrl.dll)	Provides the data retrieval serv adapter for performing queries agai arbitrary XML data sources.
Microsoft.SharePoint.Library	Microsoft.SharePoint.Library (in Microsoft.SharePoint.Library.dll)	The types and members of t namespace support Microsoft Windc SharePoint Services and are intended to be used directly from y code.
Microsoft.SharePoint.Meetings	Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides types and members that car used to customize Meeting Worksp sites.
Mıcrosoft.Sharef'olnt.Securtty	Microsoft SharePoint Security (in Microsoft.SharePoint.Security.dll)	Provides a set of code ace permission and attribute clai designed to protect \a specific set resources and operations, such access to the Windows Sharel'. Services object model, the ability tc unsafe saving on HTTP Gets, enabling point-to-point Webl connections.
Microsoft.SharePoint.SoapServer	Windows ShatePoint Services (in Microsoft.SharePoint.dll)	Contains classes that implement Windows SharePoint Services \ Service and Web services for worl with Web Part pages and Web Part, most cases, the members of ti classes are not designed to be ca

			from the server but remotely from client a:eplications.
Microsoft.SharePoint.	Utilities	Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides utilities for encoding strings and processing user information.
Microsoft.SharePoint.WebControls		Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides server controls that are used on site and list pages in a SharePoint
Microsoft.SharePoint.WebPartPages		Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides classes, interfaces, and members for creating custom Web Parts and Web Part pages
Microsoft.SharePoint.WebPartPages.Communication		Windows SharePoint Services (in Microsoft.SharePoint.dll)	Provides a set of interfaces and their supporting classes and members that can be implemented in the class of a custom Web Part to support creating connections with other Web Parts.

2.2.3 Programming with the Microsoft.SharePoint and Microsoft.SharePoint.Administration Namespaces

2.2.3.1 Major top-level classes

The Microsoft.SharePoint and Microsoft.SharePoint.Administration namespaces provide types and members for working with lists and sites, as well as for administering a server or collection of servers running Microsoft® Windows® SharePoint™ Services. These namespaces have four major top-level classes:

SPGlobalAdmin
SPVirtualServer
SPSite
SPWeb

From these classes you can gain access to all the other classes to work with lists and Web sites, or to manage one or more servers. Starting with one of these classes, you can work through the object model to the class that you need to use. The following table briefly describes these four classes.

Table 2.2

Class	Description
SPGlobalAdmin	Used for central configuration settings. In particular, you can use it to enumerate the list of virtual servers in Internet Information Services (IIS) and access SPVirtualServer objects.
SPVirtualServer	Represents a virtual server and is used for server-wide configuration settings. Use this class to create, delete, and access sites under a specific virtual server.
SPSite	Represents a site collection (a top-level site and its subsites) and is used for managing existing sites and for accessing site and list data.
SPWeb	Represents an individual site and is used for working with the lists, files, and security of the site, including users, site groups, cross-site groups, and permissions.

In order to perform actions on data within a SharePoint site, you must first obtain an **SPWeb** object, wh serves as the starting point for accessing lists, items, documents, users, alerts, etc. For example, to retur collection of lists for a site within an HTTP context, first obtain the Web object by passing the cun **System.Web.HttpContext** object to the **GetContextWeb** method of the **SPControl** class: SPWeb mySi1 SPControl.GetContextWeb(Context) (in Visual Basic .NET, use Dim mySite as SPWeb SPControl.GetContextWeb(Context)). Then use mySite.Lists to get the collection of lists for the Similarly, mySite.Title returns the title of the site, and mySite.Users returns the users on the site.

2.2.3.2 Objects

Most classes in the Micrôsoft.Sbarel'otnt and **Microsoft.SharePoint.Administration** namespaces start **SP.** Generally, classes in the **Windows SharePoint Services** assembly that don't start with this p represent Web form controls.

Windows SharePoint Services typically does not save modifications of object properties to the database you call the **Update** method on the object. The following example shows how to change the title description for the Tasks list:

```
[Visual Basic .NET]
Dim myList As SPList = myWeb.Lists("Tasks")
myList.Title = "New_Title"
myList.Description = "List_Description"
myList.Update()
[C#]
SPList myList = myWeb.Lists["Tasks"];
myList.Title="New_Title";
myList.Description="List_Description";
myList.Update();
```

An exceptional case is when a document is checked out from a document library. Metadata for the do, cannot be modified using the **Update** method of the **SPListItem** object.

2.2.3.3 Collections

Just as lists are at the center of a SharePoint site, so are collections at the center of its object models. ' use each collection to add, delete, enumerate, and update a type of object. Collection classes general the following traits:

Has a name that ends int'Collection."

Implements the **System.Collections.ICollection** interface.
Has a **Count** property of type **Int32.**
Has an **Int32** indexer that can be used to get the nth item in the collection.
Has an indexer that takes an item identifier.
Has **Add** and **Delete** methods.

Calling the **Add** method for a collection usually updıtes the database on the back-end server appropriate data, except when additional information is required in order to update data. In this case, **Add** method returns an object that you can use to gather the information. For example, to add an iten first use the **Add** method of the **SPListItemCollection** class to return an **SPListItem** object, assign

appropriate properties of the object, and then call the **Update** method to effect changes within the content database.

2.2.3.4 Indexers

Indexers provide a useful means for accessing individual items in collections. To return an item, use square brackets ([]) in C# or parentheses (()) in Visual Basic .NET to contain either an index or a string that identifies the item within the collection.

For example, if mySite represents an instance of the **SPWeb** class, then SPList myList = mySite.Lists["Announcements"] returns the Announcements list in C#, while Dim myList As SPList = mySite.Lists("Announcements") returns the same list in Visual Basic .NET. You can then use the **Items** property for the list object to return all the list items in the list {in C#, SPListitemCollection myItems = myList.Items, or in Visual Basic .NET, Dim myItems As SPListItemCollection = myList.Items). To return only a subset of items from the list, you can call one of the .list object's **GetItems** methods and pass an **SPQuery** object to specify the subset: SPListItemCollection myItems = myList.GetItems(myQuery) (in Visual Basic .NET, DimmyItems As SPListItemCollection= myList.GetItems(myQuery)).

You can use an indexer to return a specific field from a list (for example, *myList.Items["Field_Name"]* in C#, or *myList.Items("Field_Name")* in Visual Basic .NET). You can also specify a field name in an indexer and iterate through the collection of items in a list in order to return values from the field. This example displays the Due Date, Status, and Title values for each item in a list:

```
[Visual Basic .NET]
Dim myItem As SPListItem
For Each myItem In  myItems
  Response.Write(SPEncode.HtmlEncode(myItem("Due Date").ToString()) + "<BR>")
  Response.Write(SPEncode.HtmlEncode(myItem("Status").ToString()) + "<BR>")
  Response.Write(SPEncode.HtmlEncode(myItem("Title").ToString()) + "<BR>")
Next myItem
[C#]
foreach(SPListItem myItem in myItems){
  Response.Write(SPEncode.HtmlEncode(myItem["Due Date"].ToString()) + "<BR>");
  Response.Write(SPEncode.HtmlEncode(myItem["Status"].ToString()) + "<BR>");
  Response.Write(SPEncode.HtmlEncode(myItem["Title"].ToString()) + "<BR>"); }
```

In addition to requiring a **using** directive **(Imports** in Visual Basic .NET) for the **Microsoft.SharePoint** namespace, the previous example requires a directive for the **Microsoft.SharePoint.Utilities** namespace.

The next example shows how to return all Title and Status values for the Tasks list on a SharePoint site.

```
[Visual Basic .NET]
Dim myWeb As SPWeb = SPControl.GetContextWeb(Context)
Dim myTasks As SPList = myWeb.Lists("Tasks")
Dim myItems As SPListItemCollection = myTasks.Items
Dim myItem As SPListItem
For Each myItem In  myItems
  Response.Write(SPEncode.HtmlEncode(myItem("Title").ToString()) + " :: " _
    & SPEncode.HtmlEncode(myItem("Status").ToString()) + "<BR>")
```

```
Next myItem
[C#]
SPWeb mySite = SPControl.GetContextWeb(Context);
SPList myTasks = mySite.Lists["Tasks"];
SPListItemCollection myItems=myTasks.Items;
foreach(SPListItem myItem in myItems){
   Response.Write(SPEncode.HtmlEncode(myItem["Title"].ToString()) + " :: " +
      SPEncode.HtmlEncode(myItem["Status"].ToString()) + "<BR>"); }
```

In addition to requiring a **using** directive **(Imports** in Visual Basic .NET) for the **Microsoft.SharePc** namespace, the previous example requires a directive for the **Microsoft.SharePoint.Utiliti.es** namespace.

You can also use indexers to modify values in a list. In the following example, a new.list item is added collection and the values for a URL column are assigned to the item:

```
[Visual Basic .NET]
Dim myNewItem As SPListItem = myList.Items.Add()

myNewItem("URL_Field_Name") = "URL, Field_Description"
myNewItem.Update()
[C#]
SPListItem myNewItem = myList.Items.Add();
myNewItem["URL_Field_Name"] ="URL, Field_Description";
myNewItem.Update();
```

The URL field type is unique because it involves two values (separated by a comma and a space), and above example these values are both assigned to the new item through a single indexer.

The following example sets the Status and Title values for a list item.

```
[Visual Basic .NET]
Dim myItem As SPListItem = myItems(0)
myItem("Status") = "Not Started"
myItem("Title") = "Task Title"
myItem.Update()
[C#]
SPListItem myItem = myItems[0];
myItem["Status"]="Not Started";
myItem["Title"]="Task Title";
myItem.Update();
```

An indexer throws an **ArgumentüutüfRange** exception ifit does not find the specified item.

For infoıination about specific data formats used for list items in Windows SharePoint Services, **SPListItem** class.

2.2.4 Programming with. the Microsoft.HtmlTrans.Interface Namespace

A user may view a document from a document library without having an application capable of displaying the requested document installed on the client computer. Microsoft Windows SharePoint Services provides the **Microsoft.HtmlTrans.Interface** namespace as an infrastructure on which to build .custom converter applications that deliver an HTML version of the requested document to the user in these circumstances in place of the unrecognized binary file format.

The HTML Viewer feature must be enabled in SharePoint Central Administration in order to use this functionality.

The **Microsoft.HtmlTrans.Interface** namespace consis.ts of two interfaces, **IhtmlTrLoadBalancer** and **Ihtml**l'I'rf.auncher. An implementation of these interfaces also relies on an XML configuration file, called htmltransinfo.xml, which resides on the server running Windows SharePoint Services.

The **IhtmlTrLoadBalancer** interface helps to select the server on which to run the custom converter application.

The **IhtmlTrLauncher** interface launches the custom converter application and returns the HTML results to Windows SharePoint Services.

The **htmltrans**info.xml file redirects the document request to a handler page, which calls the load balancer and **launcher** components. An entry in the htmltransinfo.xml configuration file usesthe following format:

```
<HtmlTrInfo>
  <Mapping Extension="ext" AcceptHeader="application/vnd.my-app" HandlerUrl="myapphandler.aspx" ProgId=""/>
</HtmlTrInfo>
```

2.2.4.1 Using the Microsoft.HtmlTransJıiterfa.ceNa.mespa.ce

The following sequence of events occurs when a user rec:fıestsadocument from a server running Windows SharePoint Services:

Win<lows SharePoint Services identifies the file extension of the document in the htmltransinfo.xml file.ılf found, Windows SharePoint Services retrieves the matching;**AcceptHeader** attribute; if not found, Windows SharePointServicesprompts the user to download the document.

Win<lows SharePoint Services checks the **AcceptHeader** attribute value against the **Accept-Header HTTP** header of the request to see whether the client computer recognizes the requested document type. If so, Windows SharePoint Services delivers the document in its native format.

Windows SharePoint Services also retrieves the **Progld** attribute from the Htmltransinfo.xml file and attempts to open the file on the client computer by using the component designated by the Progld. If this attempt fails, Windows SharePoint Services continues with the HTML conversion process.

If the client computer does not recognize the requested document type, Windows SharePoint Services prompts the user: "Do you want to convert the document for viewing in the browser?" If the user chooses not to convert the document, Windows SharePoint Services prompts the user to download the document instead.

If the user chooses to convert the document, Windows SharePoint Services forwards th~teqtıestto the handler page specified by the Handleı-Url attribute in the Htmltransinfo.xml file. The handler page manages the conversion process and delivers the converted file to the user for viewing in the browser.

2.2.4.2 Implementing Custom Document Conversion

To implement custom document Conversion on the Windows SharePoint Services platform you must create the following:

An entry in the Htrnltransinfo.xml file for the document type.

A handler page that launches the document conversion process and returns the HTML output to the user.

A set of custom conversion components.
You must choose among three methods to build the set of custom conversion components:

Build custom implementations of the **IhtmlTrLoadBalancer** and **IhtmlTrLauncher** interfaces to launch a custom converter application and return its results to Windows SharePoint Services.

Build a set of custom conversion components that do not use the **Microsoft.HtmlTrans** interfaces at all.

Call the custom implementations of the **IhtmlTrLoadBalancer** and Ihtml'I'rLauncher interfaces provided as part of the Microsoft Office 2003 Editions Resource Kit and provide a custom converter application for the **IhtmlTrLauncher** implementation to call.

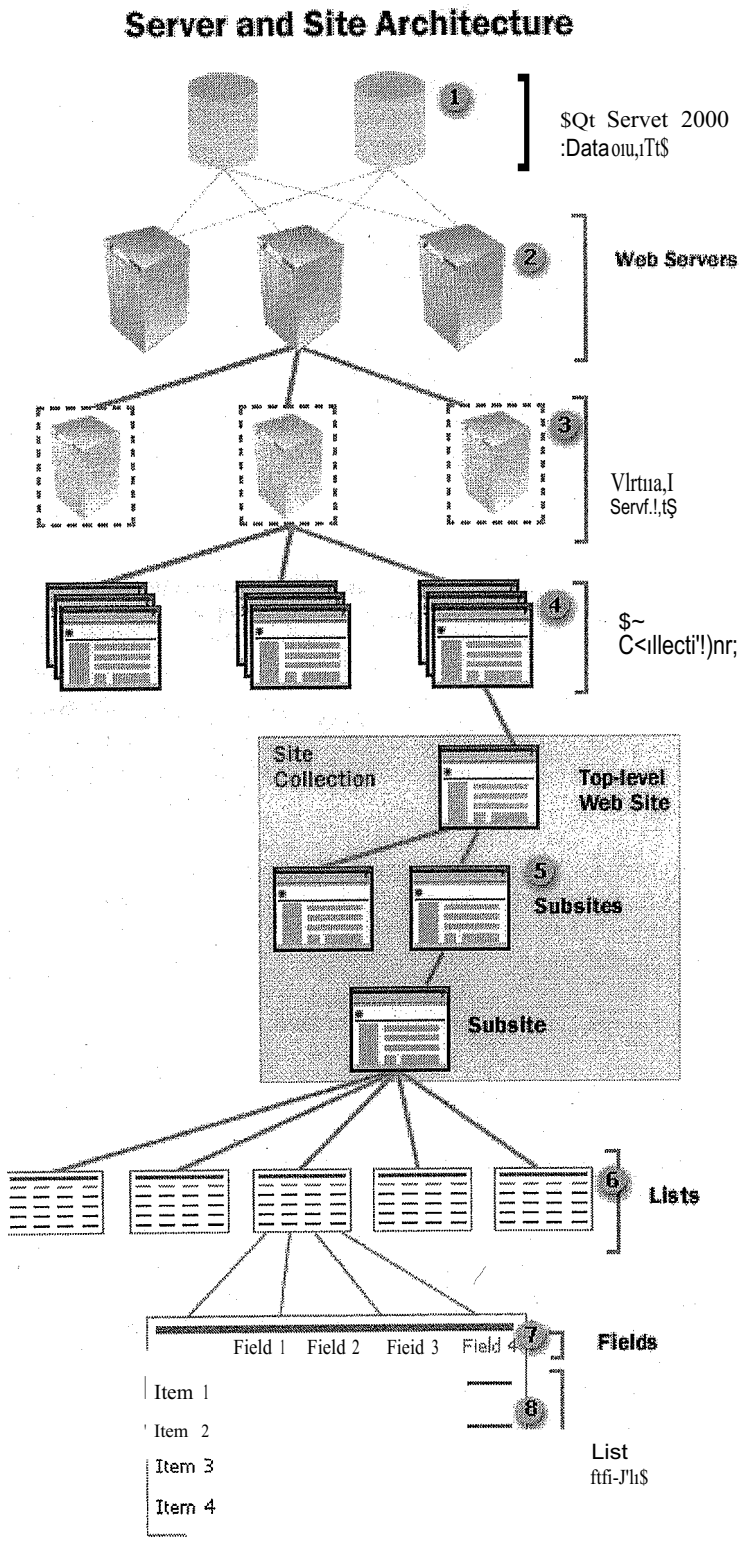
2.2.4.3 Server and Site Architecture Overview

Microsoft® Windows® .sharePoit1tJM$ervices offers a highly structured server-side object model .that make: it easy to access objects representing the various aspects of a SharePoint site. From higher level objects, yoı can drill down through the object hierarchy to obtain the object that contains the members you need to use it your code.

Depending on which part of a Windows SharePoint Services deployment that you need to customize, and th type of custom application you are creating, you can use different means of entry into the object model t, obtain the appropriate higher-level object from which to start. The **SPGlobalAdmin** class provides access t, the highest levels in the hierarchy, which are related to administrative operations, and you can also use thi class to reach the lower objects in the hierarchy. Use the constructor of this class to attain an **SPGlobalAdmi** object to customize global administrative settings in the deployment.

If you are creating a Web Part, Web serv.i?e,or Web application to work with site collections, individual site: or lists, use the **GetContextSite or GetContextWeb** method of the **SPControl** class to attain the current siı collection or site. When you create a Web application in the _layouts directory, its functionality become available to all sites on the Web server. Outside an HTTP context, such as in a console application, use tl: constructor of the **SPSite** class to attain a specified site collection.

The following diagram shows the Windows SharePoint Services site architecture in relation to the collectior and objects of the **Microsoft.SharePoint** and **Microsoft.SharePoint.Administration** namespaces.


Server and Site Architecture
1
$Qt Servet 2000
:Data
2
Web Servers
3
Vlrtıa,I
Servf.!,tŞ
4
$~
C<ıllecti'!)nr;
Site
Collection
Top-level
Web Site
5
Subsites
Subsite
6
Lists
7
Fields
Field 1
Field 2
Fieid 3
Field 4
8
Item 1
Item 2
Item 3
Item 4
List

Figüre 2.2

The **ContentDatabases** property of the **SPVirtualServer** class returns an **SPContentDatabaseCollectio** object that represents the collection of content databases used for a virtual server. Each **SPContentDatabas** object provides access to properties of the content database.

The **WebServers** property of the **SPGlobalConfig** class returns an **SPWebServerCollection** obje representing the collection of front-end Web servers in the Windows SharePoint Services deployment. Ear **SPWebServer** object provides access to properties of the Web server.

The **VirtualServers** property of the **SPGlobalAdmin** class provides access to an **SPVirtualServerCollecti** object representing all of the virtual servers in the Windows Sharef'oint Services deployment. T **OpenVirtualServer** method of the **SPGiobalAdmin** class returns a specific virtual server. Es **SPVirtualServer** object has members that can be used to manage the virtual server. The **Sites** prope provides access to the **SPSiteCollection** object representing the collection of all site collections on the virt server, and the **Add** method is used to create top-level site collections.

Each **SPSite** object represents a site collection and has members that can be used to manage the collection. The **AIIWebs** property providesaceess to the **SPWebCollectio.R_g~ject**that represents collection of all sites within the site collection, including the top-level site. The **OpenWeb** method of **SPSite** class returns a specified site.

Each site collection includes any number of **SPWeb** objects, and each object has members that can be usı manage a site, including its template and theme, .but also to access files and folders on the site. The V property returns an **SPWebCoUection** object representing all the subsites of a specified site, and the property returns an **SPListCollection** object representing all the lists in the site.

Each **SPList** object has members for managing·the list or for accessing items in the list. The **Get]** method can be used to perform queries that return specific items. The **Fields** property retun **SPF'ieldCollection** object representing all the fields, or columns, in the list, and the **Items** property retu **SPListItemCollection** object representing all the items, or rows, in the list.

Each **SPField** object has members that contain settings for the field.

Each **SPListItem** object represents a single row in the list.

2.2.5 Introduction to Templates and Definitions

Microsoft Windows SharePoint Services is designed as a platform to support different types o SharePoint Team Services 1.0 from Microsoft included only one definition for instantiating sites, the Site," but Windows SharePoint Services extends the template architecture so that multiple site definine templates can be used in a deployment. Site definitions consist of multiple XML files located in system of a front-end Web server. Custom templates are created through the user interface or throı object model and are stored in the database, providing a means for reusing customized lists and sites.

2.2.5.1 Custom Templates

A custom template is a customization applied to a site definition. When a user customizes a site or li: user interface, the custom template consists of the difference between the original state of the site c determinedby its definition and the state of the site when the custom template is generated. Custom te remain tied to a particular site definition (for example, the one for a SharePoint site or a Meeting We site), so that if the site definition is not present or is changed, the custom template will not work.

A custom template is persisted as a file with an .stp extension, which is actually a .cab file that can be renamed with the .cab file extension and opened in Windows Explorer. This file includes one Mariifest.xml file in Collaborative Application Markup Language (CAML) that the server generates as a subset of the Microsoft SharePoint Migration Tool (Smigrate.exe) manifest file format.

Custom templates include both list templates and site templates.

List templates contain the files, views, fields, Web Parts, and, optionally, the content that is associated with a list. Users create list templates on the Save as Template page for a list or through code that uses the SaveAsTemplate method of the SPList class. When saved, list templates are stored in the List Template Gallery of the top-level site in a site collection, where they become available to all sites in the site collection that derive from the same site definition and language as the site on which the list was originally created. To make a list template available to a site in another site collection, download the template from its current gallery and then upload it to the gallery of-the new site collection.

Site templates contain the same type of data as list templates, but site templates include data for the entire site. Like list templates, site templates may also include the content Of the site.

The maximum size for the content is 10 megabytes (MB).

Users create site templates on the Save Site as Template page or through code that uses the SaveAsTemplate method of the SPWeb class. When 'saved;' site templates are stored in the Site Template Gallery of the top-level site in a site collection, where they become available for subsite creation on all sites in the site collection. Similarly to list templates, site templates can be downloaded and moved to other site collection galleries. Unlike list templates, however, site templatı!~~ah also be moved into the Central Site Template Gallery, where they become available for top-level sitrcrea~o~ .ill Self-Service Site Creation in addition to subsite creation. To add a site template to the gallery from the command line, download the site template from the gallery, run stsadm.exe -o addtemplate -filename *Template_File_Name* -title *Template_Title* *[-description Template_Description],* and reset Internet Införmation Services (IIS). In code, use the AddCustomGlobalWebTemplate method of the SPGlobalAdmin class to add a site template to the gallery.

Site templates can include list templates. When a top-level si-ı:ejs saved as a template with content, it includes any list templates that are in the List Template Gallery for the site collection.

2.2.5.2 Site Definitions

A s.te definition defines a unique type of SharePoint site. Natively installed definitions .in Windows SharePoint Services include the STS type, which defines the Team Site, Blank Site, a,nd Document Workspace configurations, and the MPS type, which defines the Basic Meeting Workspace, Blank Meeting Workspace, Decision Meeting Workspace, Social Meeting Workspace, and Mu)tipage Meeting Workspace configurations. Each site definition emerges through a combination of multiple files that are placed in the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ subdirectorieson one or more front-end Web servers during installation of Windows SharePoint Services. Site definition files include core schema XML files, .aspx pages, document template files (.dot, .htm, and so on), and content files (.gif, .doc, and so on).

2.2.5.3 Ghosting

Site <:lefinition files are cached in memory on the server at process startup of Micrnsoft Internet Information Services (IIS), which improves scalability and performance by reducing unnecessary data storage orretrieval, and by allowing uncustomized pages to be reused across sites. The information contained in these files is pulled from the cache at run time. Pages and list schemas are read from the site definition files but appear to be actual files within a site, which is why these files are referred to as "ghosted." Ghosted pages are therefore

pages whose actual content does not resi.de in the database but on disk, although a row for each page can found in the database containing a column value that points to the page source in the file system. New V, Part Pages are also ghosted.

When site pages are customized, excluding browser-based customizationssuch as modifications to Web Pr the pages become "unghosted" and their contents are stored in the database. Windows SharePoint Serv does not natively provide a means for "re-ghosting" pages. In addition, uploaded .aspx files are consid unghosted automatically. The contents of unghosted pages are routed through the Safe Mode parser, w· prevents server-side code from executing, and which depends entirely on the Safe Controls list specifie the web.config file of the wwwroot directory to determine which controls can be rendered at run time.

2.2.5.4 Core Schema Files

Table 2.3 Table describes prominent XML files that can be modified for a site definition and shows th locations in the file system.

jDOCICON.XML	IIWEBTEMP.XML	l!ONET.XML	j[SCHEMA.XML
Maps file ProgIDs and file extensions of document types to specific icons and to controls for opening each type.	Specifies configurations for site definitions.	Defines the navigation areas, specifies the list definitions available on the Create Page, specifies document templates and their files, defines the base types for lists, and defines configurations and modules for site definitions.	Defines the views, forms and special fields in definition. Each definiti own SCHEMA.XML fik

\TEMPLATE\XMLII\TEMPLATE\1033\XML!i\TEMPLATE\1033\STS\XMI.,II\TEMPLATE\1033\STS'
st:._Dejinition_Name

Each of these XML files uses Collaborative Application Markup Language (CAML) as the conve defining various aspects of a site, but three of them stand out in relation to customizing site defi: WEBTEMP.XML, ONET.XML, and SCHEMA.XML.

WEBTEMP.XML specifies which configurations are available for creating sites, but this file .can modified to hide an existing configuration from the Template Selection page. If you are creatin] definition, instead of editing the original WEBTEMP.XML file, it is recommended that you create a WEBTEMP*.XML file as described in Creating a .site Definition from an Existing Site Definition. V, SharePoint Services merges the contents of all files that match WEBTEMP*.XML when showing a configurations on the Template Selection page. This makes it easier to install·and uninstall site del because their contents do not have to be merged into one WEBTEMP.XML file.

ONET.XML defines the top navigation and Quick Launch areas, specifies which list types are ava the Create Page, specifies document'templates and their file types, and definesthe base types for lis include Generic List, Document Library, Discussion Forum, Vote or Survey, and Issues List. In ONET.XML defines the configurations and modules in the site definition. A configuration specifies lists and modules that are included when a site is created, and a module specifies the file or files t during site creation and the location where they are stored. For more information about configura Using Configurations and Configuration. For more information about modules, see Using Moduk Files to a Site. Definition and Module. In ONET.XML, you can reference a list that is defined in a site definition by adding a ListTemplate element to the ListTemplates section and setting its Pati to the name of the directory for the site definition (for example, STS).

SCHEMA.XML defines the views, forms, toolbar, and any special fields for a list type. For examples of creating or customizing list definitions, see Adding a Field to a List Definition and Creating a List Definition for a Custom List.

For more information about the schema files used in a Windows SharePoint Services deployment, see Major Schema Files.

For the following reasons, it is recommended that you create a new site definition rather than modify an originally installed site definition:
Use of repairs and service packs could revert your modifications.
Custom templates based on the original site definitions may not work.
Site definitions with references to original list definitions may not work.
It is safe, however, to mark configurations as hidden in the original WEBTEMP.XML file by setting the Hidden attribute of the **Configuration** element to **TRUE.**

2.2.5.5 Pros and Cons

Customization of site definitions holds the following advantages over custom templates:
Data is stored.directly on the Web servers, so performance is typically better.
A higher level of list customization is possible through direct editing of a SCHEMA.XML file.
Certain kinds of customization to sites or lists require use of site definitions, such as introducing new file types, defining view styles, or modifying the drop-down **Edit** menu.
Site definition disadvantages include the following:
Customization of site definition requires more effort than creating custom templates.
It is difficult to edit a site definition after it has been deployed.
Doing anything other than adding code can break existing sites.
Users cannot apply a SharePoint theme through a site definition.
Users cannot create two lists of the same type with different default content.
Customizing site definitions requires access to the file system of the front-end Web server.
Custom templates hold the following advantages over customization of site definitions:
Custom templates are easy to create.
Almost anything that can be done in the user interface can be preserved in the template.
Custom templates can be modified without affecting existing sites that have been created from the templates.
Custom templates are easy to deploy.
Custom template disadvantages include the following:
Custom templates are not created in a development environment.
They are less efficient in large-scale environments.
If the site definition on which the custom template is based does not exist on the front-end server or servers, the custom template will not work.

2.2.6 Guidelines for Templates and Definitions

2.2.6.1 Site Definitions

To customize site definitions on front-end Web servers, it is recommended that you createa copy of one of the existing site definition directories and create a site definition as described in Creating a Site Definition from an Existing Site Definition. Changes that you make within originally installed files may be overwritten when you install updates or services packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

Changing a site definition after it has already been deployed can break existing sites and is not supported. If you find that you must modify a site definition after it has already been deployed, keep in mind that adding features can cause fewer problems than changing or deleting them. Changing features often results in loss of data and deleting them often results in broken views.

For backup/restore and migration operations, you must restore your site to a server th~fra.s the same front customizations as the server that you backed up or migrated from. Customizing a site definition often reqı that you reset, or stop and restart, Internet Information Services (US) before changes take effect.

Always test custom site or list definitions before deploying them.

Modifying site definition files to customize already existing sites or lists is not supported. A rule of thu: to use site definitions to modify sites that will be created, but use the object model to modify sites afte are created.

Files not located under a site definition folder (for example, STS or MPS) are global to a Windows Sharı Services deployment. Changing files within the *Local_Drive:\Program* Files\Common Files\Mic Shared\Web Server Extensions\60\TEMPLATE\LCID directory will affect the entire language-sı installation of Windows SharePoint Services, and changing files within the *Local_Drive:Wr* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\XML folder will affı entire deployment of Windows SharePoint Services on the front-end Web server.

When modifying site definitions, keep in mind that the contents of pages customized in Microsoft FrontPage 2003 are stored in the database. If you modify, ör unghost, a page in Frorıtf'ağe, changes tl later make to the definition may not affect the page. For more information on· ghosting, see Introdu Templates and Definitions.

2.2.6.2 File Name Restrictions

For security reasons, Microsoft Windows SharePoint Services only reads files in the *Local Drive:* Files\Common Files\Microsoft Shared\Web Server Extensiöns\60\Template directoryJree i:ftheir nr composed of ASCII letters, numbers, periods, underscores, or dashes. In addition, file names canno two or more consecutive periods. For example, the following are legal file names:

Allltems.aspx

Dept] 234.doc

Long.Name. With.Dots.txt

The following are not legal file names:

HailCAlsar.wav

File Name With Spaces.avi

Wow ... ThislsBad.rif

ceıi.htm

2.2.6.3 Advanced Site Definition Features

CAML provides various elements.and attributes that can be used to make advanced customizat definitions, which are described in the following tables.

Table 2.4 Elements

!Element	l[~~s_criJJdon
!ExecuteUri	.. l!Sp~e~jfya page to open immediately after site~creation.
JExternalSecurityProviderJ	[Implement a custom security :ı:ırovider.
JFileDialogPostProcessor	l!Customizethe Open or Save dialog box (File menu) used in docume

Table 2.5 Attributes of the **Project** Element

Alternate	Description
AlternateCSS	Add or modify CSS definitions.
AlternateHeader	Replace the top navigation area in site pages.
CustomJSUrl	Provide custom ECMAScript (JavaScript or Microsoft Jscript code) functions (for example, to customize the drop-down Edit menu.)
DisableWebDesignFeatures	Disable features on a site (for example, Backup, Subsite Creation, and so forth.)

2.2.7 Customizing CSS Style Definitions

You can customize CSS style definitions by creating a .css file with a unique name in *Local_Drive:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS\LCJD\STYLES*. Copy into this new file the contents of the default .css file specified within .aspx pages of the site definition, and add or update styles in the new .css file. Use the **AlternateCSS** attribute to specify the path to the new .css file, such as follows:

```
AlternateCSS =
"/_layouts/[%=System. Threading. Thread. CurrentThread. CurrentUICulture.LCID%} /styles/MyStyles. css ">
```

2.2.8 Custom Templates

When deploying a custom template, remember that the .stp file contains a unique identifier in relation to the original site definition from which the custom template is derived. This means that a custom template will not function unless the original site definition is present on the front-end Web server or servers.

CHAPTER3

3.1 Windows SharePoint Services WebService

The **Windows SharePoint Services Web Service** provided by the **Microsoft.SharePoint.SoapServ** namespace has numerous methods for accessing content onasite, including methods.for working with lists site data, as well as methods for customizing meetings, imaging, document workspaces, or search.

The Simple Object Access Protocol (SOAP) interfaces used in these services provide .NET developers w object models for creating solutions that work with Microsoft Windows SharePoint Services remotely fror client or custom application. The interfaces are integrated with the server-side object models of the **Windo SharePoint Services** assembly, and their design has been optimized to reduce the number of roundtı transacted between client computer and server.

Web services provide their functionality through the _vti_bin virtual directory, which maps to *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI phys directory in the file system.

3.2 Guidelines

3.2.1 Adding a reference

To create a reference within a Micrpsoft Visual Studio .NET project to.~ Web service in Windows Share} Services, right-click **References** in Solution Explorer and click **Add Web Reference** in the shortcut men the address bar of the Add Web Reference browser, type the address for the service as follows:

ne.asmx

If you are using methods in the Administration service, instead specify a path that goes to the administı port and that uses_vtiadm as the virtual directory, as follows:

al_Server:Port_Num

Click **Go,** and after the description page opens, click **Add Reference.**

3.3 Instantiating a service

Instantiate a Web service within code by using a constructor as follows:

e_Name();

Web_Reference_ Folder _Name represents the folder name used for the reference to the Web service in Studio .NET, and *Service Name* represents the name of the class providing methods that can be through the Web service.

3.4 Establishing user credentials

To establish system credentials for the current security context in which the application is running, following line after the previous line, which authenticates the user running the code by passing theii credentials to the Web Service from the system credential cache:

```
[Visual Basic .NET]

ServiceInstanceName.Credentials = System.Net.CredentialCache.DefaultCredentials

[C#]

ServiceInstanceName.Credentials = System.Net.CredentialCache.DefaultCredentials;
```

3.5 Programming Tasks

The following programming tasks demonstrate how to use a Web service in Windows SharePoint Services.

Adding Users to a Cross-Site Group
Returning Items from a List
Returning Lists
Updating List Items
Creating a Web Service for Remote Check-In and Check-Out

3.6 Guidelines for Using the Object Model

This section of the SDK provides important guidelines to follow when writing code that uses the object model in Microsoft Windows SharePoint Services.

Getting Started with Customizing a SharePoint Web Site in Visual Studio .NET
Determining Where to Build a Custom Application
Establishing Site Context
Registering and Importing Namespaces
Forms of URL Strings
Converting Date and Time Values
Security Validation and Making Posts to Update Data
Optimizing Code Performance
Setting the Culture and Language
Displaying ASP.NET Error Messages

3.7 Getting Started with Customizing a SharePoint Web Site in Visual Studio .NET

The Microsoft® Visual Studio® .NET integrated development environment (IDE) offers the premier environment for customizing a site based on Microsoft Windows® SharePoint™ Services. You need to take certain steps in order to use this environment when customizing a SharePoint Web site. These include using a specific path when creating a Web application, setting a reference to Microsoft.SharePoint.dll, and may include adding a Multipurpose Internet Mail Extensions (MIME) type in Internet Information Services (US).

3.8 Adding a new MIME type

Since II.Sv6 disables requests to .tmp files, if you are using Visual Studio .NET Version *7.0.nnnn* you need to define a MIME type in IIS before you can create a custom Web application on a SharePoint Web site through the Visual Studio .NET IDE. To define the MIME type, perform the following steps:

On the Windows Start menu, point to Administrative Tools and click Internet Information Services (US

In the left pane of the Microsoft Management Console,.yf'mıı1<.ithe.branches of the console tree.
Right-click Web Sites and click Properties on the shortcuftnenuJhat appears.
In the Web Sites Properties dialog box, select the HTTP Headers tab, and click MIME Types.
In the MIME Types dialog box, click New.
In the MIME Type dialog box, type .tmp in the Exteıısioıı tyxtbox, and type common/type in the Mil type text box.
Click OKin each dialog box until you exit the Web Sites Properties box.

3.9 Creating a Web **application**

Creating a Web application in Visual Studio .NET from a remote computer involves an extra step in procedure required for creating a Web application in Visual Studio .NET when it is installed on the server is running Windows SharePoint Services. To create a custom Web application within the context SharePoint deployment, perform the following steps:

On the File menu, point to New and then click Project.

In the New Project dialog box, select Visual Basic Projects or Visual C# Projects in the Project 1 box, depending on which language you prefer.

In the Templates box, select ASP.NET Web Application.

In the Location box type the following path:

http://Server _Name/ _layouts/Web *_Application _Name*

This path includes the name of your server, the layouts directory (_layouts), and the name of your c Web application. Creating the custom application in the _layouts directory makes the project accessibk sites on the server.

To create a Web application on the administrative port of the deployment, type th~ following path:

http://Server *_Nanie:Administrative _Port _#/Web _Application _Name*

Click OK.

At this point, if you are working from a remote computer, a Web Access Failed dialog box appears, you that the file path does not correspond to the URL you previously specified in the New Project dial Perform the following steps:

Under What would you like to do?, ensure that Retry using a different file share path is selected 2 the following path in the Location box:

»Server _Name\Local _Drive$\Program Files\Common Files\Microsoft Shared\Web Extensions\60\TEMPLA TE\LA YOUTS\Web *_Application _Name*

Type the following path to create an application on the administrative *\\Server _Name:Administrative _Port _:/fi.Local_Drive$\Program* Files'Cornmon Files'Microsoft Sha Server Extensions\60\TEMPLATE\ADMIN\l 033\Web *_ApplicGtion_Name*

Click **OK**.

3.9.1 Setting a reference to the Windows SharePoint Services assembly

After creating a new project, add a reference to **the/Windows SharePoint Services** assembly in order to implement 'Intellisense features in the Object Browser and Code Editor. The process for setting a reference varies, depending on whether you are using Visual Studio .NET on a remote computer or on the server running Windows SharePoint Services. To add a reference to the assembly, perform the following steps:

In Solution **Explorer,** right-click the References node and click **Add Reference** on the shortcut menu.

If you are using Visual Studio .NET on the server, perform the following steps:

On the **.NET** tab of the **Add Reference** dialog box, select **Windows SharePoint Services** in the list of components, click **Select,** and then click **OK.**

If you are using Visual Studio .NET on a remote computer, perform the following steps:

Click **Browse,** and in the **Select Component** dialog box, navigate to the *Local_Drive\Program* Files\Common Files\Microsoft Shared\Web.Server Extensions\60\ISAPI folder on the server running Windows SharePoint Services. You may instead prefer to first copy Microsoft.Sharel'oint.dll from this folder to a local drive on the remote computer and then open this local copy of the Dl.Ljnthe **Select Component** dialog box.

Select Microsoft.SharePoint.dlland then click **Open.**

In the **Add Reference** dialog box, click **OK.**

3.9.2 Intellisense

Within the integrated development environment of Visı:ıalStudiö .NET, Intellisense features are provided for namespaces in the **Windows SharePoint Services** assembly once a reference has been set to this assembly. The file that provides the information used by Jnte1lisense for Windows SharePoint Services is Microsoft.Sharel'oint.xml, which is installed in the *Local _Dr-ive:\\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI folder on each front-end Web server in the deployment. For updates to this file, check the Microsoft SharePoint Products and Technologies Developer Center.

3.9.3 Determining Where to Build a Custom Application

You can build and store custom ASPX pages and Web applications in the following directory, which supports the _layouts virtual directory:

Local_Drive:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS

Pages in this directory are accessible from all Web sites in the Windows SharePoint Services.deployment through a URL in the following form:

*http://Server _Name/[sites!][Site _Name]/[SubSite _Name]/[...]/*_layouts/*File _Name.aspx*

For code that involves use of only the **Micros.oft.SharePoint.Adnıinistration**namespace for working with global settings in a Windows SharePoint Services deployment, it is recommended that the Web application be created on the administrative port. Build and store ASPX pages and Web applications for the administrative port in the following directory:

Local _Drive:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\ADMIN\1033

Pages in this directory are accessible through a URL in the following form:

http://locallıost:Port_*#/File _Name.aspx*

For information on how to create a Web application in Microsoft Visual Studio .NET that runs in the context of Windows SharePoint Services, see Getting Started with Customizing a SharePoint Web Site in Visual Studio .NET.

You cannot use custom code inline within the native pages of Microsoft Windows SharePoint Services that are used for working with lists or sites.

Console applications can be created anywhere on the front-end Web server. For information on creating a console application that runs in the context of Windows SharePoint Services, see Creating a Console Application.

3.9.4 Registering and Importing Namespaces

Use processing instructions to register namespaces, assemblies, and other properties for the installation of Windows SharePoint Services.

For ASP.NET pages, use instructions such as the following at the head, which register the **Microsoft.SharePoint.WebControls, Microsoft.ShatePoint.Utilities, Microsoft.SharePoint,** and **Microsoft.SharePointWebPartPages** namespaces:

```
<%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
 Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
<%@ Register Tagprefix="Utilities" Namespace="Microsoft.SharePoint.Utilities"
 Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
<%@ Import Namespace="Microsoft.SharePoint" %>
<%@ Register Tagprefix="WebPartPages" Namespace="Microsoft.SharePoint.WebPartPages"
 Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
```

You can obtain the **PublicKeyToken** value for the current Windows SharePoint Services deployment from the default.aspx file in the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LCID(l033 in English)\STS folder, or from information provided for the **Microsoft.SharePoint**assembly at *Local_Drive:\WINDOWSjWINNT\assembly* in Windows Explorer.

In code-behind files or console applications, import namespaces in .C# through the **using** directive, as follows:

```
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using Microsoft.SharePoint.WebPartPages;
```

In Visual Basic .NET, use the Imports statement to import namespaces, as follows:

```
Imports Microsoft.SharePoint
Imports Microsoft.SharePoint.WebControls
Imports Microsoft.SharePoint.WebPartPages
```

3.9.5 Establishing Site Context

To work with a deployment of Microsoft Windows SharePoint Services programmatically in an HTTP context, your code must first establish the site or site collection context for requests that are made to the server.

3.10 ASPX pages and Web applications

Create custom ASPX pages and Web applications in the following directory:

Local_Drive\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS

Pages or applications located in this directory are accessible from all sites on the Web server by browsing to a URL that is in one of the following forms:

http://*Server_Name*/[sites]/[*Site_Name*]/[*SubSite _Name*]/[ ]/.:layouts/*Page_ Name.aspx*

http://*Server_ Name*/[sites]/[*Site_Name*]/[*SubSite _Name*]/[ ]!_layouts/*Application _Name!W* ebForml .aspx

Establish the context of a request that is made to the server by using the **.GetContextSite or GetContextWeb** method of the **SPControl** class to return an **SPSite or SPWeb** object that represents the context. If you create an ASPX page or Web application within the LAYOUTS directory.imd incfode the following snippet within the code, the page or application becomes available for use in any site or subsite on the Web server by browsing to the appropriate URL:

```
[Visual Basic .NET]

Dim siteCollection As SPSite = SPControl.GetContextSite(Context)
Dim site As SPWeb = SPControl.GetContextWeb(Context)
[C#]

SPSite siteCollection = SPControl.GetContextSite(Context);
SPWeb site = SPControl.GetContextWeb(Context);
```

Both methods are used in the snippet for the sake of illustration, but they can also be used individually.

The functionality of a page or application located within the LAYOUTS directory is thus available to any si on the server that links to it. The following code, for example, displays the number of users on a site:

```
[Visual Basic .NET]

Dim site As SPWeb = SPControl.GetContextWeb(Context)
Response.Write(site.Users.Count.ToString())
[C#]

SPWeb site = SPControl.GetContextWeb(Context);
Response.Write(site.Users.Count.ToString());
```

When the page or application that contains this code is browsed to through a URL that combines the pat a specific site with the path for the page or application within the LAYOUTS directory, the number of on the current site is displayed. For example, each of the following URLs could be used for an ASPX pag

http://Server _*Name!*_layoutsll 033/MyApplication.aspx,

*http://Server _Name/sitesi Site*_CollectionOayouts/1033/MyApplication.aspx

http://Server_Name/Subsite/_layouts/1033/MyApplication.aspx

Variations of the **SPControl** methods can also be used depending on the need. For example, the fol line returns the context of a specified site by using an indexer with the **AllWebs** property:

```
[Visual Basic .NET]

Dim site As SPWeb = SPControl.GetContextSite(Context).AllWebs("Site_Name")
[C#]

SPWeb site = SPControl.GetContextSite(Context).AllWebs["Site_Name"];
```

The next example returns the top-level Web site of the current site collection by using the **RootWeb** l:

```
[Visual Basic .NET]

Dim topSite As SPWeb = SPControl.GetContextSite(Context).RootWeb
[C#]

SPWeb topSite = SPControl.GetContextSite(Context).RootWeb;
```

The **GetContextSite** and **GetContextWeb** methods are also used in custom Web services and We return the HTTP context.

3.10.1 Console applications

If you are writing code within a console application or custom executable application and you want to work with a specific site collection, you must instead use the **SPSite** constructor to instantiate an object that represents the site collection, as follows:

```
[Visual Basic .NET]

Dim siteCollection As New SPSite("http://Server_Name/[sites]/Site_Name")
Dim site As SPWeb

site = siteCollection.AllWebs("Site_Name")
[C#]

SPSite siteCollection = new SPSite("http://Server_Name/[sites]/Site_Name");
SPWeb site = siteCollection.AllWebs["Site_Name"];
```

3.11 Forms of URL Strings

Microsoft Windows SharePoint Services uses different forms of URL depending on the context and member being used. Certain members require an absolute URL, such as the **SPSite** constructor, or methods in the **Microsoft.SharePoint.Administration** namespace that require a uniform resource identifier (URI) for a parameter. Most members in the object model instead require a relative URL that is based on the address of the server, such as the **Add** methods of the **SPSiteCollecti,on** . class. However, some members in the object model require or accept a relative URL that is based on the current site, such as the **Add** methods of the **SPWebCollection** class or the **Open Web** method of the **SPSite** class.

Windows SharePoint Services parses URL strings to determine the form of URL based on a specified protocol (for example, **http:)** or on the placement of a forward slash(/) within the string. You can use the following forms ofURLs:

Absolute URL that specifies a full path and that begins by specifying a protocol. For example, http*://Server_Name/[sites!]Site _Name! Folder _Name/ File_Name.*

Server-relative URL based on the server address that begins with a forward slash, specifying a complete path from site name to file name. For example,*/[sites/]Site_Name!Folder _Name!File_Name.*

Site-relative URL based on the site address that does *not* begin with a forward slash and that specifies a compJete path from folder name to file name. For example,*Folder _Name/ File_ Name.*

Relative URL based on the folder containing a file that does not contain *any* forward slashes and that specifies the name of the file. For example, *File Name.*

3.12 Security Validation and Making Posts to Update Data

For reasons of security, Microsoft Windows SharePoint Services by default does not allow you to make r from a Web application to modify the contents of the database unless you include security validation or page making the request. Two kinds of security validation can be used, depending on whether the code o page applies global1yto a virtual server or Windows SharePoint Services deployment, or to a single si site collection within the deployment.

3.13 Updating data for a site or site collection

Add a page directive and a FormDigest control to the page making the request. The following din registers the Microsoft.SharePoint.WebControls namespace:

```
<%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
  Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral,
  PublicKeyToken=71e9bce111e9429c" %>
```

You can obtain the PublicKeyToken value for the current Windows SharePoint Services deploymen the default.aspx file in the *Local_Drive:\Program* Files\Common Files-Microsoft Shared\Web Extensions\60\TEMPLATE\LCID(l033 in English)\STS folder, or from information provided f Microsoft.SharePointassembly at Local_Drive:\WINDOWSjWINNT\assemblyin Windows Explorer.

Include a FormDigest control within the form as follows:

```
<form id="Form1" method="post" runat="server">
  <SharePoint:FormDigest runat="server"/>
  <asp:Button id="Button1" style="Z-INDEX: 101; LEFT: 282px; POSITION: absolute;
    TOP: 282px" runat="server" Text="Button"></asp:Button>
</form>
```

Inserting this control on an ASPX page generates a security validation, or message digest, to help pre type of attack wherein a user is tricked into posting data to the server without knowing it. The validation is specific to a user, site, and time period and expires after a configurable amount of tim the user requests a page, the server returns the page with security validation inserted. When the u submits the form, the server verifies that the security validation has not changed. For more informati this control, see the FormDigest class.

3.14 Updating global data

Web applications that use methods of the Microsoft.SharePoint.Administration namespace, sui creating or deleting sites and for global administrative customizations involving multiple virtua require a different security validation. Add the following code to the .vb or .cs file. in an application:

```
[Visual Basic .NET]

Dim globalAdmin As New SPGlobalAdmin()
Context.Items(SPGlobalAdmin.RequestFromAdminPort) = True
```

```
Page.RegisterHiddenField("__REQUESTDIGEST", globalAdmin.AdminFormDigest)
```

[C#]

```
SPGlobalAdmin globalAdmin = new SPGlobalAdmin();
Context.Items[SPGlobalAdmin.RequestFromAdminPort] = true;
Page.RegisterHiddenField("__REQUESTDIGEST", globalAdmin.AdminFormDigest);
```

This security validation uses the **AdminFormDigest** property of the **SPGlobalAdmin** class to insert a message digest on the page in the browser, registering the digest as a hidden field through the **RegisterHiddenField** method of the System.Web.Ul.Page class. In addition, the **RequestFromAdminPort** field specifies that the context of the request is through the administrative port.

To run custom code that uses types and members in the Windows SharePoint Services object model, users and groups must have the appropriate permissions assigned to them, just as when they interact with a site or list through the user interface. For more information on permissions, see Security, Users, and Groups Overview.

3.15 Converting Date and Time Values

Microsoft Windows® SharePoint™ Services 2.0 stores date and time values in Coordinated Universal Time (UTC), and almost all date and time values that are returned by members of the object model are in UTC. The one exception is list column values that are obtained through the indexer for the **SPListItem** class, which are in the local time for the site. Use the **DateslnUtc** property of the **SPQuery** class to. use the indexer to return values in UTC.

To convert values to local time from UTC, use the **UTCToLocalTime** method of the **SPTimeZo.ne** class, which can be accessed through the **RegionalSettings** property for the current site, as follows: mySite.RegionalSettings.TimeZone.Utc'I'imeToLocalTime(date). To convert values from local time to UTC, instead use the **LocalTimeToUTC** method.

In addition to converting between local and UTC time, you may also need to convert between date and time formats, for example, from IS08601 format (YYYY-11M-DDTHH:MM:SSZ) to System.Date'I'ime format (mm/dd/yyyy hh:mm:ss AM or PM), or vice versa. The **SPUtility** class provides several methods that can be used to convert or modify the format of date and time values, such as the following:

CreateIS08601DateTimeFromSystemDateTime -- from: system Date'Time format to IS08601 DateTime format (yyyy-mm-ddThh:mm:ssZ). This method is useful, for example, when building a query with a filter based on a **System.DateTime** value. The following example returns allitems in a document library that have been modified in the past five days:

[Visual Basic .NET]

```
Dim query As New SPQuery()
query.Query = String.Format("<Where><Gt><FieldRef Name='Modified'/>" _
  & "<Value Type='DateTime' StorageTZ='TRUE'>{0}</Value></Gt></Where>", _
      SPUtility.CreateISO8601DateTimeFromSystemDateTime(DateTime.UtcNow.AddDays(-5)))
```

[C#]

```
SPQuery query = new SPQuery();
```

```
query.Query = String.Format("<Where><Gt><FieldRef Name='Modified'/>" +
  "<Value Type='DateTime' StorageTZ='TRUE'>{0}</Value></Gt></Where>",
  SPUtility.CreateISO8601DateTimeFromSystemDateTime(DateTime.UtcNow.AddDays(-5)));
```

In the example, the **CreateIS08601DateTimeFromSystemDateTime** method converts a Date'Tlmeval IS08601 format for use within a query string in Collaborative Application Markup Language (CAML), **Format** method of **System.String** inserts the converted value into the query string.

CreateSystemDateTimeFromXmlDataDateTimeFormat·~ from IS08601 DateTime forma **System.DateTime** format (opposite of the previous method).

This method can be used, for example, in a document library event handler where custom **DateTime** er values are returned through the **PropertiesBefore** and **PropertiesAfter** properties of the **SPListEven1** as strings in IS08601 format This method.can be used to convert the strings to a **DateTime** value, **FormatDate -** from **System.DateTime** format to a specified **SPDateFormat** format using the c regional settings of the site.

ParseDate - from the specified strings containing date and time values to a **System.DateTime** objec the current regional settings of the site.

3.16 Optimizing Code Performance

Avoid creating and destroying objects unnecessarily in code, as this may require that extra queries l: against the database arid may even involve code that is incorrect.

In the following example, separate objects for the Tasks list must be instantiated each time the indexe: in setting properties or calling the method for updating. This is not a recommended practice.

Example:.(Not Recommended)

```
[Visual Basic .NET]

Dim myWeb As SPWeb = SPControl.GetContextWeb(Context)

myWeb.Lists("Tasks").Title = "List_Title"
myWeb.Lists("Tasks").Description = "List_Description"
myWeb.Lists("Tasks").Update()
[C#]

SPWeb myWeb = SPControl.GetContextWeb(Context);

myWeb.Lists["Tasks"].Title = "List_Title";
myWeb.Lists["Tasks"].Description = "List_Description";
myWeb.Lists["Tasks"].Update();
```

The following example instantiates the **Tasks list** object only once and assigns it to the myList variable in order to set properties and call the method.

Example: (Recommended)

```
[Visual Basic .NET]

Dim myWeb As SPWeb = SPControl.GetContextWeb(Context)

Dim myList As SPList = myWeb.Lists("Tasks")

myList.Title="List_Title"
myList.Description="List_Description"
myList.Update()
[C#]

SPWeb myWeb = SPControl.GetContextWeb(Context);

SPList myList = myWeb.Lists["Tasks"];

myList.Title="List_Title";
myList.Description="List_Description";
myList.Update();
```

The object models in the **Windows SharePoint Services** assembly optimize performance and minimize the number of SQL queries that are made. However, to monitor code performance, it is recommended that you use the SQL Profiler.

3.17 Setting the Culture and Language

If your code runs outside the context of Microsoft Windows SharePoint Services and calls into the SharePoint object model, when the code executes, the culture of the current thread is set according to the settings of the operating system on the computer. To interact with Windows SharePoint Services, the currentUICulture and **Culture** values must be set to the values contained respectively by the **Language** and **Locale** properties of the **SPWeb** class.

Use the CurrentUICulture property to specify the language index to load and the **CurrentCulture** property to specify the formatting of numbers, date/time values, and so on. Following is an example of how to set these properties:

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new CultureInfo("de");

System.Threading.Thread.CurrentThread.CurrentCulture = new CultureInfo("de-DE");
```

In this example, the **CultureInfo** constructor requires that a **using** directive **(Imports** in Microsoft Visual Basic) be included for the **System.Globalization** namespace.

3.18 Displaying ASP.NET Error Messages

You can disable Microsoft Windows SharePoint Services error messaging so that ASP.NET error messages are displayed instead.

To display ASP.NET error messages, perform the following steps:

In the *Local_Drive:'ıProgram* Files\Common Files-Microsoft Shared\Web Serve Extensions\60\TEMPLATE\LAYOUTSfolder, change the name of the global.asax file to **global.bak.**

In the web.config file in the same folder, change **customErrors= "On" to customErrors="Off".**

3.19 Major Schema Files

Schema files in Microsoft® Windows® SharePoint™ Services use Collaborative Application Markıı Language (CAML) to define how data is displayed and how HTML is rendered. The following files are tr major schema files for the purpose of customizing site and list definitions:

DOCICON.XML
WEBTEMP.XML
ONET.XML
SCHEMA.XML

The following table describes other schema files that use CAML for data definition or HTML rendering, aı specifies their locations within the *Local_Drive:'ıProgram* Files\Common Files'Micrösoft, Shared\Web Sen Extensions\60\TEMPLATE directory.

Table 3.1

File name	Location	Description
STDVIEW.XML	\1033\Site_Definition\XML	Defines the base vi.ew used in a site definition wh creating new views. **Warning** Do not modify the contents of this file. Doi so might break the site definition.
[illegible]	\1033\Site Definition\XML	Defines the styles for viewing list data that are availa on the Create View page for the list.
BASE.XML	\1033\XML	Provides the schemas used in creating the Li Documents, and Userlnfo tables in the database whe new site is provisioned. **Warning** Do not modify the contents of this file. Do so might break the site definition.
DEADWEB.XML	\1033\XML	Defines the message that is sent to the owner of a sit, confirm_site usage r to warn that the site will automatically deleted.
FLDTYPES.XML	\1033\XML	Used during site or list Creation to define how field t) are rendered in the different modes for viewing list dan **Warning** Do not modify the contents of this file. Dı so might break the site definition.
INotifItem.xml	\1033\XML	Definei the. section that specifies- the item that

		!changed in an e-mail alert about changes to a list<>ritem.
NotifListHdr.xml	\1033\XML	1/Defi~es the section that specifies the name of the list in an e-mail alert about changes to a list or item.
NotifSiteFtr.xml	\1033\XML	ııDefi~esth~ footer section in an e-mail alert about changes to a list or item.
NotifSiteHdr.xml	\1033\XML	Defines the header section in an e-mail alert about changes to a list or item.
RGNLSTNG.XML	\1033\XML	Specifies the regional settings for currency, language, locale, and time zone.
htmltransinfo.xml	\XML	Contains mapping instructions for directing a request to the URL for handling a request when the client computer does not have Microsoft Office 2003 installed.

Files not located under a site definition folder (for example, STS or MPS) are global to a Windows SharePoint Services deployment on the front-end Web server and changing these files can affect the entire deployment. For more information, see Guidelines for Templates and Definitions.

For programming tasks that show how to customize the schema files, see Customizing Templates. For information about htmltransinfo.xml, see Programming with the Microsoft.HtmlTrans.InterfaceNamespace.

Whenever possible, create a site definition as described in Creating a Site Definition from an Existing Site Definition rather than modify the originally installed schema files. For general information about customizing site and list definitions, see Introduction to Templates and Definitions.

3.19.1 DOCICON.XML

Each front-end Web server in a Microsoft® Windows® SharePoint™ Services deployment contains a DOCICON.XML file located in the *Local_Drive\Progrnm* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\XMLfolder. DOCICON.XMLmaps file ProgIDs and file extensions to file icons and to the control to be used when opening each type offile.

You can add **Mapping** elements that map according to file ProgID or file extension.

Changes to DOCICON.XML should be made with extreme caution because they are global to a Windows SharePoint Services deployment and affect all site definitions On the front-end Web server. Changes that you make to this file may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

```
<DocIcons>
  <ByProgID>
    <Mapping
      Key="Excel.Sheet"
      Value="ichtmxls.gif"
      EditText="Microsoft Office Excel"
      OpenControl="SharePoint.OpenDocuments" />
    <Mapping
      Key="FrontPage.Editor.Document"
      Value="ichtmfp.gif"
      EditText="Microsoft Office FrontPage"
```

```
  OpenControl="SharePoint.OpenDocuments" />
  .
  .
  .
</ByProgID>
```

The **ByProglD** element maps the ProgIDs of different file types to file icons, specifies the text displayed l each file type in the drop-down menu that appears when the **Edit** arrow is clicked, and specifies the ProgID the control to use for opening documents of a given type.

```
<ByExtension>
 <Mapping
  Key="asax"
  Value="icasax.gif" />
 <Mapping
  Key="ascx"
  Value="icascx.gif" />
 <Mapping
  Key="asmx"
  Value="icasmx.gif" />
  .
  .
  .
 <Mapping
  Key="mpp"
  Value="icmpp.gif"
  EditText="Microsoft Office Project"
  OpenControl="SharePoint.OpenDocuments" />
 <Mapping
  Key="mps"
  Value="icmps.gif" />
 <Mapping
  Key="mpt"
  Value="icmpt.gif"
  EditText="Microsoft Office Project"
  OpenControl="SharePoint.OpenDocuments" />
  .
</ByExtension>
```

The **ByExtension** element maps file extensions to file icons, specifies the text displayed for eacl extension in the drop-down menu that appears when the Edit arrow is clicked, and specifies the ProgID control to use for opening documents with a given file extension.

To resolve conflicts, **ByProglD** takes precedence over ByExtension. However, document templates spe in the ONET.XML file of a site definition only work in relation to file types specified within the ByExte element.

```
<Default>
  <Mapping Value="icgen.gif" />
</Default>
<DocIcon
```

The **Default** element specifies the default image file to be used for file types not specified in DOCICON.XML.

For an example that demonstrates how to customize DOCICON.XML, see Adding a Document Template, File Type, and Editing Application to a Site Definition.

3.19.2 WEBTEMP.XML

Each front-end Web server in a deployment of Microsoft® Windows® SharePoint™ Services has at least one WEBTEMP.XML file located in the *Local_Drive\Prograrn* Files\Common Files\Microsoft Shared\Web Server. Extensions\60\TEMPLATE\l033\XML folder. A WEBTEMP.XML file contains the site definitions that are available on the Template Selection page for instantiating sites.

It is recommended that you create a custom site definition by copying an existing site definition and adding a WEBTEMP*.XML file that defines custom configurations.~orinsta11tiatingsites, rather than.modifying the original WEBTEMP.XML file .installed with Windows SharePoint Services. Changes that you make to originally installed files may be overwritten when .you iiisfa.11 updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version. For more information on the best practice to follow, see Creating a Site Definition frorn an Existing Site Definition.

3.19.2.1 Top-level elements

```
<Templates
xmlns:ows="Microsoft SharePoint">
<Template
Name="STS"
ID="1">
<Configuration
ID="0"
Title="Team Site"
Hidden="FALSE"
ImageUrl="/_layouts/images/stsprev.png"
Description="This template creates a site for teams to create,
organize, and share information quickly and easily. It includes
a Document Library, and basic lists such as Announcements,
Events, Contacts, and Quick Links." />
.
.
.
<Configuration
ID="2"
```

```
    Title="Document Workspace"
    Hidden="FALSE"
    ImageUrl="/_layouts/images/dwsprev.png"
    Description="This template creates a site for colleagues to work
     together on documents. It provides a document library for
     storing the primary document and supporting files, a Task list
     for assigning to-do items, and a Links list for resources related
     to the document." />
  </Template>
  <Template
   Name="MPS"
   ID="2">
   <Configuration
    ID="0"
    Title="Basic Meeting Workspace"
    Hidden="FALSE"
    ImageUrl="/_layouts/images/mwsprev.png"
    Description="All the basics to plan, organize and track your meeting.
     This Meeting Workspace contains the following lists: Objectives,
     Attendees, Agenda, and Document Library." />
     .
     .
     .
  </Template>
</Templates>
```

The **Templates** element specifies the configurations that can be used to instantiate sites. Each Temj specifies a unique ID and a name that corresponds to a site definition subfolder within the \TEMPLATE\ directory. A **Template** element can contain any number of **Configuration** subelements, each specifyi unique ID corresponding to the ID of a configuration in an ONET.XML file that specifies the lists modules of a site definition. Each **Configuration** element also specifies the title, description, and virtual to the preview image that is displayed on the Template Selection page. A configuration can be hidden the UI by setting **Hidden to TRUE.**

For more information about creating a custom configuration, see Using Configurations

3.19.3 FLDTYPES.XML

Each front-end Web server in a deployment of Microsoft® Windowsec SharePoint™ Services ha FLDTYPES.XML file located in the *Local_Drive\Program* Files\Common Files\Microsoft Sharee Server Extensions\60\TEMPLATE\1033\XML folder that is used during site or list creation to define field types are rendered in the different modes for viewing list data.

Modifying FLDTYPES.XML can break an installation of Windows SharePoint Services and is not supp

3.19.3.1 Top-level elements

```
<List
xmlns:ows="Microsoft SharePoint"
Name="FieldTypes"
Title="Field Definitions">
<MetaData>
 <Fields>
  <Field
   Type="Text"
   Name="TypeName"
   DisplayName="TypeName" />
  <Field
   Type="Text"
   Name="InternalType"
   DisplayName="InternalType" />
   .
   .
   .
  <RenderPattern
   Type="Note"
   Tall="TRUE"
   Name="HeaderPattern"
   DisplayName="HeaderPattern" />
  <RenderPattern
   Type="Note"
   Tall="TRUE"
   Name="DisplayPattern"
   DisplayName="DisplayPattern" />
  <RenderPattern
   Type="Note"
   Tall="TRUE"
   Name="EditPattern"
   DisplayName="EditPattern" />
   .
   .
   .
 </Fields>
 <MetaData>
```

The **MetaData** element in FLDTYPES.XML specifies the names and data types of properties that are used to define fields types within the **Data** element.

```
<Data>
 <Rows>
  <Row>
   <Field
    Name="TypeName"
    DisplayName="TypeName">Counter</Field>
    .
    .
    .
  </Row>
  <Row>
   <Field
    Name="TypeName"
    DisplayName="TypeName">Text</Field>
    <Field
     Name="InternalType"
     DisplayName="InternalType">Text</Field>
     <Field
      Name="SQLType"
      DisplayName="SQLType">nvarchar</Field>
      <Field
       Name="Sortable"
       DisplayName="Sortable">TRUE</Field>
      <Field
       Name="Filterable"
       DisplayName="Filterable">TRUE</Field>
      <RenderPattern
       Name="HeaderPattern"
       DisplayName="HeaderPattern">
        .
        .
        .
       <RenderPattern
        Name="DisplayPattern"
        DisplayName="DisplayPattern">
         .
         .
         .
       <RenderPattern
        Name="EditPattern"
        DisplayName="EditPattern">
         .
         .
```

```
.
<RenderPattern
 Name="NewPattern"
 DisplayName="NewPattern">
 .
 .
 .
<RenderPattern
 Name="PreviewDisplayPattern"
 DisplayName="PreviewDisplayPattern">
 . .
 .
 .
<RenderPattern
 Name="PreviewEditPattern"
 DisplayName="PreviewEditPattern">
 .
 .
 .
<RenderPattern
 Name="PreviewNewPattern"
 DisplayName="PreviewNewPattern">
 .
 .
 .
<RenderPattern
 Name="HeaderBidiPattern"
 DisplayName="HeaderBidiPattern">
 .
 .
 .
<RenderPattern
 Name="DisplayBidiPattern"
 DisplayName="DisplayBidiPattern">
 .
 .
 .
<RenderPattern
 Name="EditBidiPattern"
 DisplayName="EditBidiPattern">
 .
 .
 .
```

```
<RenderPattern
  Name="NewBidiPattern"
  DisplayName="NewBidiPattern">
  .
  .
  .
</Row>
.
.
.
</Rows>
</Data>
```

The **Data** element contains definitions for each of the basic field types used in Windows SharePointServ specifying how they are displayed in the different modes for viewing list data. Each **Row** element contain definition for a specific field type.

Field elements are used at the beginning of each row to specify property values for a field type in relati the properties defined within the **MetaData** element. For example, the field type used to display text is n Text, its internal type is Text, its SQL type is nvarchar, and the field is both sortable and filterable.

RenderPattern elements in-each zow define how an item is displayed in each of the possible mod viewing list data. These mod~s include the header patterns used at the top of ~achlist in the toolb modes used in forms for displ~ying, editing, or creating items, the equivalentmodes,usedfor lists that c bi-directional text, and preyi~yırendering patterns used by a Web~editi1 1g application that is compatibl Windows SharePointServices, such as Microsoft® Office FrontPage® 2003.

To create a custom field thaf>derives from a base field type, you can add a field definition SCHEMA.XML file of a list .1efinition. For programming tasks that show how to add custom fieldı schemas, see Creating a List Definition and Adding a Field to a List Definition.

3.19.4 ONET.XML

Each site definition on a front-end Web server has one ONET.XML file located in the *Local DriveIJ* Files\Common Files\Microsoft Shared\Web - Extensions\60\TEMPLATE\1033\Site_Definition_NameıXMLfolder. ONET.XML does the followin,

Defines the top and side navigation areas that appear on the home page and in list views.

Specifies the list definitions that are used in the site definition and whether they are available for ere on the Create page.

Specifies document templates that are available for creating document library lists on the New Library page and specifies the files used in the document templates.

Defines the base list types from which default Microsoft® Windows® SharePoint™ Services lists are derived.

Specifies the configurations oflists and modules that are used within site definitions.

The following kinds of tasks can be performed in ONET.XML to customize a site definition:

Specify an alternate cascading style sheet (CSS) file, JavaScript file, or ASPX header file for a site definition.

Modify navigation areas for the home page and list pages.

Add a list definition as an option to the Create page.

Add a document template for creating document libraries.

Define a configuration for a site definition, specifying the lists, modules, files, and Web Parts that are included when a site is instantiated.

Making changes to an originally installed ONET.XML file on a server running Windows SharePoint Services can break existing sites and the changes may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade aılinstallation to the next product version. Whenever possible, create a new site or list definition as opposed to modifying original schema files. For more information on the best practice to follow, see Creating a Site Definition-from an Existing Site Definition .

3.19.4.1 Top-level elements

```
<Project
 Title="Team Web Site"
 ListDir="Lists"
 xmlns:ows="Microsoft SharePoint">
```

The top-level **Project** element specifies a default name for sites created through the site definition and specifies the directory containing subfolders that the files for each list definition reside in. The **AlternateCSS** attribute can be used to specify an alternate CSS file for style definitions, the **CustomJSUrl** attribute carı be used to s:pecifyan alternate JavaScript file for custom script, and the **AlternateHeader** attribute can be used to specify an ASPX file for alternate headers. To prevent users from customizing sites with specific editing features in Microsoft FrontPage, specify which features through the **DisableWebDesignFeatures** attribute.

```
<NavBars>
<NavBar
 Name="SharePoint Top Navbar"
 Separator="   "
 Body="<a ID='onettopnavbar#LABEL_ID#' href='#URL#' accesskey='1'>#LABEL#</a>"
 ID="1002">
```

```
<NavBarLink
  Name="Documents and Lists"
  Url="_layouts/[LCID]/viewlsts.aspx" />
  .
</NavBar>
<NavBar
  Name="Pictures"
  Prefix="<table border=0 cellpadding=4 cellspacing=0>"
  Body="<tr><td><table border=0 cellpadding=0 cellspacing=0><tr><td>
   <img src='/_layouts/images/blank.gif' ID='100' alt='Icon' border=0>
    </td><td valign=top><a ID=onetleftnavbar#LABEL_ID#
   href='#URL#'>#LABEL#</td></tr></table></td></tr>"
  Suffix="</table>"
  ID="1005" />
  .
</NavBars>
```

The **NavBars** element contains definitions for the top navigation area that is displayed on the home pag list views, and definitions for the side navigation area that is displayed on the home page. Links can be to either the top or side by adding a **NavBarLink** element, and an entire **NavBar** section can be ad, grouping new links in the side area. Each **NavBar** element specifies a display name and a unique İD navigation bar, and defines how to display the navigation bar.

For a programming task on modifying navigation bars, see Customizing the Navigation Areas.

```
<ListTemplates>
  .
  .
  .
  <ListTemplate
  Name="tasks"
  DisplayName="Tasks"
  Type="107"
  BaseType="0"
  OnQuickLaunch="TRUE"
  SecurityBits="11"
  Description="Create a tasks list when you want to track a group of work items that you or your
   team needs to complete."
  Image="/_layouts/images/ittask.gif" />
  .
  .
  .
```

The list templates section specifies the list definitions that are part of a site definition. Each **List** element specifies an internal name that identifies the list definition with ϐ /Lists directory subı contains the SCHEMA.XML file, .aspx files, and other files used in the list definition. The List

element also specifies a display name for the list definition and whether the option to add a link on the Quick Launch bar appears selected by default on the New List page. In addition, this element specifies the description of the list definition and the path to the image representing the list definition, which are both displayed on the Create page. If **Hidden="TRUE"** is specified, the list definition does not appear as an option on the Create page.

The **SecurityBits** attribute is not used in Windows SharePoint Services.

The **ListTemplate** element has two attributes for type, **Type** and **BaseType.** The **Type** attribute specifies a unique identifier for the list definition, while BaseType identifies the base list type for the list definition, and corresponds to the **Type** value specified for one of the base list types defined within ONET.XML.

The **Path** attribute can be used to specify the site definition that contains a particular list definition (for example, **STS),** thus allowing a list definition from another site definition to be included in the current site definition. For example, the following **ListTemplate** elements are used in the ONET.XML file of the **MPS** site definition to include the Web Part Gallery and Data Sources list definitions, which are both defined within the **STS** site definition:

```
<ListTemplate
Path="STS"
Name="wplib"
DisplayName="Web Part Gallery"
Type="113"
BaseType="1"
Hidden="TRUE"
HiddenList="TRUE"
Unique="TRUE"
RootWebOnly="TRUE"
Catalog="TRUE"
SecurityBits="11"
Description="Place to store Web Parts"
Image="/_layouts/images/itdl.gif"
DocumentTemplate="100" />
<ListTemplate
Path="STS"
Name="datasrcs"
DisplayName="DataSources"
Type="110"
BaseType="1"
Hidden="TRUE"
HiddenList="TRUE"
SecurityBits="11"
Description="Data sources for this web."
Image="/_layouts/images/itdl.gif"
DocumentTemplate="100" />
```

For a programming task involving list templates, see Creating a List Definition.

```
<DocumentTemplates>

 <DocumentTemplate
  DisplayName="Microsoft Office Word document"
  Type="101"
  Default="TRUE"
  Description="A blank Microsoft Office Word document.">
  <DocumentTemplateFiles>
   <DocumentTemplateFile
    Name="doctemp\word\wdtmpl.doc"
    TargetName="Forms/template.doc"
    Default="TRUE" />
  </DocumentTemplateFiles>
 </DocumentTemplate>

</DocumentTemplates>
```

The document templates section defines the document templates that are listed on the New Document Libra- page. Each **DocumentTemplate** element specifies a display name, a unique identifier, and a description f the document template. If **Default** is set to **TRUE,** the template is the default template selected for docume libraries.

Each **DocumentTemplate** element contains a collection of **DocumentTemplateFile** element.s The **Nar** attribute of each **DocumentTemplateFile** element specifies the relative path to a local file that serves as t template. The **TargetName** attribute specifies the destination URL of the template file when a docume library is created. The **Default** attribute specifies whether the file is the default template file.

For a programming task involving document templates, see Adding a Document Template, File Type, a Editing Application to a Site Definition.

```
<BaseTypes>
 <BaseType
  Title="Generic List"
  Image="_layouts/images/itgen.gif"
  Type="0">
  <MetaData>
   <Fields>
    <Field
     ColName="tp_ID"
     ReadOnly="TRUE"
     Type="Counter"
     Name="ID"
     PrimaryKey="TRUE"
     DisplayName="ID" />
```

```
    <Field
     Type="Text"
     Name="Title"
     DisplayName="Title"
     Required="TRUE" />
     .
     .
     .
   </Fields>
  </MetaData>
 </BaseType>
 .
 .
 .
</BaseTypes>
```

The **BaseTypes** element is used during site or list creation to define the five list types on which all list definitions in Windows SharePoint Services are based. Each list template that is specified in the list templates section is identified with one of the base types: **Generic List, Document Library, Discussion Forum, Vote or Survey, or Issues List.**

Each Base'Iype element specifies the fields used in lists that are derived from the base type. The **Type** attribute of each **Field** element identifies the field with a field type that is defined in FLDTYPES.XML.

Do not modify the contents of this section, because doing so can break the site definition. Base list types cannot be added. For information on how to add a list definition, see Creating a List Definition.

```
<Configurations>
 .
 <Configuration
  ID="0"
  Name="Default">
  <Lists>
   <List
    Title="Shared Documents"
    Url="Shared Documents"
    QuickLaunchUrl=
     "Shared Documents/Forms/AllItems.aspx"
    Type="101" />
    .
    .
    .
  <Modules>
   <Module Name="Default" />
   <Module Name="WebPartPopulation" />
```

```
  </Modules>
 </Configuration>
 .
 .
 .
</Configurations>
```

Each **Configuration** element in the configurations section specifies the lists and modules that are created l default when the site definition is instantiated. The **ID** attribute uniquely identifies the configuration ar corresponds to the **ID** attribute of a **Configuration** element in WEBTEMP.XML.

Each **List** element specifies the title of the list definition and the URL for where to create the list. T **QuickLaunchUrl** attribute can be used to set the URL of the view page to use when adding a link on t Quick Launch bar to a list created from the list definition. The value of the **Type** attribute corresponds to t **Type** attribute of a template **in** the list templates section. Each **Module** element specifies the name ol module defined in the modules section.

For post-processing capabilities, use an **ExecuteUrl** element within a **Configuration** element to specify URL that is called following instantiation of the site.

A **Description** attribute must be specified in the List element when adding a custom list definition t configuration. The **Description** attribute overrides the **DefaultDescription** element specified within corresponding SCHEMA.XML file.

For more information on configurations, see Using Configurations.

```
<Modules>
 <Module
  Name="Default"
  Url=""
  Path="">
  <File
   Url="default.aspx"
   NavBarHome="True">
   <View
    List="104"
    BaseViewID="0"
    WebPartZoneID="Left" />
   <View
    List="106"
    BaseViewID="0"
    WebPartZoneID="Left"
     WebPartOrder="2" />
   <AllUsersWebPart
    WebPartZoneID="Right"
    WebPartOrder="1">
    <![CDATA[
```

```
    <WebPart xmlns="http://schemas.microsoft.com/WebPart/v2" xmlns:iwp="http://schemas.microsoft.com/WebPart/v2/Image">
     <Assembly>Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c</Assembly>
     <TypeName>Microsoft.SharePoint.WebPartPages.ImageWebPart</TypeName>
     <FrameType>None</FrameType>
     <Title>Site Image</Title>
     <iwp:ImageLink>/_layouts/images/homepage.gif</iwp:ImageLink>
    </WebPart> ]]>
   </AllUsersWebPart>
   <View
    List="103"
    BaseViewID="0"
    WebPartZoneID="Right"
    WebPartOrder="2" />
   <NavBarPage
    Name="Home"
    ID="1002"
    Position="Start" />
   <NavBarPage
    Name="Home"
    ID="0"
    Position="Start" />
  </File>
 </Module>


 <Module Name="WebPartPopulation" List="113" Url="_catalogs/wp" Path="lists\wplib\dwp" RootWebOnly="TRUE">
  <File Url="MSContentEditor.dwp" Type="GhostableInLibrary" />
  <File Url="MSPageViewer.dwp" Type="GhostableInLibrary" />
  <File Url="MSImage.dwp" Type="GhostableInLibrary" />
  <File Url="MSMembers.dwp" Type="GhostableInLibrary" />
  <File Url="MSSimpleForm.dwp" Type="GhostableInLibrary" />
  <File Url="MSXml.dwp" Type="GhostableInLibrary" />
 </Module>



</Modules>
```

The **Modules** collection specifies the modules to include by default in creating a site collection. Each **Module** element in turn specifies one or more files to include, often for Web Parts, which are cached in memory on the front-end Web server along with the schema files. The **Module** element also specifies a name for the module, which corresponds to a module name that is specified within a configuration in ONET.XML. When the module is used to specify a group of files, such as the Web Part (.dwp) files of the default

WebPartPopulation module, the **Uri** attribute is used to specify the virtual directory fgr>the Web F Galleries of site collections that are created through the site definition, and the **Path** attribiite specifies physical directory in the file system where the files reside/Setting **RootWebOnly to TRUE** specifies that files will be created only for the top-level site of a site collection and not for individual sub-sites in collection.

The **Uri** attribute of each **File** element in a module specifies the name of a file to create when a site js crea When the module includes a single file, such as default.aspx, **NavBarHome="TRUE"** specifies that the will serve as the destination page for the **Home** link in navigation bars. The **File** element for default.aspx specifies the Web Parts to include on the home page and information about the home page for other pages link to it. The **Type** attribute can specify either **Ghostable or GhostableinLibrary,** with either value the will be cached in memory ("ghosted"). When customizations are made to the file through the UI, such : the home page, only changes from the original definition are stored in the database, while the remainder o page definition is contained in the cached file. **GhostablelnLibrary** specifies that the file or files be ca as part of a list whose base type is **Document Library.**

For more information about modules, see Using Modules to Add Files to a Site Definition.

For information about guidelines to follow in customizing site definitions, see Guidelines for Template Definitions.

Changing a module on a front-end Web server has no effect on existing sites.

3.19.5 SCHEMA.XML

Each list definition that appears ··as an option on the Create page has its own subfok *Local_Drive\Program* Files\Common Files\Microsoft Shared\Web Extensions\60\TEMPLATE\l 033\Site*_Definition_ Name\LISTS'J.,ist _Definition _Name* that includ SCHEMA.XML file. The SCHEMA.XML file defines the views, forms, toolbar, and special fields f lists that are created through the list definition.

The following kinds of tasks can be performed in a SCHEMA.XML file to customize a list definition:

- Add custom fields based on field types defined in FLDTYPES.XML.
- Create a custom view for lists created through the list definition.
- Create custom forms for working with list items.
- Specify the default description that is displayed for the list in the UI.
- Define the Actions area that is displayed in the side navigational area of list views.

Making changes to an originally installed SCHEMA.XML file on a server rnnning Microsoft® Wii SharePoint™ Services can break existing sites and the changes may be overwritten when you install or service packs for Windows SharePoint Services, or when you upgrade an installation to the next version. Whenever possible, create a new site or list definition as opposed to modifying original schei For more information on the best practice to follow, see Creating a Site Definition from an Exisı Definition.

3.19.5.1 Top-level elements

```
<List
xmlns:ows="Microsoft SharePoint"
Name="Tasks"
Title="Tasks"
Direction="0"
```

```
Url="Lists/Tasks"
BaseType="0">
```

The top-level **List** element specifies the internal name and display name for the list definition, as well as the direction of text used in lists, and the site-relative URL at which lists are created. Like most attibutes of the **List** element in a SCHEMA.XML file, specification of the site-relative URL is largely irrelevant because it is overridden by list settings in the ONET.XML file of the site definition. This element optionally specifies **Microsoft SharePoint** as an XML namespace. The **BaseType** attribute specifies a base list type that is defined within the **BaseTypes** element of ONET.XML. Use the **VersioningEnabled** attribute to specify whether versioning is enabled by default for document libraries created through the list definition.

```
<MetaData>
 <Fields>
  <Field
   Type="Choice"
   Name="Priority"
   DisplayName="Priority">
   <CHOICES>
    <CHOICE>(1) High</CHOICE>
    <CHOICE>(2) Normal</CHOICE>
    <CHOICE>(3) Low</CHOICE>
   </CHOICES>
   <Default>(2) Normal</Default>
  </Field>
  .
  .
  .
  <Field
   Type="Number"
   Name="PercentComplete"
   Percentage="TRUE"
   Min="0"
   Max="1"
   DisplayName="% Complete" />
  <Field
   Type="User"
   List="UserInfo"
   Name="AssignedTo"
   DisplayName="Assigned To" />
   .
   .
   .
 </Fields>
```

The **Fields** element contains field definitions for special fields that may be required in a list definitio
Field element specifies a display name, an internal name, a field type defined in FLDTYPES.XML 01
the field is based, and other field properties as required.

For programming tasks that show how to add custom fields, see Creating a List Definition and Adding
to a List Definition.

```
<Views>
 <View
 BaseViewID="0"
 Type="HTML">
 .
<View
 BaseViewID="2"
 Type="HTML"
 WebPartZoneID="Main"
 DisplayName="My Tasks"
 Url="MyItems.aspx"
 ReqAuth="TRUE">
 <GroupByHeader>
 .
 <GroupByFooter>
 .
 <ViewHeader>
 .
 <ViewBody>
 .
 <ViewFooter>
 .
 <PagedRowset>
 .
 <PagedRecurrenceRowset>
 .
 <RowLimit
  Paged="TRUE">100</RowLimit>
 <ViewEmpty>
 .
 <ViewBidiHeader>
 .
 <Toolbar
  Type="Standard">
 .
 <ViewFields>
  <FieldRef Name="LinkTitle" />
```

```
<FieldRef Name="Status" />
<FieldRef Name="Priority" />
<FieldRef Name="DueDate" />
<FieldRef Name="PercentComplete" />
</ViewFields>
<Query>
<Where>
<Eq>
<FieldRef Name="AssignedTo" />
<Value Type="Integer">
<UserID />
</Value>
</Eq>
</Where>
<OrderBy>
<FieldRef Name="Status" />
<FieldRef Name="Priority" />
</OrderBy>
</Query>
</View>

</Views>
```

The Views element contains the definitions for views that are availableby default when a list is created. Each View element specifies the type of format used in the display (usually **HTML)** and a unique ID for the view. When the view is displayed In a Web Part, the **View** element also specifies the title of the view. The **Uri** attribute is used in list creation to specify the base name of the .ASPX page in which the view is displayed. The **View** also specifies the Web Part zone ID of the Web Part in which the view is displayed.

The **Vi**ew element contains subelements that define the various parts of a view, including the header, body, and focter, but also the Group By section, a limit on the number of rows, or items, to display, rowsets that define how to display items when the number exceeds the row limit, what to display when no items are returned in the view, the toolbar area that·is displayed above lists, the fields displayed in the view, and the query **th**at filters the view.

```
<Forms>
<Form
Type="DisplayForm"
Url="DispForm.aspx"
WebPartZoneID="Main">

</Form>
<Form
Type="EditForm"
Url="EditForm.aspx"
WebPartZoneID="Main">
```

```
<ListFormOpening>
.
<ListFormButtons>
.
<ListFormBody>
.
<ListFormClosing>
.
</Form>
</Forms>
```

The **Forms** element contains definitions of the forms used in working with individual list items. Each **Forrr** element specifies the form type, which can be **DisplayForm, EditForm, or New Form,** as well as the name o the ASPX page used for the form and the Web Part zone ID of the Web Part in which the form is displayed oı the page. Each **Form** element can also contain subelements that define the different parts of the form.

```
<DefaultDescription>Use the Tasks list to keep track of work
that you or your team needs to complete.</DefaultDescription>
```

A **DefaultDescription** element specifies the description that is displayed for the list definition in the UL Th element is overridden by the **Description** attribute of the **List** element corresponding to the current li definition that is specified within a configuration in ONET.XML.

A value for this element must be specified in custom list definitions.

```
<Toolbar
Type="RelatedTasks">
.
</Toolbar>
```

The **Toolbar** element defines the kind of toolbat that is displayed in the navigational areas of list. vie· Possible values include the following:

RelatedTasks for the Actions section that is displayed in the side navigational area of list views.

Standard for normal views.

Freeform for summary views on home pages.

3.19.6 Programming Tasks

This section contains information about the types of customizations that can be made in Microsoft® Windows® SharePoint™ Services:

Programming with the Object Model

Customizing Templates

Programming Web Services

Programming with the Microsoft.SharePoint.MeetingsNamespace

3.19.7 Programming with the Object Model

This section contains the following programming tasks:

- Creating a Web :A.'pplicatfori ôri a ShafePoint Web Site
- Creating a Console Application
- Customizing a Web Part
- Returning Sites and Site Collections
- Creating or Deleting a Site or a Site Collection
- Creating or Deleting Lists
- Adding or Deleting a List in Multiple Web Sites
- Returning Items from a List
- Adding or Deleting List Items
- Adding a Recurring Event to Lists on Multiple Sites
- Accessing, Copying, and Moving Files
- Uploading a File to a SharePoint Site from aLöcal' Folder
- Handling Document Library Events
- Adding or Removing Users
- Returning Central Administrative Properties
- Setting the Administrative Port Identity

3.19.7.1 Creating a Web Application on a SharePoint Web Site

This programming task describes how to create a Web application in Microsoft Visual Studio .NET. This example creates a tool for reporting a list of all the SharePoint Web sites to which a specified user belongs, as well as a list of that user's membership in the security groups for each Web site.

1. In Visual Studio .NET, create a new Web application and set a reference to Microsoft.SharePoint.dll. For information on how to perform these steps, see Getting Started with Customizing a Team Web Site in Visual Studio .NET.

2. In Design View, use the Toolbox to add a label, a text box, anda button to WebForml .aspx. To open the Toolbox, click **Toolbox** on the **View** menu.

To create a Web application that modifies data in the content database, you need to add a **FormDigest** contn and a page directive to the .aspx file containing the form. For information on how to add this control, se Security Validation and Making Posts to Update Data.

3. Double-click the button to open the code-behind file named WebForml.aspx.cs.

4. At the beginning of the file, add **using** directives·**(lınports** statements in Visual Basic .NET) th reference the **Microsoft.SharePoint** , **Microsoft.SharePoint.Utilities** ar **Microsoft.SharePoint.WebControls**namespaces, as follows:

```
[Visual Basic .NET]
Imports Microsoft.SharePoint
Imports Microsoft.SharePoint.WebControls
Imports Microsoft.SharePoint.Utilities
[C#]
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using Microsoft.SharePoint.Utilities;
```

The **Microsoft.SharePoint.WebControls**namespace must be referenced in order to instantiate a site obj that corresponds to the current site context. The Microsoft.Sharel'ofnt namespace must be referenced order to return the specified user and the groups to which the user belongs. 1 **Microsoft.SharePoint.Utilities**namespace must be referenced in order to encode stringsthat are displayed the example.

After you instantiate the site object using SPSite mySite = SPControl.GetContextSite(Context)(in. Vis Basic .NET, use Dim mySite As SPSite = SPControl.GetContextSite(Context)),you can use the. **AllW** property of the **SPSite** class, the **Users** property of the **SPWeb** class, and the **Roles** property of the **SPU:** class within nested **foreach** statements in order to return a user object for the user name specified in the t box, as well as the list of site groups to which that user belongs.

The **Context** value that is passed in the **GetContextSite**method is inherited from System.Web.Ul.Page.

Within the Buttonl_Click handler that Visual Studio .NET creates on the code-behind page, add the follow code block:

```
[Visual Basic .NET]

Dim userList As String = SPEncode.HtmlEncode(TextBox1.Text) & " is a user in the following webs:<BR>"
Dim mySite As SPSite = SPControl.GetContextSite(Context)
Dim allSites As SPWebCollection = mySite.AllWebs
Dim subSite As SPWeb

For Each subSite In allSites

  Dim listGroups As String = ""
  Dim allUsers As SPUserCollection = subSite.Users
  Dim user As SPUser

  For Each user In allUsers
```

```
    If user.LoginName = TextBox1.Text Then

        Dim allGroups As SPRoleCollection = user.Roles
        Dim group As SPRole

        For Each group In  allGroups

            listGroups += SPEncode.HtmlEncode(group.Name) & " "

        Next group

        userList += subSite.ServerRelativeUrl.ToString() & " :: " & listGroups & "<BR>"
    End If

  Next user

Next subSite

Label1.Text = userList
```

[C#]

```
string userList = SPEncode.HtmlEncode(TextBox1.Text) + " is a user in the following webs:<BR>";

SPSite mySite = SPControl.GetContextSite(Context);
SPWebCollection allSites = mySite.AllWebs;

foreach (SPWeb subSite in allSites)
{
string listGroups = "";

SPUserCollection allUsers = subSite.Users;

foreach (SPUser user in allUsers)
{

if (user.LoginName == TextBox1.Text)
{
        SPRoleCollection allGroups = user.Roles;

        foreach (SPRole group in allGroups)
        {
        listGroups += SPEncode.HtmlEncode(group.Name) + " ";
```

```
            }

            userList += subSite.ServerRelativeUrl.ToString() + " -- " + listGroups + "<BR>";
        }
    }
}

Label1.Text = userList;
```

The code example iterates through all the users of all the subsites for the current top-level site, and if the valı.ie of the LoginName property for a given user matches the user name that is entered in the text box, .then the. names of all the user's site groups for the site are collected and displayed.

5. On the Build menu, click Build Solution.

The browser opens the page. When you type the logon name of a user on the site, the label displays the Web sites and site groups to which the specified user belongs.

6. To run the application after creating it in Visual Studio .NET, navigate to http:// *Server , Name/[* sites/][*Site_Name/]__Iayouts!Web_Application_Namelwebforrn* l .aspx.

3.19.7.2 Creating a Console Application

This programming task describes how to create a console application in Microsoft Visual Studio .NET. Th, example displays the number of lists within a site collection.

For users to run a console application in the context of Microsoft Windows SharePoint Services 2.0, the must be administrators on the computer where the script is executed.

On the File menu in Visual Studio .NET, point to New and then click Project.

In the New Project dialog box, select Visual Basic Projects or Visual C# Projects in the Project Typı box, depending on which language you prefer.

In the Templates box, select Console Application.

In the Location box, type the path to where to create the application, and then click OK

In Solution Explorer, right-click the References node, and then click Add Reference on the shortcut meni

On the .NET tab of the Add Reference dialog box, select Windows SharePoint Services in the list components, click Select, and then click OK

In the .vb or .cs file, add a using directive for the Microsoft.SharePoint namespace, as follows.

[Visual Basic .NET]

```
Imports Microsoft.SharePoint
[C#]
using Microsoft.SharePoint;
```

Add the following code to the **Main** method in the .vb or .cs file.

```
[Visual Basic .NET]
Overloads Sub Main(args() As String)
  Dim siteCollection As New SPSite("http://Server_Name")
  Dim sites As SPWebCollection = siteCollection.AllWebs
  Dim site As SPWeb

  For Each site In sites

    Dim lists As SPListCollection = site.Lists

    Console.WriteLine("Site: " + site.Name + "  Lists: " + lists.Count.ToString())
  Next site
  Console.WriteLine("Press ENTER to continue")
  Console.ReadLine()

  End Sub 'Main
  [C#]
  static void Main(string[] args)
  {
    SPSite siteCollection = new SPSite("http://Server_Name");
    SPWebCollection sites = siteCollection.AllWebs;
    foreach (SPWeb site in sites)
    {
      SPListCollection lists =  site.Lists;
      Console.WriteLine("Site: " + site.Name + "  Lists: " + lists.Count.ToString());
    }
    Console.WriteLine("Press ENTER to continue");
    Console.ReadLine();
  }
```

Click **Start** on the **Debug** menu or press F5 to run the code.

3.19.7.3 Customizing a Web Part

You can create custom Web Parts to work with site or list data. This programming task shows how to create a simple Web Part that displays the titles and number of items for all lists that contain more than ten items on subsites in the current Web site.

Create a Web Part as described in Creating a Basic Web Part, This example assumes that you have created a SimpleWebPart application.

Open WebCustomControll.cs or WebCustomControll.vb for the SimpleWebPart application and add directives for the **Microsoft.SharePoint, Microsoft.SharePoint.Utilities,** and **Microsoft.SharePoint.WebControls** namespaces, as follows:

```
[Visual Basic .NET]

Imports Microsoft.SharePoint
Imports Microsoft.SharePoint.Utilities
    Imports Microsoft.SharePoint.WebControls
    [C#]

    using Microsoft.SharePoint;
    using Microsoft.SharePoint.Utilities;
    using Microsoft.SharePoint.WebControls;
```

Remove the **HtmlControl** objects used in the example, including declarations for their variables, t _mybutton_click handler, and the **CreateChildControls** method.

Replace the contents of the **RenderWebPart** method with the following codeblock,

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)

output.Write(SPEncode.HtmlEncode(mySite.Title))

Dim subSites As SPWebCollection = mySite.Webs
Dim site As SPWeb

For Each site In  subSites

  output.Write(SPEncode.HtmlEncode(site.Title) & "<BR>")

  Dim lists As SPListCollection = site.Lists
  Dim list As SPList

  For Each list In  lists

    If list.ItemCount > 10 Then

        output.Write(SPEncode.HtmlEncode(list.Title) & " :: " & list.ItemCount & "<BR>")

    End If

  Next list

    Next site
    [C#]
```

```
SPWeb mySite = SPControl.GetContextWeb(Context);

output.Write(SPEncode.HtmlEncode(mySite.Title));

SPWebCollection subSites = mySite.Webs;

foreach(SPWeb site in subSites)
{

  output.Write(SPEncode.HtmlEncode(site.Title) + "<BR>");

  SPListCollection lists=site.Lists;

  foreach(SPList list in lists)
  {

    if (list.ItemCount>10)
    {
      output.Write(SPEncode.HtmlEncode(list.Title) + " : " + list.ItemCount + "<BR>");
    }
  }
}
```

The example first writes out the title of the current Web site.Jt.then iterates through all the subsites to print out their titles, and then through all the lists in each subsite to print out the list title and number of items for cases where there are more than ten list items in a list.

On the **Build** menu, click **Build Solution.**

Increase the trust level in Windows SharePoint Services from minimal (default) to medium by opening the web.config file at *Local_Drive:\Inetpı.İb\wwwroot* and replacing the following line:

```
<trust level="WSS_Minimal" originUrl="" />
```

with the following:

```
<trust level="WSS_Medium" originUrl="" />
```

Reset Microsoft Internet Information Services (IIS) for changes in trust level to take effect.

The Web Part can be imported through the user interface into a Web Part Page or into the home page for viewting the list data.

3.19.7.4 Returning Sites and Site Collections

You can return all the sites within a site collection, including the top-level site and all subsites, by using the AUWebsproperty of the SPSite class. The following example displays the titles of all the sites and lists in the current site collection.

```
[Visual Basic .NET]

Dim mySite As SPSite = SPControl.GetContextSite(Context)
Dim subSites As SPWebCollection = mySite.AllWebs
Dim i As Integer

For i = 0 To subSites.Count - 1

  Dim lists As SPListCollection = subSites(i).Lists
  Dim j As Integer

  For j = 0 To lists.Count - 1

    Response.Write(SPEncode.HtmlEncode(subSites(i).Title) & " :: " _
      & SPEncode.HtmlEncode(lists(j).Title) & "<BR>")

  Next j

Next i
[C#]

SPSite mySite = SPControl.GetContextSite(Context);
SPWebCollection subSites = mySite.AllWebs;

for (ini i=0;i<subSites.Count;i++)
{
  SPListCollection lists = subSites[i].Lists;

  for (int j=0;j<lists.Count;j++)
  {
    Response.Write(SPEncode.HtmlEncode(subSites[i].Title) + " :: " +
      SPEncode.HtmlEncode(lists[j].Title) + "<BR>");
  }
}
```

The previous example requires using directives (Imports in Visual Basic) for the Microsoft.SharePo Microsoft.SharePoint. Utilities, and Microsoft.SharePoint. WebControls namespaces.

To return a list of all the first-tier subsites for a site, use the **Webs** property of the **SPWeb** class. The following example displays a list of subsite titles.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim sites As SPWebCollection = mySite.Webs
Dim subSite As SPWeb

For Each subSite In sites

  Response.Write(SPEncode.HtmlEncode(subSite.Title) & "<BR>")

Next subSite
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPWebCollection sites = mySite.Webs;

foreach (SPWeb subSite in sites)
{
  Response.Write(SPEncode.HtmlEncode(subSite.Title) + "<BR>");
}
```

The previous example requires **using** directives **(Imports** in Visual Basic) for the **Microsoft.SharePoint, Microsoft.SharePoint.Utilities,** and **Microsoft.SharePoint.WebControls** namespaces.

Instead of using a **foreach** statement, the following example uses nested **for** statements to display a list of subsite URLs and list titles.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim sites As SPWebCollection = mySite.Webs
Dim i As Integer

For i = 0 To sites.Count - 1

  Dim lists As SPListCollection = sites(i).Lists
  Dim j As Integer

  For j = 0 To lists.Count - 1

    Response.Write(sites(i).Url & " :: " & SPEncode.HtmlEncode(lists(j).Title) & "<BR>")
```

```
  Next j

Next i
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPWebCollection sites = mySite.Webs;

for (int i=0;i<sites.Count;i++)
{
  SPListCollection lists = sites[i].Lists;

  for (int j=0;j<lists.Count;j++)
  {
    Response.Write(sites[i].Url + " :: " + SPEncode.HtmlEncode(lists[j].Title) + "<BR>");
  }
}
```

The previous example requires **using** directives **(Imports** in Visual Basic) for the **Microsoft.ShareF Microsoft.SharePoint. Utilities,** and **Microsoft.SharePoint. WebControls** namespaces.

The next example displays all the subsites and lists for the current site, as well as the number of items in list. The example uses nested **foreach** statements to iterate through the collections of Web sites and lists.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim sites As SPWebCollection = mySite.Webs
Dim subSite As SPWeb

For Each subSite In  sites

  Response.Write("<B>" & SPEncode.HtmlEncode(subSite.Title) & "</B>" & "<BR>")

  Dim lists As SPListCollection = subSite.Lists
  Dim list As SPList

  For Each list In  lists

    Response.Write(SPEncode.HtmlEncode(list.Title) & " :: " & list.ItemCount & "<BR>")

  Next list

Next subSite
```

```
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPWebCollection sites = mySite.Webs;

foreach (SPWeb subSite in sites)
{
  Response.Write("<B>" + SPEncode.HtmlEncode(subSite.Title) + "</B>" + "<BR>");

  SPListCollection lists = subSite.Lists;

  foreach (SPList list in lists)
  {

    Response.Write(SPEncode.HtmlEncode(list.Title) + " :: " + list.ItemCount + "<BR>");
  }
}
```

The previous example requires **using** .directives **(Imports** in Visual Basic) for the **Microsoft.SharePoint, Microsoft.SharePoint.Utilities,** and **Microsoft.SharePoint.WebControls** namespaces.

To return the collection of site collections on a virtual server, use the **Sites** property of the **SPVirtualServer** class. Instantiate an **SPGlobalAdmin** object and use the **OpenVirtualServer** method to return a specific virtual server. The following example displays the URLs of all the site collections on a specified virtual server.

```
[Visual Basic .NET]

Dim globalAdmin As New SPGlobalAdmin()

Dim uri As New System.Uri("http://Server_Name")
Dim virtualServer As SPVirtualServer = globalAdmin.OpenVirtualServer(uri)
Dim siteCollections As SPSiteCollection = virtualServer.Sites
Dim siteCollection As SPSite

For Each siteCollection In  siteCollections

  Response.Write(siteCollection.Url + "<BR>")

Next siteCollection
[C#]

SPGlobalAdmin globalAdmin = new SPGlobalAdmin();

System.Uri uri = new System.Uri("http://Server_Name");
```

```
SPVirtualServer virtualServer = globalAdmin.OpenVirtualServer(uri);
SPSiteCollection siteCollections = virtualServer.Sites;

foreach (SPSite siteCollection in siteCollections)
{
  Response.Write(siteCollection.Url + "<BR>");
}
```

The previous example requires **using** directives **(Imports** in Visual Basic) for both **Microsoft.SharePoint.Administration** and **Microsoft.SharePoint. Utilities** namespaces.

3.19.7.5 Creating or Deleting a Site or a Site Collection

To create a site, use one of the **Add** methods of the **SPWebCollection** class. To create a subsite beneai site, use the **Webs** property of the **SPWeb** class to return the collection of subsites and call one of the *i* methods for the collection.

The following example creates a new subsite based on the template of the current site and on informa gathered from three text boxes. The text boxes specify the name to use in the new URL, the title to use foı site, and a description for the site.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim subSites As SPWebCollection = mySite.Webs
Dim currentTemplate As String = mySite.WebTemplate

Dim siteUrl As String = TextBox1.Text.ToString()
Dim siteTitle As String = TextBox2.Text.ToString()
Dim siteDescription As String = TextBox3.Text.ToString()

subSites.Add(siteUrl, siteTitle, siteDescription, Convert.ToUInt32(1033), currentTemplate, True, False)
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPWebCollection subSites = mySite.Webs;
string currentTemplate = mySite.WebTemplate;

string siteUrl = TextBox1.Text.ToString();
string siteTitle = TextBox2.Text.ToString();
string siteDescription = TextBox3.Text.ToString();

subSites.Add(siteUrl, siteTitle, siteDescription, 1033, currentTemplate, true, false);
```

In the example, the **WebTemplate** property of the **SPWeb** class returns the name of the cum definition, which is passed as a parameter of the **Add** method. In addition, three parameters for this pass the information gathered from the three text boxes, and three other parameters specify the locale

true to create a site with unique permissions, and **false** to convert any existing Web site at the same location to a SharePoint site.

To delete a site, use the **Delete** method of the **SPWeb** class or the **Delete** method of the **SPWebCollection** class.

The following example assumes the use of a text box to specify the URL of a site to delete and uses the Delete method of the **SPWebCollection** class to delete the site.

```
[Visual Basic .NET]

Dim deleteSite As String = TextBox1.Text.ToString()

Dim mySite As SPSite = SPControl.GetContextSite(Context)
Dim sites As SPWebCollection = mySite.AllWebs

sites.Delete(deleteSite)
[C#]

string deleteSite = TextBox1.Text.ToString();

SPSite mySite = SPControl.GetContextSite(Context);
SPWebCollection sites = mySite.AllWebs;

sites.Delete(deleteSite);
```

In the example, the **AllWebs** property of the **SPSite** class returns the collection of all sites within the current site collection.

To create a site collection, use the **Sites** property of the **SPVirtualServer** class to return the collection of site collections on the virtual server and use one of the **Add** methods of the **SPSiteCollection** class.

The following example creates a site collection within the collection of site collections for a specified virtual server.

```
[Visual Basic .NET]

Dim globAdmin As New SPGlobalAdmin()

Dim uri As New System.Uri("http://Server_Name")
Dim virtualServer As SPVirtualServer = globAdmin.OpenVirtualServer(uri)
Dim siteCollections As SPSiteCollection = virtualServer.Sites

siteCollections.Add("http://Server_Name/sites/Site_Collection_Name", "User_Name", "User_Email")
[C#]

SPGlobalAdmin globAdmin = new SPGlobalAdmin();
```

```
System.Uri uri = new System.Uri("http://Server_Name");
SPVirtualServer virtualServer = globAdmin.OpenVirtualServer(uri);
SPSiteCollection siteCollections = virtualServer.Sites;

siteCollections.Add("http://Server_Name/sites/Site_Collection_Name","User_Name","User_Email");
```

The example instantiates the **SPGlobalAdmin** class in order to call the **OpenVirtualServer** method r return the virtual server with the specified URI.

To delete a site collection from a virtual server requires security validation that uses the **AdminForml)i~** property of the **SPGlobalAdmin** class to insert a message digest on the page in thebro\\'.ser. This can bed, by registering the digest as a hidden field through the **RegisterHiddenField** method of System.Web.Ul.Page class. In addition, use the **RequestFromAdminPort** field of the **SPGlobalAdı** class to specify that the context of the request is through the administrativeport.

The following example uses the **Page_Load** event to include an administrative form digest on the page an establish the context of the request.

```
[Visual Basic .NET]

Private globalAdmin As New SPGlobalAdmin()

Private Sub Page_Load(sender As Object, e As System.EventArgs)

   Context.Items(SPGlobalAdmin.RequestFromAdminPort) = True
   Page.RegisterHiddenField("__REQUESTDIGEST", globalAdmin.AdminFormDigest)

End Sub 'Page_Load

Private Sub Button1_Click(sender As Object, e As System.EventArgs)

   Dim globalAdmin As New SPGlobalAdmin()
   Dim uri As New System.Uri("http://Server_Name")
   Dim vServer As SPVirtualServer = globalAdmin.OpenVirtualServer(uri)
   Dim siteCollections As SPSiteCollection = vServer.Sites

   siteCollections.Delete("sites/Site_Collection")

End Sub 'Button1_Click
[C#]

private SPGlobalAdmin globalAdmin = new SPGlobalAdmin();

private void Page_Load(object sender, System.EventArgs e)
{
   Context.Items[SPGlobalAdmin.RequestFromAdminPort] = true;
```

```
Page.RegisterHiddenField("__REQUESTDIGEST", globalAdmin.AdminFormDigest);
}

private void Button1_Click(object sender, System.EventArgs e)
{
SPGlobalAdmin globalAdmin = new SPGlobalAdmin();
System.Uri uri = new System.Uri("http://Server_Name");
SPVirtualServer vServer = globalAdmin.OpenVirtualServer(uri);
SPSiteCollection siteCollections = vServer.Sites;

siteCollections.Delete("sites/Site_Collection");
}
```

3.19.7.6 Creating or Deleting Lists

To create a new list, use the one of the **Add** methods of the **SPListCollection** class.

The following example adds a new Generic, Events, or Announcements list based on user input. A **Switch** clause is used to determine the type of list that the user specifies and sets the type of list template accordingly.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim lists As SPListCollection = mySite.Lists

Dim listTitle As String = TextBox1.Text
Dim listDescription As String = TextBox2.Text
Dim listType As String = ListBox1.SelectedItem.Text

Dim listTemplateType As New SPListTemplateType()

Select Case listType

Case "Generic List"

listTemplateType = SPListTemplateType.GenericList
Exit

Case "Events"

listTemplateType = SPListTemplateType.Events
Exit

Case "Announcements"
```

```
    listTemplateType = SPListTemplateType.Announcements
    Exit

End Select

lists.Add(listTitle, listDescription, listTemplateType)
```

[C#]

```
SPWeb mySite = SPControl.GetContextWeb(Context);
SPListCollection lists = mySite.Lists;

string listTitle = TextBox1.Text;
string listDescription = TextBox2.Text;
string listType = ListBox1.SelectedItem.Text;

SPListTemplateType listTemplateType = new SPListTemplateType();

switch(listType)
{
  case "Generic List":
  {
    listTemplateType = SPListTemplateType.GenericList;
    break;
  }

  case "Events":
  {
    listTemplateType = SPListTemplateType.Events;
    break;
  }

  case "Announcements":
  {
    listTemplateType = SPListTemplateType.Announcements;
    break;
  }
}

lists.Add(listTitle, listDescription, listTemplateType);
```

The previous example instantiates an **SPListTemplateType** object in order to contain the type of list templ specified by the user. This object must be passed as a parameter in the **Add** method. The example assumes

existence of two text boxes where the user can type a title and a description, as well as a drop-down list box that displays the list types for the user to select from.

In addition to using the **SPListTemplateType** enumeration to create a list, you can also create a list from an SPListTemplate object. The **ListTemplates** property of the **SPWeb** class can be used to return a collection of list template objects and a name indexer can be used to specify the list template to use, as in the following example, which assumes the existence of a Decision Meetings Workspace site:

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)

Dim template As SPListTemplate = mySite.ListTemplates("Decisions")
mySite.Lists.Add("My Decisions", "This is a list of decisions", template)
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);

SPListTemplate template = mySite.ListTemplates["Decisions"];
mySite.Lists.Add("My Decisions", "This is a list of decisions", template);
```

Using the **GetCustomListTemplates** method of the **SPSite** class, the next example returns the custom list templates for a specified site and creates a new list based on a specified list template:

```
[Visual Basic .NET]

Dim siteCollection As SPSite = SPControl.GetContextSite(Context)
Dim mySite As SPWeb = SPControl.GetContextWeb(Context)

Dim listTemplates As SPListTemplateCollection = siteCollection.GetCustomListTemplates(mySite)
Dim listTemplate As SPListTemplate = listTemplates("Custom List Template")
mySite.Lists.Add("Custom List", "A list created from a custom list template in the list template catalog", listTemplate)
[C#]

SPSite siteCollection = SPControl.GetContextSite(Context);
SPWeb mySite = SPControl.GetContextWeb(Context);

SPListTemplateCollection listTemplates = siteCollection.GetCustomListTemplates(mySite);
SPListTemplate listTemplate = listTemplates["Custom List Template"];
mySite.Lists.Add("Custom List", "A list created from a custom list template in the list template catalog", listTemplate);
```

To delete a list, you must specify the GUID of the list as the parameter for the **Delete** method. Use the **ID** property of the **SPList** class to find the GUID.

```
[Visual Basic .NET]
```

```
Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim lists As SPListCollection = mySite.Lists

Dim list As SPList = lists(TextBox1.Text)
Dim listGuid As System.Guid = list.ID

lists.Delete(listGuid)
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPListCollection lists = mySite.Lists;

SPList list = lists[TextBox1.Text];
System.Guid listGuid = list.ID;

lists.Delete(listGuid);
```

This example assumes the existence of a text box where the user specifies the name of the list.

3.19.7.7 Adding or Deleting a List in Multiple Web Sites

You can use one of the **Add** methods of the **SPListCollection"class** to add a list to multiple Web sites acres site collection. The following example creates a generic list on every Web site, based on the title : description that is passed to the code from two text boxes. The **AIIWebs** property of the **SPSite** class is u to return the collection of all Web sites that exist on the site.

This example assumes the existence of two text boxes on the .aspx page containing a form.

```
[Visual Basic .NET]

Dim listTitle As String = TextBox1.Text.ToString()
Dim listDescription As String = TextBox2.Text.ToString()

Dim mySite As SPSite = SPControl.GetContextSite(Context)
Dim allWebs As SPWebCollection = mySite.AllWebs
Dim web As SPWeb

For Each web In  allWebs

   Dim allLists As SPListCollection = web.Lists

   allLists.Add(listTitle, listDescription, SPListTemplateType.GenericList)

Next web
```

```
[C#]

string listTitle = TextBox1.Text.ToString();
string listDescription = TextBox2.Text.ToString();

SPSite mySite = SPControl.GetContextSite(Context);
SPWebCollection allWebs = mySite.AllWebs;

foreach (SPWeb web in allWebs)
{
  SPListCollection allLists = web.Lists;
  allLists.Add(listTitle,listDescription, SPListTemplateType.GenericList);
}
```

If you want to delete a list from every Web site in which it appears, use the **Delete** method of the **SPListCollection** class. The following example uses nested loops to drill down to a list that has a title matching the title specified in a text box.

This example assumes the existence of a text box on the .aspx page containing a form.

```
[Visual Basic .NET]

Dim mySite As SPSite = SPControl.GetContextSite(Context)
Dim allWebs As SPWebCollection = mySite.AllWebs
Dim web As SPWeb

For Each web In  allWebs

  Dim allLists As SPListCollection = web.Lists
  Dim i As Integer

  For i = 0 To allLists.Count - 1

    Dim list As SPList = allLists(i)

    If list.Title = TextBox1.Text Then

      Dim listGuid As Guid = list.ID

      allLists.Delete(listGuid)

    End If

  Next i
```

```
Next web
[C#]

SPSite mySite = SPControl.GetContextSite(Context);
SPWebCollection allWebs = mySite.AllWebs;

foreach (SPWeb web in allWebs)
{
  SPListCollection allLists = web.Lists;

  for (int i=0; i<allLists.Count; i++)
  {
    SPList list = allLists[i];

    if (list.Title == TextBox1.Text)
    {
      Guid listGuid = list.ID;
      allLists.Delete(listGuid);
    }
  }
}
```

The **Title** property of the **SPList** class is used to identify a list in the collection of lists for each Web sit, matches the specified title. The ID property returns the globally unique identifier (GUID) of the list, wh passed as the parameter for the **Delete** method.

3.19.7.8 Returning Items from a List

To return items from a list, you can instantiate an **SPWeb** object and drill down through the object me the **SPListItemCollection** object for the list. Once you return the collection of all items Sor a list, ye iterate through the collection and use indexers to return specific field values.

The following example returns all the items for a specified events list. It assumes the existence of a te for typing the name of an events list.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim listItems As SPListItemCollection = mySite.Lists(TextBox1.Text).Items
Dim i As Integer

For i = 0 To listItems.Count - 1

  Dim item As SPListItem = listItems(i)
```

```
    Response.Write(SPEncode.HtmlEncode(item("Title").ToString()) & " :: " _
      & SPEncode.HtmlEncode(item("Location").ToString()) & " :: " _
      & SPEncode.HtmlEncode(item("Begin").ToString()) & " :: " _
      & SPEncode.HtmlEncode(item("End").ToString()) & "<BR>")

Next i
```

[C#]

```
SPWeb mySite = SPControl.GetContextWeb(Context);
SPListItemCollection listItems = mySite.Lists[TextBox1.Text].Items;

for (int i=0;i<listItems.Count;i++)
{
  SPListItem item = listItems[i];

  Response.Write(SPEncode.HtmlEncode(item["Title"].ToString()) + " : " +
    SPEncode.HtmlEncode(item["Location"].ToString()) + " : " +
    SPEncode.HtmlEncode(item["Begin"].ToString()) + " : " +
    SPEncode.HtmlEncode(item["End"].ToString()) + "<BR>");
}
```

The previous example requires **using** directives **(Importsjn** Visua.I Basic) for the Mtcrosoft.Sharel'olnt, **Microsoft.SharePoint. Utilities,** and **Microsoft.SharePoint.WebCo.ntrols** namespaces.

In the example, indexers are used both to return the list typed by the user and to return specific items from the list. To return the items, the indexers specify the name of each column whose value is returned. In this case all the field names pertain to a common Events list.

You can also use one of the **GetItems** methods of the **SPList** class to return a subset of items from a list. The following example returns only Title column values where the Stock column value surpasses 100.

[Visual Basic .NET]

```
Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim list As SPList = mySite.Lists("Books")

Dim query As New SPQuery()
query.Query = "<Where><Gt><FieldRef Name='Stock'/><Value Type='Number'>100</Value></Gt></Where>"

Dim myItems As SPListItemCollection = list.GetItems(query)
Dim item As SPListItem

For Each item In myItems

  Response.Write(SPEncode.HtmlEncode(item("Title").ToString()) & "<BR>")
```

```
Next item
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPList list = mySite.Lists["Books"];

SPQuery query = new SPQuery();
query.Query = "<Where><Gt><FieldRef Name='Stock'/><Value Type='Number'>100</Value></Gt></Where>";

SPListItemCollection myItems = list.GetItems(query);

foreach (SPListItem item in myItems)
{
   Response.Write(SPEncode.HtmlEncode(item["Title"].ToString()) + "<BR>");
}
```

The previous example uses a constructor to instantiate an **SPQuery** object, and then assigns to the· Que property of the query object a string in Collaborative Application Markup Language (CAML) that specif the inner XML for the query (in other words, the **Where** element). Once the **Query** property is set, the qui object is passed through the **GetItems** method to return and display items,

The previous example requires **using** directives **(Imports** in OVisual Basic) for the **Microsoft.SharePoi Microsoft.SharePoint. Utilities,** and **Microsoft.SharePoint. WebControls** namespaces.

The example assumes the existence of a Books list that has aStock column containing number values.

3.19.7.9 Adding or Deleting List Items

To add items to a list, use the **Add** method of the **SPListItemCollection** class to create an item object, then use the **Update** method of the **SPListItem** class to update the database with the new item.

The following example assumes the existence of five text boxes, one that specifies the name of the list to to, and four others that are used to specify the values to add. Indexers are used to gather the input from all sources.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim listItems As SPListItemCollection = mySite.Lists(TextBox1.Text).Items

Dim item As SPListItem = listItems.Add()

item("Title") = TextBox2.Text
item("Stock") = Convert.ToInt32(TextBox3.Text)
item("Return Date") = Convert.ToDateTime(TextBox4.Text)
item("Employee") = TextBox5.Text
```

```
item.Update()
```

[C#]

```
SPWeb mySite = SPControl.GetContextWeb(Context);

SPListItemCollection listItems = mySite.Lists[TextBox1.Text].Items;

SPListItem item = listItems.Add();

item["Title"] = TextBox2.Text;
item["Stock"] = Convert.ToInt32(TextBox3.Text);
item["Return Date"] = Convert.ToDateTime(TextBox4.Text);
item["Employee"] = TextBox5.Text;

item.Update();
```

The example first creates an **SPListItem** object through the **Add** method of the collection. It then assign values to specific fields by using an indexer on the list item. For example, item["Title"] specifies the Title column value for the item. Finally, the example calls the **Update** method of the list item to effect changes in the database.

To create list items with preserved metadata, you can use the **Author, Editor, Created,** and **Modified** fields as indexers, where **Author or Editor** specify a Windows SharePoint Services user **ID.** For an. example, see the **SPListItem** class.

To delete items from a list, use the **Delete** method of the **SPListItemCollection** class, which takes as its parameter an index into the collection.

[Visual Basic .NET]

```
Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim listItems As SPListItemCollection = mySite.Lists(TextBox1.Text).Items
Dim itemCount As Integer = listItems.Count
Dim k As Integer

For k = 0 To itemCount - 1

  Dim item As SPListItem = listItems(k)

  If TextBox2.Text = item("Employee").ToString() Then

    listItems.Delete(k)

  End If
```

```
Next k
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);

SPListItemCollection listItems = mySite.Lists[TextBox1.Text].Items;

int itemCount = listItems.Count;

for (int k=0; k<itemCount; k++)
{
  SPListItem item = listItems[k];

  if (TextBox2.Text==item["Employee"].ToString())
  {
    listItems.Delete(k);
  }
}
```

Based on input from two text boxes, this example iterates through the collection of items for the specified H and deletes an item if the Employee field value matches the specified value.

3.19.7.10 Adding a Recurring Event to Lists on Multiple Sites

This programming task shows how to add a recurring event with a Meeting Workspace site to the Events l of every site in a collection of subsites.

1. Create a console application in Microsoft Visual Studio .NET, as described in Creating a Cons Application.

2. Add a using or Imports directive to the opening of the .cs or .vb file for the Microsoft.ShareP< and Microsoft.SharePoint.Meetings namespaces, as follows:

```
[Visual Basic .NET]

Imports Microsoft.SharePoint
Imports Microsoft.SharePoint.Meetings
[C#]

using Microsoft.SharePoint;
using Microsoft.SharePoint.Meetings;
```

3. Use the SPSite constructor to instantiate a specified site collection. This example uses an inde, the AllWebs property of the SPSite class to return a specific site, and the Webs property of the SPWeli to return the collection of subsites beneath the site. Set up a foreach loop to iterate through all the su and obtain the Events list for each site and the collection oflist items in each Events list, as follows:

```
[Visual Basic .NET]

Dim evtTitle As String = Console.ReadLine()

Dim siteCollection As New SPSite("Absolute_Url")
Dim site As SPWeb = siteCollection.AllWebs("Site_Name")
Dim subSites As SPWebCollection = site.Webs
Dim subSite As SPWeb

For Each subSite In subSites

  Dim list As SPList = subSite.Lists("Events")
Dim listItems As SPListItemCollection = list.Items
[C#]

string evtTitle = Console.ReadLine();

SPSite siteCollection = new SPSite("Absolute_Url");
SPWeb site = siteCollection.AllWebs["Site_Name"];
SPWebCollection subSites = site.Webs;

foreach (SPWeb subSite in subSites)
{
SPList list = subSite.Lists["Events"];
SPListItemCollection listItems = list.Items;
```

To create a list item, the example uses the **Add** method of the **SPListItemCollection** class to create unitialized list item, uses indexers to set various properties for the new item, and then uses the **Update** ethod to finish creating the item.

```
[Visual Basic .NET]

Dim recEvent As SPListItem = listItems.Add()

Dim recdata As String = "<recurrence><rule><firstDayOfWeek>su</firstDayOfWeek>" _
  & "<repeat><daily dayFrequency='1' /></repeat>" _
  & "<repeatInstances>5</repeatInstances></rule></recurrence>"

recEvent("Title") = evtTitle
recEvent("RecurrenceData") = recdata
recEvent("EventType") = 1
recEvent("EventDate") = New DateTime(2003, 8, 15, 8, 0, 0)
recEvent("EndDate") = New DateTime(2003, 9, 25, 9, 0, 0)
recEvent("UID") = Guid.NewGuid()
```

```
recEvent("TimeZone") = 13
recEvent("Recurrence") = -1
recEvent("XMLTZone") = "<timeZoneRule><standardBias>480</standardBias>" _
   & "<additionalDaylightBias>-60</additionalDaylightBias>" _
   & "<standardDate><transitionRule month='10' day='su' weekdayOfMonth='last' />" _
   & "<transitionTime>2:0:0</transitionTime></standardDate>" _
   & "<daylightDate><transitionRule month='4' day='su' weekdayOfMonth='first' />" _
   & "<transitionTime>2:0:0</transitionTime></daylightDate></timeZoneRule>"

recEvent.Update()
[C#]

SPListItem recEvent = listItems.Add();

string recData = "<recurrence><rule><firstDayOfWeek>su</firstDayOfWeek>" +
"<repeat><daily dayFrequency='1' /></repeat>" +
"<repeatInstances>5</repeatInstances></rule></recurrence>";
recEvent["Title"] = evtTitle;
recEvent["RecurrenceData"] = recData;
recEvent["EventType"] = 1;
recEvent["EventDate"] = new DateTime(2003,8,15,8,0,0);
recEvent["EndDate"] = new DateTime(2003,9,25,9,0,0);
recEvent["UID"] = System.Guid.NewGuid();
recEvent["TimeZone"] = 13;
recEvent["Recurrence"] = -1;
recEvent["XMLTZone"] = "<timeZoneRule><standardBias>480</standardBias>" +
"<additionalDaylightBias>-60</additionalDaylightBias>" +
"<standardDate><transitionRule month='10' day='su' weekdayOfMonth='last' />" +
"<transitionTime>2:0:0</transitionTime></standardDate>" +
"<daylightDate><transitionRule month='4' day='su' weekdayOfMonth='first' />" +
"<transitionTime>2:0:0</transitionTime></daylightDate></timeZoneRule>";
recEvent.Update();
```

The **recData** variable contains an XML fragment that specifies properties for a recurring event tak place daily for five days, and the XMLTZone indexer is used to assign time zone informatioİı for current site. The XML that defines the recurrence and that specifies the time zone informatiotı contained in the ntext3 and ntext4 columns of the UserData table in the content database.

Table 3 The following table shows examples of the different kinds ofrecurrence that can be used

jDescription	l~IE;x!am~p~le~
Every other day until a specifie end date	[illegible]

	<firstDayOfWeek>su</firstDayOfWeek> <repeat><daily dayFrequency='2' /></repeat> <windowEnd>2003-09-20T09:00:00Z</windowEnd> </rule></recurrence>
Weekly on Monday	<recurrence><rule> <firstDayOfWeek>su</firstDayOfWeek> <repeat><weekly mo='TRUE' weekFrequency='1' /></repeat> <repeatForever>FALSE</repeatForever> </rule></recurrence>
Every two months on the third day for five sessions	<recurrence><rule> <firstDayOfWeek>su</firstDayOfWeek> <repeat><monthly monthFrequency='2' day='3' /></repeat> <repeatInstances>5</repeatInstances> </rule></recurrence>
Monthly on the first Tuesday until a specified end date	<recurrence><rule> <firstDayOfWeek>su</firstDayOfWeek> <repeat> <monthlyByDay tu='TRUE' weekdayOfMonth='first' monthFrequency='1' /> </repeat> <windowEnd>2003-08-02T10:00:00Z</windowEnd> </rule></recurrence>
Yearly on the twentieth day of the ninth month until a specified end date	<recurrence><rule> <firstDayOfWeek>su</firstDayOfWeek> <repeat><yearly yearFrequency='1' month='9' day='20' /></repeat> <windowEnd>2007-09-20T07:00:00Z</windowEnd> </rule></recurrence>

5. To add a Meeting Workspace site to the recuirihg event, .use one of the **Add** methods of the **SPWebCollection** class and the **LinkWithEvent** methodöftheSPMeetingclass.

```
[Visual Basic .NET]

Dim mwsSites As SPWebCollection = subSite.Webs

Dim path As String = recEvent("Title").ToString()

Dim newSite As SPWeb = mwsSites.Add(path, "Workspace_Name", _
    "Description", Convert.ToUInt32(1033), "MPS#0", False, False)

Dim mwsSite As SPMeeting = SPMeeting.GetMeetingInformation(newSite)

Dim guid As String = list.ID.ToString()
Dim id As Integer = recEvent.ID

Try

    mwsSite.LinkWithEvent(subSite, guid, id, "WorkspaceLink", "Workspace")

Catch ex As System.Exception

    Console.WriteLine(ex.Message)

End Try

Next subSite
[C#]

SPWebCollection mwsSites = subSite.Webs;

string path = recEvent["Title"].ToString();

SPWeb newSite = mwsSites.Add(path, "Workspace_Name", "Description", 1033, "MPS#0", false, false);

SPMeeting mwsSite = SPMeeting.GetMeetingInformation(newSite);

string guid = list.ID.ToString();
int id = recEvent.ID;

try
{
mwsSite.LinkWithEvent(subSite, guid, id, "WorkspaceLink", "Workspace");
```

```
}

catch (System.Exception ex)
{
Console.WriteLine(ex.Message);
}
}
```

After the Meeting Workspace site is created, the **GetMeetingInformation** method is used to return an **SPMeeting** object representing the site.

6. Press *F5* to start the Console Application, at the command prompt type a name for the Meeting Workspace site, and then press ENTER to add a recurring event with a Meeting Workspace site to the Events list in all the subsites beneath a site.

3.19.7.11 Accessing, Copying, and Moving Files

You can use the **GetFolder** method of the **SPWeb** class to return a specified folder and then access individual files in the folder. After instantiating an **SPWeb** object (for example, as mySite), use SPFolder myFolder = mySite.GetFolder("Shared Documents") (in Visual Basic .NET, use Dim myFolder As SPFolder = mySite.GetFolder("Shared Documents")) to return the Shared Documents folder for the site. The following example returns the collection of files in the folder and displays information about the files.

```
[Visual Basic .NET]

Dim myFiles As SPFileCollection = myFolder.Files
Dim file As SPFile

For Each file In myFiles

  Response.Write(file.Url.ToString() & "<BR>")
  Response.Write(file.Length.ToString() & "<BR>")

Next file
[C#]

SPFileCollection myFiles = myFolder.Files;

foreach (SPFile file in myFiles)
{
  Response.Write(file.Url.ToString() + "<BR>");
  Response.Write(file.Length.ToString() + "<BR>");
}
```

The previous example lists the URL and size of every file within the folder.

The example requires **using** directives **(Imports** in Visual Basic) for the **Microsoft.SharePoint** namespace.

To copy files from one location to another, use one of the **CopyTo** methods of the **SPFile** class. In following example, the PageLoad event handler instantiates an **SPWeb** object for the current site cont- The **Click** event handler iterates through all the files in the folder, listing the name and size (in kilobytes: each file that surpasses a multiple of the value specified by the user in a text box, before copying the file specified folder named Archive.

```
[Visual Basic .NET]

Private mySite As SPWeb

Private Sub Page_Load(sender As Object, e As System.EventArgs)

  mySite = SPControl.GetContextWeb(Context)

End Sub 'Page_Load

Private Sub Button1_Click(sender As Object, e As System.EventArgs)

  Dim maxSize As Integer = Convert.ToInt32(TextBox1.Text)
  Dim myFolder As SPFolder = mySite.GetFolder("Shared Documents")
  Dim myFiles As SPFileCollection = myFolder.Files
  Dim file As SPFile

  For Each file In  myFiles

    If file.Length > maxSize * 1024 Then

      Response.Write(SPEncode.HtmlEncode(file.Name) & " :: " & file.Length / 1024 & "kb<BR>")
      file.CopyTo("Archive/" & file.Name, True)

    End If

  Next file

End Sub 'Button1_Click
[C#]

private SPWeb mySite;

private void Page_Load(object sender, System.EventArgs e)
{
  mySite = SPControl.GetContextWeb(Context);
}
```

```
private void Button1_Click(object sender, System.EventArgs e)
{
int maxSize = Convert.ToInt32(TextBox1.Text);

SPFolder myFolder = mySite.GetFolder("Shared Documents");
SPFileCollection myFiles = myFolder.Files;

foreach (SPFile file in myFiles)
{

  if (file.Length>(maxSize*1024))
  {
    Response.Write(SPEncode.HtmlEncode(file.Name) + ": " + file.Length/1024 + "kb<BR>");
    file.CopyTo("Archive/" + file.Name, true);
  }
}
}
```

In the example, the **CopyTô** method uses two parameters, one that specifies the destination URL for the copied file, and the other a Boolean value that Specifies whether to overwrite any file of the same name that is located at the destination.

The previous example requires **using** directives **(Impor.t~** ill Visual Basic) for the **Microsoft.SharePoint, Microsoft.SharePoint.Utilities,** and **Microsoft.SharePoint.WebControls** namespaces.

The following example moves all files from the Shared Documents list of the current site to another folder named StorageFolder, overwriting any file of the same name that may be located there.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim myFolder As SPFolder = mySite.GetFolder("Shared Documents")
Dim myFiles As SPFileCollection = myFolder.Files
Dim i As Integer

For i = myFiles.Count - 1 To 0 Step -1

  myFiles(i).MoveTo("StorageFolder/" & myFiles(i).Name, True)

Next i
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);
SPFolder myFolder = mySite.GetFolder("Shared Documents");
SPFileCollection myFiles = myFolder.Files;
```

```
for (int i = myFiles.Count - 1; i > -1; i--)
{
  myFiles[i].MoveTo("StorageFolder/" + myFiles[i].Name, true);
}
```

As the example illustrates, when collections are modified in the course of code execution by deleting or moving items, the counter for iterating through the collection must decrease in value with each iteration.

The previous example requires using directives (Imports in Visual Basic) for the Microsoft.SharePoint and Microsoft.SharePoint. WebControls namespaces.

3.19.7.12 Uploading a File to a SharePoint Site from a Local Folder

This programming task shows how to upload a file to a folder on a SharePoint site from a local folder. Thı task'uses the EnsureParentFolder method to ensure that the destination folder exists.

1. Create a Web application in Visual Studio .NET as described in Creating a Web Application on SharePoint Site, adding a FormDigest control and a page directive for Microsoft.SharePoint.WebControls namespace to the .aspx file, as follows:

```
< %@ Register Tagprefix= "Sharei'otnı" Namespace= "Microsoft.Shareroint. WebControls"
  Assembly= "Microsoft.Sharel'oint, Version= 11. 0.0. 0, Culture=neutral,
        PublicKeyToken=7 le9bcel 1le9429c" %>
```

   You can obtain the fublicKeyToken value for the current. Windows SharePoint Servic deployment from the default.aspx file in the *Loca!_prive:\Jlrogram* Files\Common Files\Micros Shared\Web Server Extensions\60\TEMPLATE\LCID(l033 in English)\STS folder, or fr information pr<>vided for the Microsoft.SharePoint assembly Local_Drive:\WINDOWSjWINNT\assemblyin Windows Explorer.

Add an Htmllnputl'ile control, a text box, and a button to theform on the .aspx page:

```
<form id= "Form] " method= "post" runat= "server">
  <SharePoint: Formbigesı runat= "server"?>
  <input id= "Filel " type= "file" runat= "server" title= "upldFileBox ">
  <asp.Button id= "Button]" runat= "server" Text= "Upload File"><lasp:Button>
  <asp: TextBox id='Textboxl " runot=t'server'c-c/asp: TextBox>
<form>
```

2. In the .aspx.cs code-behind file, add using directives for the System.IO and **Microsoft.Sharel** namespaces, as follows:

[Visual Basic .NET]

```
Imports System.JO
Imports Microsofı.Sıiarel'oinı
```

[C#]

```
using System.JO;
using Microsoft.Sharel'oint:
```

3. Add the following code to the Click event for the button:

[Visual Basic .NET]

```
If File] .PostedFile Is Nothing Then

    Return

End!/

Dim dest Ur! As String= TextBoxl. Text

Dim site As SP Web = New SPSite(destUrl).OpenWebO

DimjStream As Stream = Filel.PostedFile.InputStream
Dim contents(JStream.Length)As Byte

fStream.Read(coıtents, 0, Clnt(JStream.Length))
fStream. Closet)

EnsureParentFolder(site,destUrl)

site.Files.Addıdestllrl, contents)
[C#j

if (Filel.PostedFile == null)
        return;

string destUrl = TextBoxl. Text;

SPWeb site= new SPSite(destUr/).OpenWeb();

StreamjStream = File] .PostedFile.lnputStream;
byte[} contents= new byteffStream.Leıugth];

JStream.Read(contents,0, (int)jStream.Length);
fStream. Close();

EnsureParetıtFo/der(site, destUrl);

site.Files.Add(destUrl,contents);
```

The value typed in the text box for the destination must be an absolute URL, including the file name, which is assigned to the *destUrl* parameter.

In addition to instantiating an **SPWeb** object for the parent site, the combined use of the **SPSite** constructor and **Open Web** method validates the URL and throws an argument exception if the URL is not served by the current Windows SharePoint Services deployment. A **System.Web.UI.HtmlControls.HtmllnputFile** object is used to read the source file into a byte array for use with the **Add** method of the **SPFileCollection** class.

4. The **EnsureParentFolder** method ensures that the parent folder in the destination URL exists in the specified site, and it returns the site-relative URL of the parent folder. The **EnsureParentFolder** method accepts two parameters: an **SPWeb** object that represents the parent site, and a string that contains the absolute URL that is passed from the **UploadFile** method. If the parent folder does not exist, the **EnsureParentFolder** method creates it.

```
[Visual Basic .NET}

Public Function EnsureParentFolder(parentSiteAs SPWeb, destinUriAs String) As String

    destinUri =parentSite. GetFi/e(destinUri).Uri

    Dim index As Integer= destinUrl.Lasılndexôfi"/")
    DimparentFolderUrl As String= String.Empty
```

```
If index > - 1 Then
    parentFolderUrl = destinUr/.Substring(O, index)
    Dim parentFolder As SPFolder= parentSite.GetFolder(parentFolderUrl)
    If Not parentFolder.Exists Then
        Dim currentFolder As SPFolder= parentSite.RootFolder
        Dim folder As String
        For Each folder In parentFolderUrl.Split("!"c)
            currentFolder = currentFolder.SubFolders.Add(folder)
        Next folder
    End If
End If
Return parentFolderUrl
End Function 'EnsureParentFolder
```

[C#]

```
public string EnsuteParentFôlder(SPWeb parentSite, string destin Uri)
{
destin Uri= parentSite.GetFile(destinUrl). Uri;
    int index= destinUrl.LastindexOJ("I'?;
    string parentFolderUrl = string.Empty;
    if (index> -1)
    parentFolderUrl = destinUr/.Substring(O, index);
        SPFolder parentFolder = parentSite.GetFolder(parentFolderUrl);
        if(! parentFolder.Exists)
        SPFolder currentFolder = parentSite.RootFolder;
```

```
foreach(stringfolder  in parentFolderUrl.Split('/J)
{
currentFolder  = currentFolder.SubFolders.Add(Jolder);
}
)
}

return parentFolderUrl;
}
```

The **Getpile** method of the **SPWeb** class is used in combination with the **Uri** property of the **SPFile** class to convert the URL to a site-relative URL, throwing an exception if the specified URL is not found within the scope of the site. The URL of the parent folder is calculated by using the **String.LastlndexOf** method to determine the last appearance of a forward slash (/) within the destination URL. If there is no slash (in other words, the index equals -1), then the destination is the root folder for the site and the *parentFolderUrl* parameter returns an empty string. Otherwise, the example uses the **GetFolder** method of the **SPWeb** class to return the destination parent folder. If the folder does not exist, the example constructs the folder.

To upload a file from a local folder on the same server that is running Windows SharePoint Services, you can instead use a **System.IO.FileStream** object. In this case, add a **using** directive for the **System.IO** namespace, in addition to directives for **System** and **Microsoft.SharePoint** . The following example uses the **Click** event handler to call an **UploadFile** method, which in tum calls the previously described **EnsureParentFolder** method:

```
[Visual Basic .NET]

PubUc Sub  UploadFile(srcUrl  As String,  des/Uri  As String)

 If Not  File.Exists(srcUrl)  Then

    Throw  New  ArgumentException(String.Format("{O)     does  not  exist",  srcllrl),  "srcllrl")

 End If

 Dim  site  As  SP Web=  New  SPSite(destUrl).OpenWeb()

 Dim [Stream  As  FileStream  = File. OpenRead(srcUrl)
 Dim  contents(!Stream.Length)  As  Byte
[Stream.Readıcontenıs, 0, Clnt(!Stream.Lengtlı))
 fStream.Close()

 EnsureParentFolder(site,  destUrl)

stte.Files.Addıdesıilrl, contents)
```

```
End Sub 'Up/oadFile
[C#]

public void UploadFile(string srcUrl, string destUr/)
{
  if(! File.Exists(srcUrl))
  {
    throw new ArgumentException(String.Format("{O}does not exist", srcUrl), "srcilrl");
  }

  SPWeb site= new St'Sitetdestllrlt.Opeuwebi);

  FileStreamjStream = File.OpenRead(srcUrl);
  byte[] contents = new byte[!Stream.Length];
  jStream.Read(contents, 0, (int)jStream.Length);
  jStream.Close();
  Ensurerarentlioldertsue, destUrl);
  site.Files.Addıdestllrl, contents);
}
```

The UploadFile method accepts two parameters. The *srcUrl* parameter specifies the path of the sour location in the file system of the local computer, and the *dest Ur!* parameter specifies the absolute URL of t destination. A System.10.FileStream object is used to read the source file into a byte array for use with t Add method of the SPFileCollection class.

3.19.7.13 Handling Document Library Events

Microsoft Windows SharePoint Services 2.0 provides document library events that enable developers to bu upon the SharePoint platform. Developers can create managed code assemblies that define handlers for th events and then bind the handlers to document libraries. The event handlers can call into the SharePoint obj model to access the configuration and content databases directly, or they can invoke external services, us Windows SharePoint Services as a user interface for other systems.

Table 3.4 The following table describes the events for document libraries that Windows SharePoint Servi provides.

!Event	liDescription
jcancel Check Out	!IChangesmade to a checked-outdocument are undone.
!Check In	l!A document is checked in to the library.
jCheck Out	l[A document is checked out from the library.
jcopy	j[A document in the library is copied.
!Delete	liA document is deleted from the library.
jınsert	IIA new document is saved to the library.
jMove or Rename	l!A document is moved or renamed.
jUpdate	JJAn existing document or the value of a custom column in the library is edited.

In the context of Windows SharePoint Services, an event handler is a .NET class that implements the IListEventSink interface, whose one method, OnEvent, is used within the handler. The SPListEvent object contains information about an event that occurs, and you can return the type of event through the Type property. Use the Site property to access the object model of the Microsoft.SharePoint namespace within the handler.

You must install the managed assembly that defines an event handler in the Global Assembly Cache (GAC). In a server farm configuration, you must install this managed assembly in the GAC of each front-end Web server.

J'o deploy a,r,, event handler on a server, event handling must be enabled on the Virtual Server General Settings page in SharePoint Central Administration.

Only users with full permissions for a document library can add event handlers.

3.19.7.13.1 Event settings

The list metadata for a document library binds the class of an event handler to a library by means of the following properties. The metadata can be set on the Document Library Advanced Settings page for the document library through code that implements the EventSinkAssembly, EventSinkClass, and EventSinkData properties of the SPList class, or through definitions contained in front-end site templates.

Table 3.5 The following table summarizes the event settings.

Setting	Type	Description
Assembly Name	String	The strong name of the event handler assembly file that lives in the GAC.
Class Name	String	The fully qualified, case-sensitive name of the class within the assembly.
Properties	St . rıng	An arbitrary string of custom properties for use by the event handler whose length cannot exceed 255 characters.

After you install the event handler assembly, on the Document Library Advanced Settings page for the document library, specify the strong name of the assembly in the Assembly Name box, which must be in the format *Assembly _Name,* Version=*Version, Culture=Culture,* PublicKeyToken=Public*_Key_Token.* You can obtain these values by browsing to %windir%\assembly in Win~o\>/s Explorer.As an example, the strong name for the Windows SharePoint Services asserribly looks like Microsoft.SharePoint, Version=l 1.0.0.0, Culture=Neutral,PublicKeyToken=7le9bcel l le9429c.

The value that you specify in the Class Name box must be the full, case-sensitive name of a class defined in the specified assembly that implements the IListEventSink interface. For example, the full name of the class in the following example would be myNamespace.myClass.

```
[Visual Basic .NET]

Namespace myNamespace

  Public Class myClass

    Inherits Microsoft.SharePoint.IListEventSink
```

```
    Public Sub OnEvent(evt As Microsoft.SharePoint.SPListEvent)

        'Add an announcement
        AddAnnouncement(evt)

    End Sub 'OnEvent

    Public Sub AddAnnouncement(evt As Microsoft.SharePoint.SPListEvent)
        .
        .
        .
    End Sub 'AddAnnouncement

End Class 'myClass

End Namespace 'myNamespace
```

[C#]

```
namespace myNamespace
{
  public class myClass : Microsoft.SharePoint.IListEventSink
  {
    public void OnEvent(Microsoft.SharePoint.SPListEvent evt)
    {
      //Add an announcement
      AddAnnouncement(evt);
    }

    public void AddAnnouncement(Microsoft.SharePoint.SPListEvent evt)
    {
      .
      .
      .
    }
  }
}
```

Typing a value in the **Properties** box is optional. This value can contain up to 255 characters for use by t event handler.

3.19.7.13.2 Events in relation to other events

An event handler running in the context of Windows SharePoint Services cannot generate other events. To program side effects for an event, you must provide the code within the event handler.

The event handler is called asynchronously from the actual request

A new instance of an event handler class is created for each event. Consequently, you cannot store state in your event handler class and expect it to persist across multiple events. You can, however, create a base class for an event handler or for multiple event handlers that caches state so that the same sink object can be used for all events on a document library.

3.19.7.13.3 Caching credentials

Event handlers run in the context of the Internet Information Services (IIS) J\pplication Pool Identity, an account that typically has no permissions in Windows SharePoint Services. If your handler needs to interact with Windows SharePoint Services through the object model, you need to .establish and cache credentials through impersonation of a user. For information, see Impersonating and Reverting.

3.19.7.13.4 Security context

The event handler thread runs within the saıne application pool account as the SharePoint virtual server. If, however, the event handler needs to be tun in a different account in order to manipulate Windows SharePoint Services or an external service, the developer must maintain his or her own user credentials and manage his or her own security context.

3.19.7.13.5 Exceptions and errors

Exceptions for event handlers used in Windows SharePoint Services are thrown in the following conditions, for which a Microsoft Windows NT event log entry is created:

The assembly for the handler cannot be loaded.
The class defining the handler is not found.
The class does not implement the IListEventSink interface.
The OnEvent method throws an exception.

When you restart IIS, each IIS worker process terminates only after all its queued work items, including HTTP requests and document library events, complete their tasks.

3.19.7.13.6 Definitions and templates

Developers can define event handler assemblies within list definitions, which provides a useful means for deploying solutions such as workflow-enabled document libraries in which events are already bound to the appropriate handlers. An event handler can be bound to a document library through the EventSinkA~'sembly, EventSinkCiass, and EventSinkData attributes of the List element in the SCHEMA.XML file for the list type. When an end user creates a library through the UI from a list definition that specifies an event handler, list event settings are automatically set to the values specified in the definition. When an end user saves a document library as a list template, event settings specified through the UI are preserved in the template.

For information about definitions and templates for sites or lists, see Introdüctiori tö Templates and Definitions.

3.19.7.13.7 Event handler example

The following programming task creates an event handler that deletes the oldest version of a file in a document library when the number of versions for the fiie reaches a specified number. To establish credentials for the handler, a method is used that returns a token to the event handler for impersonation of a user.

1. On the File menu in Visual Studio .NET, point to New and then click Project.
2. In the New Project dialog box, select Visual Basic Projects or Visual C# Projects in the Project Types box.
3. In the Templates box, select Class Library.
4. In the Name box, type the name of the project, which in this task is "DocLibRandler".
5. In the Location box, type the path to where to create the project, and then click OK.
6. In Solution Explorer, right-click the References node, and click Add Reference on the menu.
7. On the .NET tab of the Add Reference dialog box, select Windows SharePoint Services in of components, click Select, and then click OK.
8. Add the following directives to the Classl .cs or Classl .vb file of the DocLibHandler project System directive that is included by default.

```
[Visual Basic .NET]

Imports System.Security.Principal
Imports System.Runtime.InteropServices
Imports Microsoft.SharePoint
[C#]

using System.Security.Principal;
using System.Runtime.InteropServices;
using Microsoft.SharePoint;
```

1. Change the namespace and class name in the Classl .cs or Class1.vb file to be DocLibEventij and DocVersionHandler, as follows.

```
[Visual Basic .NET]

Namespace DocLibEventHandler

  Public Class DocVersionHandler
Implements IListEventSink
[C#]

namespace DocLibEventHandler
```

```
{
public class DocVersionHandler : IListEventSink
{
```

2. Add the OnEvent method of the IListEventSink interface to the DocVersionHandler class, as follows.

```
[Visual Basic .NET]

Private maxVersions As Integer = 5

Sub IListEventSink.OnEvent(listEvent As Microsoft.SharePoint.SPListEvent)

  ' TODO: ***** PROVIDE YOUR IMPLEMENTATION HERE ****
  ' Get the credentials (User_Alias, Domain, Password) that this event sink instance should
  ' run as and initialize the wic object by calling the CreateIdentity method

  Dim wic As WindowsImpersonationContext = CreateIdentity(User_Alias, Domain, Password).Impersonate()

  If listEvent.Type = SPListEventType.Update OrElse listEvent.Type = SPListEventType.CheckIn Then

    Dim site As SPWeb = listEvent.Site.OpenWeb()
    Dim file As SPFile = site.GetFile(listEvent.UrlAfter)
    Dim nDocVersions As Integer = file.Versions.Count

    While maxVersions > 0 AndAlso nDocVersions > maxVersions - 1

      file.Versions.Delete(0)
      nDocVersions = file.Versions.Count

    End While

  End If

  wic.Undo()

End Sub 'IListEventSink.OnEvent
[C#]

private int maxVersions = 5;

void IListEventSink.OnEvent(Microsoft.SharePoint.SPListEvent listEvent)
{
/* TODO: ***** PROVIDE YOUR IMPLEMENTATION HERE ****
```

```
Get the credentials (User_Alias, Domain, Password) that this event sink instance should run as
and initialize the wic object by calling the CreateIdentity method*/

WindowsImpersonationContext wic = CreateIdentity(User_Alias, Domain, Password).Impersonate();

if ((listEvent.Type == SPListEventType.Update) ||
(listEvent.Type == SPListEventType.CheckIn))
{
SPWeb site = listEvent.Site.OpenWeb();
SPFile file = site.GetFile(listEvent.UrlAfter);
int nDocVersions = file.Versions.Count;

while ((maxVersions > 0) && (nDocVersions > (maxVersions - 1)))
{
        file.Versions.Delete(0);
        nDocVersions = file.Versions.Count;
}
}

wic.Undo();
}
```

3. The **OnEvent** method calls the following method to impersonate an account with adminisı permissions.

It is not recommended that You pass credentials in clear-text, because this presents a security ri impersonate a user more securely, you can return the information programmatically with fur provided by the operating system for getting secret data from the user, such as the Cryptl'rotectüs **CryptUnprotectData** functions of the Data Protection API (DPAPI).

Include the following method within the **DocVersionHandler** class.

```
[Visual Basic .NET]

Protected Shared Function CreateIdentity(User As String, Domain As String, Password As String) As WindowsIdentity

  ' The Windows NT user token.
  Dim tokenHandle As New IntPtr(0)

  Const LOGON32_PROVIDER_DEFAULT As Integer = 0
  Const LOGON32_LOGON_NETWORK As Integer = 3

  tokenHandle = IntPtr.Zero

  ' Call LogonUser to obtain a handle to an access token.
```

```
  Dim returnValue As Boolean = LogonUser(User, Domain, Password, LOGON32_LOGON_NETWORK,
LOGON32_PROVIDER_DEFAULT, tokenHandle)

  If False = returnValue Then

    Dim ret As Integer = Marshal.GetLastWin32Error()
    Throw New Exception("LogonUser failed with error code: " + ret)

  End If

  System.Diagnostics.Debug.WriteLine(("Created user token: " + tokenHandle))

  'The WindowsIdentity class makes a new copy of the token.
  'It also handles calling CloseHandle for the copy.
  Dim id As New WindowsIdentity(tokenHandle)
  CloseHandle(tokenHandle)
  Return id

End Function 'CreateIdentity
<DllImport("advapi32.dll", SetLastError := True)> _
  Private Shared Function LogonUser(lpszUsername As String, lpszDomain As String, _
    lpszPassword As String, dwLogonType As Integer, dwLogonProvider As Integer, _
    ByRef phToken As IntPtr) As Boolean
-----------------------------------------------------------------------------
<DllImport("kernel32.dll", CharSet := CharSet.Auto)> _
  Private Shared Declare Auto Function CloseHandle Lib "kernel32.dll" (handle As IntPtr) As Boolean
[C#]

protected static WindowsIdentity CreateIdentity(string User, string Domain, string Password)
{
  // The Windows NT user token.
  IntPtr tokenHandle = new IntPtr(0);

  const int LOGON32_PROVIDER_DEFAULT = 0;
  const int LOGON32_LOGON_NETWORK = 3;

  tokenHandle = IntPtr.Zero;

  // Call LogonUser to obtain a handle to an access token.
  bool returnValue = LogonUser(User, Domain, Password,
  LOGON32_LOGON_NETWORK, LOGON32_PROVIDER_DEFAULT,
  ref tokenHandle);
```

```
    if (false == returnValue)
    {
        int ret = Marshal.GetLastWin32Error();
        throw new Exception("LogonUser failed with error code: " + ret);
    }

    System.Diagnostics.Debug.WriteLine("Created user token: " + tokenHandle);

    //The WindowsIdentity class makes a new copy of the token.
    //It also handles calling CloseHandle for the copy.
    WindowsIdentity id = new WindowsIdentity(tokenHandle);
    CloseHandle(tokenHandle);
    return id;
}

[DllImport("advapi32.dll", SetLastError=true)]
private static extern bool LogonUser(String lpszUsername, String lpszDomain, String lpszPassword,
int dwLogonType, int dwLogonProvider, ref IntPtr phToken);

[DllImport("kernel32.dll", CharSet=CharSet.Auto)]
private extern static bool CloseHandle(IntPtr handle);
```

4. To create a strong name for the assembly, go to the *Local_Drive:\J>rögram* Files\Microsoft Vi Studio .NET 2003\SDK\vl.1\Bin directory *(Local_Drive:\Program* Files\Microsoft Visual St .NET\FrameworkSDK\Binin Visual Studio .NET Version 7) through a command prompt, and type s Local_Drive:\DocLibHandler.snk,which creates a key file for-theassembly.

5. Add the following attributes for the assembly version and key file to the Assemblylnfo file o project, replacing the attributes that are provided by default.

```
[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyKeyFile("Local_Drive:\\DocLibHandler.snk")]
```

6. Click **Build Solution** on the **Build** menu.

7. To install your solution in the GAC, drag the dynamic-link library (DLL) of your assembly *Local_Drive:\WINDOWS\assembly* directory in Windows Explorer.

8. On the Virtual Server General Settings page in SharePoint Central Administration, select Or Event Handlers section to enable use of event handlers on the virtual server.

9. From a view of the document library to which you want to attach the event handler assembl Modify settings **and** columns, and then click Change advanced settings on the page for customir document library.

10. On the Document Library Advanced Settings page, type the strong name for the assembl Assembly Name box. For example, DocLibHandler, Version=I.0.0.0, Culture= PublicKeyToken=Ofafac2aOcc92888You can get the values used in the strong name by right-clic DLL of your assembly in the *Local_Drive:\WINDOWS\assembly* directory and then clicking Prop, the shortcut menu.

11. Type the fully qualified, case-sensitive name of the class in the Class Name box, which in this example is DocLibEventHandler.DocVersionHandler.

12. Reset IIS by typing iisreset at a command prompt.

To implement the handler in a document library, versioning must be enabled on the Document Library Settings page for the library.

3.19.7.14 Adding or Removing Users

You can add users to a site group by using one of the AddUser methods of the SPRole class.

For example, after instantiating an SPWeb object, return the Administrator group for the site as follows: SPRole admins = web.Roles["Administrator"] (in Visual Basic .NET, use Dim admins As SPRole = web.Roles("Administrator")).Next, call the AddUser method on the SPRole object as follows:

```
[Visual Basic .NET]

admins.AddUser("Domain_Name\Someone", "someone@example.com", "Someone", "Public relations.")
[C#]

admins.AddUser("Domain_Name\\Someone","someone@example.com","Someone","Public relations.");
```

In this example, the logon name, e-mail address, user'name, and notes are specified for the AddUser method.

In the following example, user input is gathered from)f()~Ltext boxes to supply parameter values for the AddUser method. If successful, a message is displayedthrough a label; otherwise, a try catch block traps the exception, whose message is displayed in the label.

The example assumes the existence of a .aspx page thatcontains a label control.

```
[Visual Basic .NET]

Dim mySite As SPWeb = SPControl.GetContextWeb(Context)

Dim userName As String = TextBox1.Text
Dim displayName As String = TextBox2.Text
Dim email As String = TextBox3.Text
Dim notes As String = TextBox4.Text
Dim admins As SPRole = mySite.Roles("Administrator")

Try

  admins.AddUser(userName, email, displayName, notes)
  Label1.Text = "Successfully added user."

Catch ex As Exception
```

```
    Label1.Text = SPEncode.HtmlEncode(ex.ToString())

End Try
```

```
[C#]

SPWeb mySite = SPControl.GetContextWeb(Context);

string userName = TextBox1.Text;
string displayName = TextBox2.Text;
string email = TextBox3.Text;
string notes = TextBox4.Text;

SPRole admins = mySite.Roles["Administrator"];

try
{
   admins.AddUser(userName,email,displayName,notes);
   Label1.Text = "Successfully added user.";
}

catch(Exception ex)
{
   Label1.Text = SPEncode.HtmlEncode(ex.ToString());
}
```

The previous example requires using directives (Imports in Visual Basic) for the Microsoft.SharePoint, Microsoft.SharePoint. Utilities, and Microsoft.SharePoint. WebControls namespaces.

To remove a user from a site group, use the RemoveUser method of the SPRole class. The following example removes the user with the logon name that is typed in a text box and displays the number of users before and after the user is removed. An indexer is used with the Users property of SPRöle in order to create a user object for the specified user. For information about indexers, see Programming with the Microsoft.SharePoint and Microsoft.SharePoint.Administration Namespaces.

```
[Visual Basic .NET]

Dim userName As String = TextBox1.Text
Dim mySite As SPWeb = SPControl.GetContextWeb(Context)
Dim adminRole As SPRole = mySite.Roles("Administrator")
Dim userRemove As SPUser = adminRole.Users(userName)

Label1.Text = adminRole.Users.Count.ToString()

adminRole.RemoveUser(userRemove)
```

```
Label2.Text = adminRole.Users.Count.ToString()
```

```
[C#]

string userName = TextBox1.Text;

SPWeb mySite = SPControl.GetContextWeb(Context);
SPRole adminRole = mySite.Roles["Administrator"];
SPUser userRemove = adminRole.Users[userName];

Label1.Text = adminRole.Users.Count.ToString();
adminRole.RemoveUser(userRemove);
Label2.Text = adminRole.Users.Count.ToString();
```

Use one of the **AddUser** methods or the **Remove User** method of the **SPGroup** class to add or remove users similarly in a cross-site group. You can also use the **AddCollection** method of the **SPPermissionCollection** class to add users, site groups, and cross-site groups to the set of permissions for a list. In addition, you can use the **AddCollection** method of the **SPUserCollection** class to add users to the collection of users in a site, site group, or cross-site group.

The previous example requires **using** directives **(Imports** in Visual Basic) for the **Microsoft.SharePoint** and **Microsoft.SharePoint. WebControls** namespaces.

The example assumes the existence of a .aspx page that contains two label controls.

Microsoft SharePoint Products and Technologies no longer rely on role-based security for assigning rights and permissions to users. Instead, SharePoint Products and Technologies use site groups and cross-site groups to assign rights and permissions to users. Site groups are custom security groups that apply to a specific Web site. Cross-site groups are custom security groups that apply to more than one Web site. For more information, see Microsoft Windows SharePoint Services Help.

.SharePoint,

he following
nber of users
rder to create
ng with the

3.19.7.15 Returning Central Administrative Properties

Use classes in the **Microsoft.SharePoint.AdministraHon** namespace to access global administrative property settings for a deployment of Microsoft Windows SharePoint Services. The top-level class, **SPGlobalAdmin,** provides accessor properties to various supplementary classes, such as **SPVirtualServer, SPGlobalConfig,** and **SPUsageSettings.** Together, these classes provide access to global administrative settings that are maintained within the configuration database. This programming task shows how to create a custom report of administrative settings and usage.

Use the **SPGlobalAdmin** constructor to create an **SPGlobalAdmin** object, and then use its properties to return instances of the other classes. The following example shows how to create a global administration object and then use it to return various property settings:

```
[Visual Basic .NET]

Dim globAdmin As New SPGlobalAdmin()

Dim mailFrom As String = globAdmin.MailFromAddress
Dim mailTo As String = globAdmin.MailReplyToAddress
```

```
Dim mailCodePg As String = globAdmin.MailCodePage.ToString()

Dim apPools As System.Collections.IDictionary = globAdmin.ApplicationPools
Dim myEntry As DictionaryEntry

For Each myEntry In  apPools

  applicationPools += myEntry.Key + "  " + myEntry.Value + "<BR>"

Next myEntry
[C#]

SPGlobalAdmin globAdmin = new SPGlobalAdmin();

string mailFrom = globAdmin.MailFromAddress;
string mailTo = globAdmin.MailReplyToAddress;
string mailCodePg = globAdmin.MailCodePage.ToString();
System.Collections.IDictionary apPools = globAdmin.ApplicationPools;

foreach (DictionaryEntry myEntry in apPools)
{
  applicationPools += myEntry.Key + "  " + myEntry.Value + "<BR>";
}
```

The example iterates through the collection of dictionary key-and-value pairs returned. **ApplicationPools** property, which represent the name and type of each application pool currently bei in Internet Information Services (IIS).

In the next example, the **Config** property of the **SPGlobalAdmin** class returns an **SPGlobalConfiı** which in tum gets the name of the SharePoint Administration group, as well as a list of the file t) cannot be saved or retrieved on the server:

```
[Visual Basic .NET]

Dim globConfig As SPGlobalConfig = globAdmin.Config
Dim admGroup As String = globConfig.AdminGroup
Dim blkdFileTypes As System.Collections.Specialized.StringCollection = globConfig.BlockedFileTypes
Dim i As Integer

For i = 0 To blkdFileTypes.Count - 1

  blockedTypes += blkdFileTypes(i) + "<BR>"

Next i
[C#]
```

```
SPGlobalConfig globConfig = globAdmin.Config;
string admGroup = globConfig.AdminGroup;
System.Collections.Specialized.StringCollection blkdFileTypes = globConfig.BlockedFileTypes;

for (int i=0; i<blkdFileTypes.Count; i++)
{
   blockedTypes += blkdFileTypes[i] + "<BR>";
}
```

As in the above example, use the **BlockedFileTypes** property to return a string collection of the blocked file extensions, and then use an indexer to iterate through the collection and concatenate the strings into a single string.

Use the **UsageSettings** property of the **SPGiobalAdmin** class to return an **SPUsageSettings** object, whose properties define usage analysis settings. In this example, the object is used to report the directory where log files are stored, the number oflog files created, and whether logging is enabled, as well as the processing start and end times and whether usage processing is enabled.

```
[Visual Basic .NET]

Dim useSettings As SPUsageSettings = globAdmin.UsageSettings
Dim logFilesDir As String = useSettings.LogFilesDirectory
Dim numberLogFiles As String = useSettings.NumberLogFiles.ToString()

If useSettings.LoggingEnabled = True Then

   logEnab = "Enabled"

Else

   logEnab = "Not enabled"

End If

Dim processEnd As String = useSettings.ProcessingEndTime.ToString()
Dim processStart As String = useSettings.ProcessingStartTime.ToString()

If useSettings.UsageProcessingEnabled = True Then

   useProcEnab = "Enabled"

Else

   useProcEnab = "Not enabled"
```

```
End If
[C#]

SPUsageSettings useSettings = globAdmin.UsageSettings;

string logFilesDir = useSettings.LogFilesDirectory;
string numberLogFiles = useSettings.NumberLogFiles.ToString();

if (useSettings.LoggingEnabled == true)
{
  logEnab = "Enabled";
}
else
{
  logEnab = "Not enabled";
}

string processEnd = useSettings.ProcessingEndTime.ToString();
string processStart = useSettings.ProcessingStartTime.ToString();

if (useSettings.UsageProcessingEnabled == true)
{
  useProcEnab = "Enabled";
}
  else
{
  useProcEnab = "Not enabled";
}
```

The **VirtualServers** property of the **SPGlobalAdmin** class returns the collection of all the virtual ser server running Windows SharePoint Services. In addition to reporting several administrative sett following example iterates through the collection of virtual servers and prints the version numb installation if the server state (represented by an **SPVirtualServerState** value) is **Ready.**

```
[Visual Basic .NET]

Dim serverVersion As String
Dim virtServers As String
Dim vServers As SPVirtualServerCollection = globAdmin.VirtualServers
Dim j As Integer

For j = 0 To vServers.Count - 1
```

```
If vServers(j).State.ToString().Equals("Ready") Then

  serverVersion = vServers(j).Version.ToString()

Else

  serverVersion = "N/A"

End If

virtServers += "Application Pool ID: " _
  & vServers(j).ApplicationPoolId.ToString() _
  & "<BR>Description: " & vServers(j).Description.ToString() _
  & "<BR>Host Name: " & vServers(j).HostName.ToString() _
  & "<BR>IIS Instance ID: " & vServers(j).IISInstanceId.ToString() _
  & "<BR>Port Number: " & vServers(j).Port.ToString() _
  & "<BR>State: " & vServers(j).State.ToString() & "<BR>URL: " _
  & vServers(j).Url.ToString() & "<BR>Version: " & serverVersion _
  & "<BR><BR>"

Next j
[C#]

string serverVersion, virtServers = "";

SPVirtualServerCollection vServers = globAdmin.VirtualServers;

for (int j = 0;j<vServers.Count;j++)
{
  if (vServers[j].State.ToString().Equals("Ready"))
  {
    serverVersion = vServers[j].Version.ToString();
  }
  else
  {
    serverVersion = "N/A";
  }

  virtServers += "Application Pool ID: " + vServers[j].ApplicationPoolId.ToString() +
  "<BR>Description: " + vServers[j].Description.ToString() +
  "<BR>Host Name: " + vServers[j].HostName.ToString() +
  "<BR>IIS Instance ID: " + vServers[j].IISInstanceId.ToString() +
  "<BR>Port Number: " + vServers[j].Port.ToString() +
```

```
  "<BR>State: " + vServers[j].State.ToString() +
  "<BR>URL: " + vServers[j].Url.ToString() +
  "<BR>Version: " + serverVersion + "<BR><BR>";
}
```

3.19.7.16 Setting the Administrative Port Identity

When a configuration database is created, the current application pool identity for SharePoint Ce Administration is stored in the configuration database in the format *DOMAJMAccount_Name.* When a computer is connected to the configuration database, Microsoft Windows SharePoint Services 2.0 verifies the new application pool identity matches the identity stored in the configuration database. If the identitii not match, the operation to connect the new computer does not succeed.

To connect a new computer to the configuration database after changing the application pool .identit- identity that is stored in the configuration database must be set to the newjdentity. The following example shows how to set the application pool identity by using the **Properties** property o **SPGiobalConfig** class.

```
[Visual Basic .NET]

Dim spGlobalAdmin As New SPGlobalAdmin()

spGlobalAdmin.Config.Properties("AdminPortIdentity") = "Identity"

spGlobalAdmin.Config.Properties.Update()

spGlobalAdmin.Close()
[C#]

SPGlobalAdmin spGlobalAdmin = new SPGlobalAdmin();

spGlobalAdmin.Config.Properties["AdminPortIdentity"] = "Identity";

spGlobalAdmin.Config.Properties.Update();

spGlobalAdmin.Close();
```

The global string **AdminPortIdentity** is used as an indexer on, the **Properties** property, which r **SPPropertyBag** object representing properties of the configuration database. The **Update** methoc used for changes to take effect in the database, and the **Close** method is used to release the consumedby the **SPGlobalAdmin** object.

In the context of a server farm, create separate instances of the **SPGlobalAdmin** class to n administrativeport identity and to refresh the configuration cache, as in the following example.

```
[Visual Basic .NET]

Dim globalAdmin1 As SPGlobalAdmin = Nothing
```

```
Dim globalAdmin2 As SPGlobalAdmin = Nothing

Try

  globalAdmin1 = New SPGlobalAdmin()
  globalAdmin1.RefreshConfigCache()

  globalAdmin2 = New SPGlobalAdmin()
  globalAdmin2.Config.Properties("AdminPortIdentity") = "Identity"
  globalAdmin2.Config.Properties.Update()

Finally

  If Nothing <> globalAdmin1 Then

    globalAdmin1.Close()

  End If

  If Nothing <> globalAdmin2 Then

    globalAdmin2.Close()

  End If

End Try
```

[C#]

```
SPGlobalAdmin globalAdmin1 = null;
SPGlobalAdmin globalAdmin2 = null;

try
{
  globalAdmin1 = new SPGlobalAdmin();
  globalAdmin1.RefreshConfigCache();

  globalAdmin2 = new SPGlobalAdmin();
  globalAdmin2.Config.Properties["AdminPortIdentity"] = "Identity";
  globalAdmin2.Config.Properties.Update();
}

finally
{
```

```
if (null != globalAdmin1)
{
    globalAdmin1.Close();
}

if (null != globalAdmin2)
{
    globalAdmin2.Close();
}
}
```

3.19.8 Customizing Templates for Microsoft Windows SharePoint Services

This section contains the following programming tasks:

- Creating a Site Definition from an Existing Site Definition
- Using Configurations
- Using Modules to Add Files to a Site Definition
- Customizing the Navigation Areas
- Customizing Themes
- Customizing the Logos for SharePoint Sites
- Adding a Document Type and File Type Icon
- Creating a List Definition
- Adding a Field to a List Definition
- Customizing the Toolbar for a List
- Customizing the Shortcut Menu for List Items
- Customizing the Message Text for Alerts
- Working with web.config Files
- Working with Form Libraries
- Extending Help

3.19.8.1 Creating a Site Definition from an Existing Site Definition

You can create a site definition by copying and modifying an existing site definition. This task involves use Collaborative Application Markup Language (CAML) in two schema files: one Jhaf is a copy of WEBTEMP.XML file, and the other a copy of an ONET.XML file. It is recommended that you create as definition as described in this topic rather than modifying the originally installed WEBTEMP.XML file.

Copy the existing site definition folder located in the *Local_Prive:\Program* Files\Common Files'Microsı Shared\Web Server Extensions\60\Template\1033directory.

For example, to create a custom site definition that derives from the site definition for MicrosoftWindo: SharePoint Services, copy the STS folder located in the 1033 directory. Name the new folder using all capi letters.

Make a copy of the WEBTEMP.XML file located at *Local_Drive:\Program* Files\Common Files\Microsı Shared\Web Server Extensions\60\TEMPLATE\l 033\XML.

Give the file a unique name by appending a string to the, name of-the original file; for example, WEBTEMPACTION.XML. At run time, the compiler merges information contained in this file with the information contained in the original file in order to specify which site templates are available for creating new sites.

Customize the contents of the new WEBTEMP file.

Each WEBTEMP.XML file contains a collection of **Template** elements. and **Configuration** subelements, which identify to the compiler all the site definitions that can be instantiated. The **Configuration** element defines a title, a description, and a URL for the image displayed in the user interface, properties common to each Web site created using the site definition.

In each **Template** element defined in the WEBTEMP file, the **Name** attribute must contain the same name, in all capital letters, that is assigned to the new folder. Also..in order to avoid conflict with IDs already used in Windows SharePoint Services, use unique values greater than 10,000 for the ID attribute.

The following example defines a single site definition. The example assumes the existence of an ACTIONCOMMITTEE directory that has been created as previously described.

```
<?xml version="1.0" encoding="utf-8" ?>
<Templates xmlns:ows="Microsoft SharePoint">
  <Template Name="ACTIONCOMMITTEE" ID="98">
    <Configuration ID="0" Title="Action Committee Team Site" Type="0" Hidden="FALSE"
      ImageUrl="images/stsprev.jpg" Description="This template provides a forum
      for the team to create, organize, and share information quickly and easily. It includes a
      Document Library, and basic lists such as Announcements, Events, Contacts, and Quick Links.">
    </Configuration>
  </Template>
</Templates>
```

Save the file.

You may need to reset Internet Information Services (IIS) forthe' new template to appear as an option on the Template Selection page.

3.19.8.2 Using Configurations

If you are creating a site definition, configurations allow yoıto specify which lists to include in the creation of a site. Through configurations, you can reuse existing .list definitions in the ONET.XML file for a given site definition, which prevents you from having to copy .or recode list definitions. You can create multiple configurations in one ONET.XML file, each one enabling the creation of a site with a different set of lists.

It is recommended that you create a custom site definition by copying an existing site definition, rather than modifying the original files installed with Windows SharePoint Services. Changes that you make to originally installed files may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version. For information on creating a custom site definition, see Creating a Site Definition from an Existing Definition.

To set up a configuration within a site definition, use **Configuration** elements within a -custom WEBTEMP*.XML file and within one or more ONET.XML files that are referenced in the WEBTEMP file.

For information on how to create a custom WEBTEMP*.XML file, see Creating a Site Definition from an Existing Definition.

For example, the following WEBTEMP*.XML file uses two configurations to define different configurations for instantiating a site, one for a Research Collaboration site and the other for a Research Documen1 Workspace site. This example involves use of only two configurations within a single site definition, but yor can include multiple definitions with different configurations within a single WEBTEMP* .XML file, eacl one referencing different site definition directories and their respective ONET.XML files.

```
<?xml version="1.0" encoding="utf-8" ?>
<Templates xmlns:ows="Microsoft SharePoint">
  <Template Name="RESEARCH" ID="95">
    <Configuration ID="0" Title="Research Collaboration site" Type="0" Hidden="FALSE"
      ImageUrl="images/stsprev.jpg" Description="This definition creates a site for the Research
      team to create, organize, and share general information.">
    </Configuration>
    <Configuration ID="1" Title="Research Workspace" Type="0" Hidden="FALSE"
      ImageUrl="images/dwsprev.jpg" Description="This definition creates a site for Research
      team colleagues to work together on specific documents. ">
    </Configuration>
  </Template>
</Templates>
```

As indicated by the value of the Name attribute in the Template element, thı.s e1<ample assumes that a definition directory called "RESEARCH" exists. If a WEBTEMP*.XML file specifies more than one definition, the definitions are distinguished by their unique ID values.

Each Configuration element also contains an ID attribute, which references a specific configuration def in the ONET.XML file. The combination of this ID and the value of the Name attribute in the Temp element provides a reference to the contents of a specific Configuration element in a specific ONET.:X file. In the example, the Name attribute contains RESEARCH and the ID attributes contain 0 and' 1, W reference the RESEARCH site definition and configurations with IDs of 0 or 1 in ONET.XML.

In the ONET.XML file, each configuration defines a specific type of site that can be created from the definition. All configurations within this file share a set of available list definitions, document temp navigation areas, base list types, and modules that are defined within the file. You can add areference to that is defined in ONET.XML by adding a List element to the collection of lists specified wit] Configuration element. For example, if you define a list type named "My Custom.List" in ONET. with a Type attribute of 143, then you can add <List Title="My_Custom_List" Type=' Url="Lists/My_CustomList" /> to make the list part of the configuration. The following example shov arrangement of configurations in an ONET.XML file for a specific WEBTEMP*.XML file:

```
<Configurations>
  <Configuration ID="0" Name="Default">
    <Lists>
      <List Title="My_Custom_List" Type="143" Url="Lists/My_Custom_List" />
      <List Title="Shared Documents" Url="Shared Documents" QuickLaunchUrl="Shared Documents/Forms/AllItems.aspx"
        Type="101" />
      <List Title="General Discussion" Url="Lists/General Discussion" QuickLaunchUrl="Lists/General Discussion/AllItems.
```

```
      Type="108" />
    <List Title="Announcements" Type="104" Url="Lists/Announcements" />
    <List Title="Links" Type="103" Url="Lists/Links" />
    <List Title="Contacts" Url="Lists/Contacts" QuickLaunchUrl="Lists/Contacts/AllItems.aspx" Type="105" />
    <List Title="Events" Type="106" Url="Lists/Events" />
    <List Title="Tasks" Url="Lists/Tasks" QuickLaunchUrl="Lists/Tasks/AllItems.aspx" Type="107" />
    <List Title="Site Template Catalog" Type="111" Url="_catalogs/wt" RootWebOnly="TRUE" />
    <List Title="Web Part Catalog" Type="113" Url="_catalogs/wp" RootWebOnly="TRUE" />
    <List Title="List Template Catalog" Type="114" Url="_catalogs/lt" RootWebOnly="TRUE" />
  </Lists>
  <Modules>
    <Module Name="Default" />
    <Module Name="WebPartPopulation" />
  </Modules>
 </Configuration>
 <Configuration ID="1" Name="DWS">
  <Lists>
    <List Title="Shared Documents" Type="101" />
    <List Title="Image Library" Type="109" />
    <List Title="General Discussion" Url="Lists/General Discussion"
      QuickLaunchUrl="Lists/General Discussion/AllItems.aspx" Type="108" />
    <List Title="Announcements" Type="104" Url="Lists/Announcements" />
    <List Title="Contacts" Url="Lists/Contacts" QuickLaunchUrl="Lists/Contacts/AllItems.aspx"
      Type="105" />
    <List Title="Links" Type="103" Url="Lists/Links" />
    <List Title="Events" Url="Lists/Events" QuickLaunchUrl="Lists/Events/AllItems.aspx"
      Type="106" />
    <List Title="Tasks" Type="107" />
    <List Title="Site Template Catalog" Type="111" Url="_catalogs/wt" RootWebOnly="TRUE" />
    <List Title="Web Part Catalog" Type="113" Url="_catalogs/wp" RootWebOnly="TRUE" />
    <List Title="List Template Catalog" Type="114" Url="_catalogs/lt" RootWebOnly="TRUE" />
  </Lists>
  <Modules>
    <Module Name="DWS" />
    <Module Name="WebPartPopulation" />
  </Modules>
 </Configuration>
</Configurations>
```

The value of the ID attribute for each **Configuration** element corresponds to the IDs specified in the WEBTEMP*.XMLfile for configurations. In this example, the default site is a Research Collaboration she (0), and not a Research Document Workspace site(]). The **Type** attribute for each **List** elementreferences.a list type defined in ONET.XML. The **Uri** attribute contains the URL for the folder containing the list definition for each list, which includes the ASPX files, SCHEMA.XML, and any related files. When

specified, the **QuickLaunchUrl** attribute contains the full path to the Allitems.aspx file for a list, displays the list in the Quick Launch area.

3.19.8.3 Using Modules to Add Files to a Site Definition

You can specify files to include as part of a site definitio11byusing modules within the ONET.XML f the site definition. Use the Modules .elementwithin a **Ccınfiguration** element to declare modules, an use the **Modules** element outside the configuration to define the URL of any file to include as part definition.

It is recommended that you create a custom site definitionby copying an existing site definition, rathı modifying the original ONET.XMLfile installed with Windows SharePoint Services. Changes that yor to the originally installed file may be overwritten when you install updates or service packs for Wi SharePoint Services, or when you upgrade an installation to the next product version. For informat creating a custom site definition, see Creating a Site Definition from an Existing Definition.

The following example from ONET.XMLillustrates how to declare a module.

```
<Configurations>
 <Configuration ID="0" Name="Default">
  <Lists>
   <List Title="Shared Documents" Url="Shared Documents"
    QuickLaunchUrl="Shared Documents/Forms/AllItems.aspx" Type="101" />
   <List Title="General Discussion" Url="Lists/General Discussion"
    QuickLaunchUrl="Lists/General Discussion/AllItems.aspx" Type="108" />
   <List Title="Announcements" Type="104" Url="Lists/Announcements" />
   <List Title="Links" Type="103" Url="Lists/Links" />
   <List Title="Contacts" Url="Lists/Contacts"
    QuickLaunchUrl="Lists/Contacts/AllItems.aspx" Type="105" />
   <List Title="Events" Type="106" Url="Lists/Events" />
   <List Title="Tasks" Url="Lists/Tasks"
    QuickLaunchUrl="Lists/Tasks/AllItems.aspx" Type="107" />
   <List Title="Site Template Catalog" Type="111"
    Url="_catalogs/wt" RootWebOnly="TRUE" />
   <List Title="Web Part Catalog" Type="113" Url="_catalogs/wp"
    RootWebOnly="TRUE" />
   <List Title="List Template Catalog" Type="114" Url="_catalogs/lt"
    RootWebOnly="TRUE" />
  </Lists>
  <Modules>
   <Module Name="Default" />
  </Modules>
 </Configuration>
</Configurations>
```

In the example, the "Default" module is identified as part of the configuration for the site definition.

In the **Modules** element, the following module definition specifies the Web Parts used on the home page in the Default module.

```
<Modules>
 <Module Name="Default" Url="" Path="">
  <File Url="default.aspx" NavBarHome="True">
   <View List="104" BaseViewID="0" WebPartZoneID="Left"/>
   <View List="106" BaseViewID="0" WebPartZoneID="Left" WebPartOrder="2"/>
   <AllUsersWebPart WebPartZoneID="Right" WebPartOrder="1">
    <![CDATA[<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2"
     xmlns:iwp="http://schemas.microsoft.com/WebPart/v2/Image">
      <Assembly>Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral,
      PublicKeyToken=71e9bce111e9429c</Assembly>
      <TypeName>Microsoft.SharePoint.WebPartPages.ImageWebPart</TypeName>
      <FrameType>None</FrameType>
      <iwp:ImageLink>_layouts/images/homepage.gif</iwp:ImageLink>
      </WebPart>]]>
   </AllUsersWebPart>
   <View List="103" BaseViewID="0" WebPartZoneID="Right" WebPartOrder="2"/>
   <NavBarPage Name="Home" ID="1002" Position="Start"></NavBarPage>
   <NavBarPage Name="Home" ID="0" Position="Start"></NavBarPage>
  </File>
 </Module>
</Modules>
```

Any Web site instantiated from the site definition i~~IBct~S}'/~itet~ı-fjles are specified in this module. In this example, a module for the home page is defined. fachView e1.ement within the module represents a List View Web Part, including the list that is displayed in the \Veb Part and .how the Web Part is positioned on the page. For the left Web Part zone, this module specifies fwoList View Web Parts for the Announcements and Events lists, and. for the right zone, it specifies an Imag~.'YfB P~ı-t containing the image used on home pages and a List View Web Part for the Links list. The **Nav~a.rll.onıe**attribute specifies that this page will serve as the destination page for the **Home** link in the top navigationbars of sites that are created.

3.19.8.4 Customizing the Navigation Areas

This programming task shows how to customize the topriavigatiôn area that is displayed on every page within a SharePoint site and the Quick Launch area that is displayed on home pages. The first example involves modifying the ONET.XML file of the site definition, while the second example involves modifying both ONET.XML and default.aspx.

It is recommended that you create a custom site definition by copying an existing site definition, rather than modifying the original ONET.XML file installed with Windows SharePoint Services. Changes that you make to the originally installed file may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version. For information on creating a custom site definition, see Creating a Site Definition from an Existing Definition. Modifying an originally installed ONET.XML file to customize the navigation areas of pages in already existing sites is not supported.

The **NavBars** element near the beginning of ONET.XML contains several **NavBar** elements, the first of which pertains to the top navigation area and is named **SharePoint Top Navbar.** All other **NavBar** elements contained within the **NavBars** element define the Quick Launch area of the home page.

3.19.8.4.1 To customize the top navigation area

Customize the top navigation area of future sites created through the site definition by adding **aNavBarLiıı~** element to the first **NavBar** element within .ONET.XML(in *Local_Drive:\M.icrosoft* Shared\Web Serv.er *Extensions\60\TEMPLATE\Locale_ID\SJTE_DEFINITJOMXML).* You can add links to local files and pages within the Microsoft Windows SharePoint Services deployment or links to files and pages located elsewhere.

The following example adds two **NavBarLink** elements, one .that provides a link to a local file call~d Sample.aspx on the server, and another that provides a link to an external URL, http://example.microsoft.Ç◊m.

```
<NavBars>
  <NavBar Name="SharePoint Top Navbar" ID="1002">
    <NavBarLink Name="Documents and Lists"
      Url="_layouts/[%=System.Threading.Thread.CurrentThread.CurrentCulture.LCID%]/viewlsts.aspx">
    </NavBarLink>
    <NavBarLink Name="Create"
      Url="_layouts/[%=System.Threading.Thread.CurrentThread.CurrentCulture.LCID%]/create.aspx">
    </NavBarLink>
    <NavBarLink Name="Site Settings"
      Url="_layouts/[%=System.Threading.Thread.CurrentThread.CurrentCulture.LCID%]/settings.aspx">
    </NavBarLink>
    <NavBarLink Name="Get Current Data" Url="_layouts/Sample.aspx"></NavBarLink>
    <NavBarLink Name="microsoft" Url="http://example.microsoft.com/"></NavBarLink>
    <NavBarLink Name="Help" Url="javascript:HelpWindow()"></NavBarLink>
  </NavBar>
  <NavBar Name="Documents" ID="1004">
  .
  </NavBar>
  <NavBar Name="Pictures" ID="1005">
  .
  </NavBar>
  <NavBar Name="Lists" ID="1003">
  .
  </NavBar>
  <NavBar Name="Discussions" ID="1006">
  .
  </NavBar>
  <NavBar Name="Surveys" ID="1007">
  </NavBar>
</NavBars>
```

The value of the **Uri** attribute in the loca) link is relative to the top-level SharePoint site.

3.19.8.4.2 To customize the Quick Launch area

Just as for the top navigation area, you can add NavBarLink elements to any of the existing NavBar elements to customize the Quick Launch area.

The following example adds two links for existing lists on other sites to the Lists navigation bar. Both links appear beneath the Lists heading on home pages.

```
<NavBar Name="Lists" Prefix="&lt;table border=0 cellpadding=4 cellspacing=0&gt;"
  Body="&lt;tr&gt;&lt;td&gt;&lt;table border=0 cellpadding=0
  cellspacing=0&gt;&lt;tr&gt;&lt;td&gt;&lt;img src='/_layouts/images/blank.gif' ID='100'
  alt='Icon' border=0&gt;&amp;nbsp;&lt;/td&gt;&lt;td valign=top&gt;&lt;a
  ID=onetleftnavbar#LABEL_ID# href='#URL#'&gt;#LABEL#&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;&lt;/td&gt;&lt;/tr&gt;"
  Suffix="&lt;/table&gt;" ID="1003">
    <NavBarLink Name="List_1" Url="/Site1_Name/Lists/List1_Name/AllItems.aspx"/>
    <NavBarLink Name="List_2" Url="/Site2_Name/Lists/List2_Name/AllItems.aspx"/>
</NavBar>
```

You can also define a n.ew navigation bar for displayj~ the Quick Launch area of home pages, which involves modifying both ONET.XML and default.aspx of a site definition.

In ONET.XML, copy one of the existing NavBar elements to create a navigation bar, providing a unique name and ID as follows:

```
<NavBar Name="MyNewNavBar" Prefix="&lt;table border=0 cellpadding=4 cellspacing=0&gt;"
  Body="&lt;tr&gt;&lt;td&gt;&lt;table border=0 cellpadding=0 cellspacing=0&gt;&lt;tr&gt;&lt;td&gt;&lt;img
  src='/_layouts/images/blank.gif' ID='100' alt='Icon' border=0&gt;&amp;nbsp;&lt;/td&gt;&lt;td
  valign=top&gt;&lt;a ID=onetleftnavbar#LABEL_ID#
href='#URL#'&gt;#LABEL#&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;&lt;/td&gt;&lt;/tr&gt;"
  Suffix="&lt;/table&gt;" ID="1009">
    <NavBarLink Name="List_1" Url="/Site1_Name/Lists/List1_Name/AllItems.aspx"/>
    <NavBarLink Name="List_2" Url="/Site2_Name/Lists/List2_Name/AllItems.aspx"/>
    <NavBarLink Name="List_3" Url="/Site3_Name/Lists/List3_Name/AllItems.aspx"/>
</NavBar>
```

Open· default.aspx, which is located in the tpp Jcıl<:lefof thy site definition at *Local_Drive:\Microsoft* Shared\Web Server *Extensions\60\TEMPLATE\1033\SITE_DEFINJTJON,* and find the navigation section on the page, which is marked<!-- Navigation-->.

Add two table rows to the group of rows containing the different sections of the Quick Launch area. The following example inserts a new section, My New Navigation Bar, beneath the ,existing Surveys section. The first row contains the header and provides a link to a Web application, while the second row inserts the new navigation bar on the page by _means of a Navigation control.

```
<TR><TD class="ms-navheader">
  <A HREF="_layouts/<%=System.Threading.Thread.CurrentThread.CurrentUICulture.LCID%>/viewlsts.aspx?BaseType=4">
    Surveys</A>
</TD></TR>
```

```
<TR><TD style="height: 6px"><!--webbot bot="Navigation" S-Btn-Nobr="FALSE" S-Type="sequence"
   S-Rendering="html" S-Orientation="Vertical" B-Include-Home="FALSE" B-Include-Up="FALSE"
   U-Page="sid:1007" S-Bar-Pfx="<table border=0 cellpadding=4 cellspacing=0>" S-Bar-Sfx="</table>"
   S-Btn-Nml="<tr><td><table border=0 cellpadding=0 cellspacing=0><tr><td>
   <img src='_layouts/images/blank.gif' ID='100' alt='Icon' border=0>&amp;nbsp;</td>
   <td valign=top><a ID=onetleftnavbar#LABEL_ID# href='#URL#'>#LABEL#</td></tr></table></td></tr>"
   S-Target TAG="BODY" startspan --><SharePoint:Navigation LinkBarId="1007" runat="server"/>
   <!--webbot bot="Navigation" endspan -->
</TD></TR>

<TR><TD class="ms-navheader"><A HREF="_layouts/Web_Application/webform1.aspx">My New Navigation Bar</A>
</TD></TR>
<TR><TD><SharePoint:Navigation LinkBarId="1009" runat="server"/>
</TD></TR>
```

For changes to become apparent, customizations described in this topic may require resettini Information Services (US).

3.19.8.5 Customizing Themes

You can add new themes or customize existing ones for application to Web sites in Microsoft SharePoint Services 2.0. This programming task shows how to customize an existing theme.

Copy one of the theme fold~rs.in *Local _Drive:* \Program/Files\Comm.on.J:'iles\fyficroS()~ Shared\V Extensions\60\TEMPLATE\THEN.lESand give the.folder a uni.~ue.n~ıne Jn..this example, th MyTheme. This folder co~t~ins cascading style sheets (CSS) files, imag~ files, and other files that styles, formatting, and color for the various user interface (UI) elements that are used in the theme.

Find the .inf file in the copied folder, and rename it with the name given to the folder.

Open the .inf file and assign the same name to the title in the **[info]** section of the file.

Customize the styles defined in the .css files of the copied folder as needed. See CSS Class De Windows SharePoint Services for information about the classes used in Windows SharePoir including sample code füat can be .added to ASPX pages to learn which classes apply to which I The following example from THEME.CSS changes the color that is used for sections of the naviga

```
.ms-navframe{
background:#009999;}
.ms-navline{
border-bottom:1px solid #8D4D03;}
.ms-nav .ms-navwatermark{
color:#008999;}
```

Modify the image files in the copied folder by using the business graphics software of your choice

Add thumbnail and preview image files for your custom theme to the *Local_Drive\Program* Fi Files\MicrosoftShared\Web Server Extensions\60\TEMPLATE\IMAGESdirectory. In this exarr are called rny'Ihurnbnail.png and myPreview.gif.

Add a theme template definition to SPTHEMES.XML, which is the file that determines which themes are available as options on the Apply Theme to Web site page. This XML file is located in the *Local_ Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS\1033 directory. The following example specifies a template for the custom theme.

```
<Templates>
  <TemplateID>mytheme</TemplateID>
  <DisplayName>My Theme</DisplayName>
  <Description>Description</Description>
  <Thumbnail>../images/myThumbnail.png</Thumbnail>
  <Preview>../images/myPreview.gif</Preview>
</Templates>
```

Your custom theme now appears in the list of options on the Apply Theme to Web site page and can be applied to SharePoint sites.

Changes that you make to SPTHEM:Ş§,XMLtı:1ay be <>verwtjtten when you install updates or service packs for Windows SharePoint Services, or when you upgrade art installation to the next product version.

3.19.8.6 Customizing the Logos for SharePointSites

To customize the logos used on the home pages ofSharePöintWebsites, do the following:

1. Open the *Local_Drive\Prdgrnm* Files\Cörnrnön Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\IMAGESdirectory.
2. Copy the images that you want to appear on the hortlepages of your Web sites to this directory.
3. Remove the image files HOME.GIP and HOMEPAGE.GIP, which contain the logos used on the left and right sides of the home page, respectively.
4. Rename the new image files HOME.GIF and HOMEPAGE.GIF.

All sites now display new logos on their home pages.

The image files that you add may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

3.19.8.7 Adding a Document Template, File Type, and Editing Application to a Site Definition

This programming task provides steps for customizing Microsoft® Windows® SharePoint™ Services so that documents can be created or edited from·a third-party application. The task involves creating a document template file to complement the ONET.XML file of the site definition, modifying the DOCICON.XML file, and creating a DLL that provides the same functionality as described for the OpenDocuments control.

Adding the ability to create or edit documents within an application involves the following subtasks:

- Creating a document template file, which İn effect adds a DocumentTemplate element to the ONET.XMLfile of the site definition.

- Adding a Mapping element to DOCICON.XML for a file type icon and for identification of the control to use to open the file,
- Creating a DLL that provides the necessary functions for creating or editing documents in the application.

To perform the customizations described in this topic, you must be an administrator on the front-end server running Windows SharePoint Services.

Changes that you make to originally installed files may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

3.19.8.7.1 Adding a document template

To add an existing application document as a template that can be used in document libraries, create an XML file in the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server *Extensions\60\TEMPLATE\Locale_lD\Site_Definition\XML* folder containing the ONET.XML file. Give the new file a name in the format doctemp* .xml, where * represents arbitrary text used as part of the file name. Windows SharePoint Services searches this directory for any .xml file that begins with "doctemp" and merges its contents with the document template definitions contained in ONET.XML. The advantage to adding a doctemp* .file, as opposed to modifying ONET.XML directly, is that the custom definition will not be overwritten when Windows SharePoint Services is updated.

The following example shows the format of a doctemp* .xml file for adding a document template. The file can be created in Notepad and given any name that begins with doctemp, for example, *doctempmyTemplate.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<DocumentTemplates>
  <DocumentTemplate DisplayName="Blank Document" Type="105" Default="FALSE" Description="A blank document.">
    <DocumentTemplateFiles>
      <DocumentTemplateFile Name="doctemp/Template_Folder/psdimp1.psd" TargetName="Forms/template.psd" Default="True"/>
    </DocumentTemplateFiles>
  </DocumentTemplate>
</DocumentTemplates>
```

The DispJayName attribute of the. DocumentTemplate element specifies the .text that is displayed in the drop-down list for selecting a· document template type when creating a new document library. The Type attribute uniquely identifies the document type and can be any integer that is not already used for another document template. The Default attribute specifies whether the template is selected by default in the drop-down list. The Name attribute of the Decument'l'emplatefrtle element specifies the physical path to the template file on the server computer, while the TargetName attribute specifies the address of the template relative to the document library.

Reset Microsoft Internet Information Services (IIS) for changes to take effect.

3.19.8.7.2 Adding a mapping definition for a file type

To map a file extension to a document type, and to identify the control to use when opening a document that has the file extension, add a Mapping element to DOCICON.XML. DOCICON.XML resides in the

Local_Drive:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\XML directory.

Adding the following line within the ByExtension element in DOCICON.XML maps the .psd extension to an image file named icpsd.gif, which provides the icon that is displayed in document libraries for files of this type:

```
<Mapping                    lue="icpsd.gif"/>
```

To enable editing within an application, the Mapping element must also include EditText and OpenControl attributes, as follows:

```
<Mapping ]                  Value="icpsd.gif" EditT<                  i" OpenControl="ProgID"/>
```

The EditText attribute specifies the application name that is displayed on the drop-down menu when a user clicks the Edit arrow for a document. The Openüontrol attribute specifies the ProglD of the control to use for opening files of the specified type.

The image file that is specified by the Value attribute must reside in the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Exfensioıis\60\TEMPLATE\IMAGES directory. If the specified image file is missing from this directory, a "missing image" icon appears beside the document.

To prevent users fromheingprompted for credentialsJo ' seeJhejcon when Anonymous Access is enabled, you must set inheritable permissions on the image file by performing the following steps:

Right-click the referenced image file in the IMAGES folder, andthen click Properties.

On the Security tab in the Properties dialog box, click Advanced.

On the Permissions tab in the Advanced Security Setthıgs dialog box, ensure that the check box is selected for Allow inheritable permissions from the parent tô propagate to this object and all child objects. Include these with entries specifically defined here.

The default size foriconsin Windows Sharel'ointServices fal6x 16 pixels.

Reset IIS for changes to take effect.

3.19.8.7.3 Adding an editing application

Create a DLL providing the control to load for a document of the specified type whose ProgID identifies the control, for example, SharePoint.OpenDocuments. The control that you create must provide the same methods for creating, opening, and viewing documents as described for the OpenDocuments control.

3.19.8.8 Creating a List Definition

You can create a list definition by modifying two files within a site definition: the SCHEMA.XML file that applies to the list, and the ONET.XMLfile that applies to the site as a whole.

It is recommended that you create a custom site definition by copying an existing site definition, rather than modifying the original files installed with Windows SharePoint Services. Changes that you make to originally installed files may be overwritten when you install updates or service packs for Windows SharePoint

Services, or when you upgrade an installation to the next product version. For information on creating a custom site definition, see Creating a Site Definition from an Existing Definition.

Copy the Custlist folder located in the *Local_Drive:'\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE*\033\Site _Template_Name\LISTS* directory.

Rename the new folder as appropriate for the list.

The following example creates a list to organize shared commuting for employees, and so the folder is named COMMUTING.

Open the SCHEMA.XML file of the new list folder and create field definitions by adding **Field** elements within the empty **Fields** element in the opening **MetaData** element.

Definitions added in the **MetaData** element specify the names and types of fields to use in the list. This example defines seven fields, including Text, Choice, Number, and Note fields. The Choice fields create drop-down lists for making selections.

```
<Fields>
 <Field Name="LinkTitle" DisplayName="Driver" Required="TRUE"/>
 <Field Name="Title" DisplayName="Driver" Required="TRUE"/>
 <Field Name="ParkingLocation" Type="Choice" DisplayName="Park & Ride Location" Required="TRUE">
  <CHOICES>
   <CHOICE>Eastgate</CHOICE>
   <CHOICE>North Park</CHOICE>
   <CHOICE>South Terrace Center</CHOICE>
   <CHOICE>Lake Shores</CHOICE>
  </CHOICES>
 </Field>
 <Field Name="ToWork" Type="Choice" DisplayName="To Work" Required="TRUE">
  <CHOICES>
   <CHOICE>7am</CHOICE>
   <CHOICE>8am</CHOICE>
   <CHOICE>9am</CHOICE>
   <CHOICE>10am</CHOICE>
  </CHOICES>
 </Field>
 <Field Name="FromWork" Type="Choice" DisplayName="From Work" Required="TRUE">
  <CHOICES>
   <CHOICE>4pm</CHOICE>
   <CHOICE>5pm</CHOICE>
   <CHOICE>6pm</CHOICE>
   <CHOICE>7pm</CHOICE>
  </CHOICES>
 </Field>
 <Field Name="Capacity" Type="Number" DisplayName="Capacity" Required="TRUE"/>
```

```
<Field Name="Preferences" Type="Note" DisplayName="Personal Preferences"/>
</Fields>
```

All fields except the last Note field require that users provide information in order to sign up.

After you define the fields, you must specify them as view fields in order for them to display in the list. To do this, search for the View Fields element within the section of SCHEMA.XML that defines the All Items view and add field references, as follows:

```
<ViewFields>
<FieldRef Name="LinkTitle"></FieldRef>
<FieldRef Name="ParkingLocation"></FieldRef>
<FieldRef Name="ToWork"></FieldRef>
<FieldRef Name="FromWork"></FieldRef>
<FieldRef Name="Capacity"></FieldRef>
<FieldRef Name="Preferences"></FieldRef>
</ViewFields>
```

It is required that a custom list definition have a dc:fault description. Add a DefaultDescription element within the MetaData element of the SCHEMA.XML file, as fo11ows:

```
<DefaultDescription>
Use this list to organize shared rides to and from work with others.
</DefaultDescription>
```

To display the list as an option on the Create Page,J:you .IBu.st specify the new .list type within the ListTemplates element of the ONET.XML file for the site definition, located at *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE*\033\[Site_Template_Name]\XML.* To do this, find the ListTemplates element and add a ListTemplate element, as follows:

```
<ListTemplate Name="COMMUTING" DisplayName="Shared Rides" Type="133" BaseType="0"
OnQuickLaunch="FALSE" SecurityBits="12" Description="Save gas and enjoy the ride to work with co-workers."
Image="_layouts/images/ittask.gif">
</ListTemplate>
```

The example specifies both the displayed and internal names, as well as an image and description to use on the Create Page. Since the list definition is based on a the Geiieric List type, the value of its BaseType attribute is set to 0.

The Name attribute must be set to the exact name of the new list folder. To identify the list, assign a numerical ID that is not being used elsewhere and that is less than 1000 to the Type attribute.

The example sets the OnQuickLaunch attribute to FALSE. This attribute specifies whether the option to display the list in the Quick Launch area appears selected by default on the page for creating an instance of the list.

The SecurityBits attribute specifies permissions for a list, where the first bit represents Read access and the second bit represents Write access. The bit for Read access contains one of two possible values: 1 allows all users Read access to all items, and 2 allows users Read access only to items they create. The bit for Write access contains one of three possible values: 1 allows all users to modify all items, 2 allows users to modify

only items they create, and 4 prevents users from modifying the list: This example sets the security bits to 12 which prevents users from modifying the items of other users.

The new list list definition now appears on the Create Page and users can add lists to the Microsoft SharePoir site using the new definition.

For another example about creating custom lists, see Adding a Field to a Custom List.

3.19.8.9 Adding a Field to a List Definition

To add a field to a list definition, make modifications to the SCHEMA.XML file associated with the list a to the ONET.XML file for the site definition. In the programming task described here, an ID box is added the Upload Document form of a document library in order to require that users provide an ID befc uploading documents.

It is recommended that you create a custom site definition by copying an existing site defi1 1ition,'.rather' modifying the original files installed with Windows SharePoint Services. Changes that you make to origins installed files may be overwritten when you install updates or service packs for Windows SharePt Services, or when you upgrade an installation to the next product version. For information on creatin custom site definition, see Creating a Site Definition from an Existing Definition.

1. Make a copy ofthe DOCLIB folder in the following directory:

   Local_ Drive: \Program Files\Common Files\Microsoft .Shared\Web Extensions\60\TEMPLATE\l033*Custom_ Site_Template\LISTS*

2. Rename the DOCLIB folder. For example, MYDOCLIB.

3. To define the new field, open SCHEMA.XML in the copied folder and place a field definition w the opening Fields element of the openingMetalratasection.

In the following example, a field is added with EmployeeID as the user name and Employee ID as the cfü name.

```
<Field Name="EmployeeID" DisplayName="Employee ID" Type="Number" Required="TRUE"
 Description="Enter the ID from your employee badge."/>
```

The Type attribute specifies that the field is a Number data type, and the Required attribute is set to **1** so that users are required to enter a value. The Description attribute contains the text that appears besii ID text box on the Upload Document form.

4. Open the ONET.XML file located at:

   Local_Drive:\Program Files\Common Files\Microsoft Shared\Web Extensions\60\TEMPLATE1033\Custom_Site_Template\XML

5. Specify the list definition within the collection of list definitions contained in the ListTeni element. In the following example, the DisplayName, Image, and Description attributes spec name, image, and description that appear on the Create Page for the Web site.

```
<ListTemplate Name="MYDOCLIB" DisplayName="Document Library with ID" Type="501"
 BaseType="1" OnQuickLaunch="FALSE" SecurityBits="11"
 Description="This document library has an additional employee ID field."
```

```
Image="_layouts/images/itdl.gif" DocumentTemplate="101">
</ListTemplate>
```

The Type attribute uniquely identifies each list definition within the collection of list definitions, and its value must be less than 1000. The Name attribute must be set to the exact name of the custom document library folder.

The Base'Iype attribute of the ListTemplate element specifies a reference to the list base type for document libraries that is defined within the Base'Iypes element in the ONET.XML file. The OnQuickLaunch attribute, set to FALSE in the example, specifies whether the option to display the list in the Quick Launch area appears selected by default on the New Document Library page.

TheSecllrityBits attribute specifies permissions for a list, where the first bit specifies Read accessand the second bit specifies Write access. The bit that gives Read access contains one of two possible values, 1 gives all users Read permission to items on the list, and 2 gives users Read permission only to items they create. The bit that gives Write access contains one of three possible values; 1 gives all users permission to modify items in the list, 2 gives users permission to modify only items they create, and 4 gives users no access to modify the list. Inthis example, 11 gives users permission to Read and Write access to all items in the list.

6. To replace theJtdl.gif file with another image, add the new image to the following directory:

Local _Drive:\Prfgraın Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\l033\IMAGES

3.19.8.10 Customizing the Toolbar for a List

To customize the toolbar for a list, .you need to make changes to the Toolbar element within the SCHEMAJ(ML file for the list. To add a new link to the toolbar, for example, you can copy an existing definition for a link and then modify it. This maintains the nesting of tables that is used for displaying links and ensures that the appearance of the link you add is corisistentwiththe other links.

It is recomm7ndedthatyou create a custom site definitionb'.YfÔI)yİngn existing site.definition, rather than modifyin~ t~e o~ginal SCFIEMAXML file installed 'ıVith JYi~1ows.SharePointServices. Changes that you make to the 01ginally. installed file may. be overwritten \'/he.~ you install updates or service. packs for Windows SharePoint Services, or when you upgrade an installation to thy next product version. For information on creating a custom site definition, see Creating aSite Definition from an Existing Definition.

The following example adds a link on the toolbar ofa listt6 a document that is contained in the Shared Documents folder.

```
<tr><td style="padding-left: 2px;padding-bottom: 2px" width=100%>
<table border=0 cellpadding=0 cellspacing=0 width=100%>
<tr><td width=100% class="ms-unselectednav" colspan=2>
<table cellpadding=0 cellspacing=0 border=0><tr><td valign=top>
<img src="../../_layouts/images/rect.gif">  </td>
<td><A HREF="]]></HTML><HtupVDir/>
<HTML><![CDATA[/Shared Documents/MeetingNotes.doc">]]></HTML>
<HTML>Type Conversions</HTML><HTML><![CDATA[</A></td></tr></table>
</td></tr></table>
</td></tr>
```

The example shows the toolbar link nested within three tables. The two tables in the example are cont within opening and closing TR and TD elements that belong to a third table which serves as the frarr containing aJl the links in the toolbar.

The J-lttpVDir element, which returns the root directory of the current subsite, is used to construe complete path for the URL.

When a new list is created based on this template, a link to a MeetingNotes.doc file appears in the toolba

3.19.8.10.1 Extending Form Libraries

This topic describes ways in which a form library in Microsoft® Windows® SharePoint™ Services. c extended. Third-party developers can create a DLL that provides a launcher control for opening the XMI of a form library in an XML editor that is compatible with Windows SharePoint Services. A propertiı file can be created that defines functionality for a form library.

A form .Iibrary list type is an extension of the Document Library base list type and is designed .for files. A form library provides the foJlowing functionality for working with XML files:

- Property Promotion. Promotes data stored in the XML file as columns in the SharePoint library.
- Link Management. Keeps links in the XML files associated with the template of the form lib
- Merge Forms. Sends multiple XML filesto a client application to be merged.

3.19.8.10.2 Creating and registering a launcher control

Third-party developers can create a DLL that provides the launcher .control for opening XML docume an XML editor. The control must provide the same interface for creating, editing, and mergingXML fi well as for customizing templ~tes, asdescribed for the OpenXMLDocuments control. The control mus a ProgID, such as SharePoint.OpenXMLDocuments, and be instaJled and registered on the client com]

The DOCICON.XML file for a Windows SharePoint Services deployment can be customized so tlı ProgID (for example, Word.Doçumeut) and the file extension ofaspecific document type map to theP of the launcher control and to an icon for representing documents in the form library. For informati customizing DOCICON.XML, see Adding a Document Template, File Type, and Editing Applicatio Site Definition.

3.19.8.10.3 Property Promotion

The XML property promotion process defines how to present information from XML files in the colum Windows SharePoint Services form library. The columns that are to be promoted, and the data that the parser of a form library requires to extract information and store it in list columns,..are defined properties.xfp file, which is stored in the Forms folder of the form library. Each field defined properties.xfp file creates a column in the UserData table of the database, and each XML file in the library adds a corresponding row to the table. XML Property Promotion works in conjunction with cc already in the SharePoint form library but adds additional columns,

Using Collaborative ApplicationNı.t~rkııp~anguage (CAML),yQıı.can create a properties.xfp file and.place it in the Forms folder of a form library;!<? customize. XML Property Promotion.

File format for properties.xfp

A properties.xfp file supports the folfowfogformat in CAML.

Syntax

```
<Fields [ FormAggregation="TRUE|FALSE" ]>
  <Field
   [ ReadOnly="TRUE|FALSE" ]
   [ Hidden = "TRUE|FALSE" ]
   [ Name="Internal Name" ]
   [ DisplayName="Display Name" ]
   [ Node="..." ] or [ PITarget="PI_Target" PIAttribute="PI_Attribute" ]
   [ Aggregation="Count | Sum | ..." ]
   [ MaxLength="255" ]
   [ Required="TRUE|FALSE" ]
   [ Viewable="TRUE|FALSE" ]
   [ Type="Note | Text | Boolean | Number | DateTime | Signature | ProgId | Link" ]  (Signature, ProgId and Link will change to Text on
server)
   [ Format="DateOnly" ]
   [ Sortable = "TRUE" ]
   [ ... ANY OTHER CAML FIELD ATTRIBUTES... ] />
</Fields>
```

Descriptions

Fields element. Required root node that has the following attribute and that contains repeating Field elements:

FormAggregation attribute. Optional Boolean. true if Merge Forms is available for the form library. The default value is false.

Field element. Required repeating element that defines one field that maps to a single column in the form library for property promotion. Possible attributes include the following:

ReadOnly attribute. Optional Boolean. true if values cannot be edited directly in the form.library and then saved to the XML file. The default value is false.

Hidden attıibute. Optional Boolean. true if the field cannot be displayed as a column in the view. The default value is false.

Name attıibute. Optional Text. The name of the field as used in the logic of the SharePoint site, such as in object model calls, CAML, and script. This element is optional for fields of type Signature, Progld, or Link but required for all other types.

DisplayName attribute. Optional Text. The title of the field.

Node attribute. OptionaJ Text. The XPath expression.for the node that is beingprornoted.The XMI parser populates the value of the field with the value in the XML file at the location specified b the value of this. attribute.

PITarget attribute. Optional Text. Used in conjunction with the PIAttribute value. PITargı specifies the name of the processing instruction (Ply.The XML parser.populates the value ofth field with the value in the XML file in the PI specified by PITarget and PIAttribute.

PIAttribute attribute. Optional Text. Used for Link Management PI links, specifying the attribute the PITarget element.

Type attribute. Optional Text. Specifies the field type of the field. Possible values include Note, Te Boolean, Number, Date Time, Signature, Progld, and Link. Signature, .ProdID and. Li change to Text on the server. The type is not extensible by third parties and must be one of thi values in the properties.xfp file. The default value is Text.

Aggregation attribute. Optional Text. If the XPath expression retums.tlıeçollection ofvaluesc; attribute specifies the action on the set returned. This action can be either an aggregation func or an indication of the particular element within the collection.

Possible values include the following:

sum
count
average
min
max
merge

plaintext, Converts node text content into plain text.

first. Specifies that property promotion and demotion be applied to the first element collection.

last. Specifies that. property promotion and demotion be applied to'the last element collection.

MaxLength attribute. Optfô11al Integer. Sets the maximum length of a string for the field. strings are concatenated. The value of the MaxLength attribute is restricted by the quota the user in Windows SharePoint Services. The default value is dependent on the field t governed by site settings.

Required attribute. Optional Boolean. true if the field does not accept null values. The defa is false.

Viewable attribute. Optional Boolean. true if the field is added to the default view. The defa is true.

Format attribute. Optional Text. Used to distinguish between Date and **DateTime** values. DateOnly for type xsd:date in data.

Sortable attribute. Optional Boolean. true if the column can be sorted. Must be **false** for types in Windows SharePoint Services.

The preceding format is suppOrt<:!d for the properties.xfp file, but other attributes of the **Field** element may also be included.

Example

The following example shows the format of a properties.xfp file.

```
<Fields FormAggregation="TRUE" . . . >
 <Field Type="ProgID" . . . Viewable="no"/>
 <Field Type="Link" . . . />
 <Field Type="Signature" . . . />
 <Field Type="Text" DisplayName="Issue"
   Node="/issue/title" . . . />
 <Field Type="Number" DisplayName="# of Actions" Node="/issue/actions/action"
   Aggregation="count" . . . />
 <Field ReadOnly="TRUE" DisplayName="Status" Name="{01Aâ/ \60}"
   Node="/issue/status" Viewable="yes" Type="Text"/>
   .
   .
   .
</Fields>
```

For purposes of illustration only, ellipses have been used in place ôf attributes.

Link Management

Link Management allows you to keep the forms in a form library synchronized with the form library template. The mechanisms for Link Management include the following:

Automatic link management. Önce a form is synchronized with the template; if any site, subsite, or library is renamed, the link is automatically kept synchronized :w.ith the template URL of the form library.

Link management does not occur right away. A .flag is. set and it occurs when the file is requested for download.

Manual relink. If the form is initially out of synch with the fo:rriplate URL of the form library, selecting **Relink forms to this form library** on the Custo:rriizepage allows a ınanual relinking. This is used for uploaded documents, or for form libraries copied from one server to another.

Both mechanisms require that the field to be relinked is specified by using **Type="Link".** If the field to be relinked is in the Pl, use the following format in the properties.xfp file:

```
<Field Type="Link" PITarget="PI_Elem                ute="PI_Attribute"/>
```

For a normal XPath expression, use the following format:

```
Field Type="Lin
```

The **Name** attribute is not specified.

Merge Forms

Merge Forms provides form libraries a mechanism for processing multiple XML files at the same time. Th function must be enabled in the properties.xfp file and implemented by the launcher control through **MergeDocuments2** method as described for the **OpenXMLDocuments** control.

Merge Forms is specified by the **FormAggregation** attribute of the **Fields** element, as follows:

```
<Fields FormAggregation="TRUE">
.
.
.
</Fields>
```

In addition to making the customizations described in this topic, a custom list definition can be created f form libraries. For more information, see Creating a List Definition.

3.19.8.11 Customizing the Shortcut Menu for List Items

This programming task shows how to add a command to the shortcut menu that appears \¥hen you right-eli a list item. The example adds to document libraries a **Check Out & Save** command that checks out selected file and opens the **Save** As dialog box.

The check out and save op~ra~on implemented in this example is not an atomic operation. If the user cane saving changes, the document remains checked out.

This task involves creating a custom Microsoft JScript file that contains the logic for the command, whicl specified by a **CustomJSUrl** attribute within the ONET.XML file of the site definition.

To include this command **in** all document libraries created within a site, open ONET.XML *Local_Drive:'\program* Files\Common Files\Microsoft Shared\Web Ser Extensions\60\TEMPLATE*l033\SITE_DEFINITION_NAME\XML,* and add a **CustomJSUrl** attrit within the opening **Project** element, as follows:

```
<Project Title="Team Web Site" ListDir="Lists" xmlns:ows="Microsoft SharePoint"
 CustomJSUrl="/_layouts/[%=System.Threading.Thread.CurrentThread.CurrentUICulture.LCID%]/custom_ows.js">
```

Specifying the **CustomJSUrl** attribute makes the code in the custom JScript file available throughout a that is created from the site definition. In this example, the custom file is named Custom_ows.js.

It is recommended that you create a custom site definition by copying an existing site definition, rather modifying the original ONET.XML file installed with Windows SharePoint Services. Changes that you m to the originally installed file may be overwritten when you install updates or service packs for Wind SharePoint Services, or when you upgrade an installation to the next product version. For infonnatio creating a custom site definition, see Creating a Site Definition from an Existing Definition.

The code that actually draws the shortcut menu is contained in the OWS.JS file, but you can override functions in this file to add, change, or remove the functionality of the existing menu commands. To find the specific function to customize to suit your needs, trace the path of function calls from the existing **CreateMenu** function in this file.

When making customizations; you should minimize changes to OWS.JS and instead implement a custom .js file as described in this task. Changes to OWS.JS affect all sites on the server and will be at risk of being lost during an update such as through service packs or patches.

To add a custom **Check Out & Save** command to the shortcut menu, you will override the **AddCheckinCheckoutMenuitem** function in OWS.JS by defining a new function of the same name within a custom JavaScript file (Custom_ows.js).

Create a file named Custom_ows.js in the *Local_Drive:\Program* Files'Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS\1033 directory with the following function, which displays either **Check In** or both **Check Out** and **Check Out & Save** on the shortcut menu, depending on whether the file is checked out:

```
function AddCheckinCheckoutMenuItem(m, ctx, url){
  if (currentItemCheckedOutUserId == null)
    currentItemCheckedOutUserId = itemTable.COUId;
  if (currentItemCheckedOutUserId != "")
  {
    strDisplayText = L_Checkin_Text;
    strAction = "NavigateToCheckinAspx('" + ctx.HttpRoot + "', 'FileName=" + url + "')";
    strImagePath = ctx.imagesPath + "checkin.gif";
    CAMOpt(m, strDisplayText, strAction, strImagePath);
  }
  else
  {
    strDisplayText = L_Checkout_Text;
    strAction = "NavigateToCheckinAspx('" + ctx.HttpRoot + "', 'FileName=" + url + "&Checkout=true')";
    strImagePath = ctx.imagesPath + "checkout.gif";
    CAMOpt(m, strDisplayText, strAction, strImagePath);
    strAction = "CheckOutAndSave('" + ctx.HttpRoot + "', 'FileName=" + url + "&Checkout=true')";
    CAMOpt(m, "Check Out & Save", strAction, strImagePath);
    CAMSep(m);
  }
}
```

The second **if** clause draws the menu for when the file is already checked out, and the **else** clause draws the menu for when it is not checked out.

When a user clicks **Check Out & Save** on the shortcut menu, the following method is called:

```
function CheckOutAndSave(strHttpRoot, strArgs)
{
  window.onfocus = RefreshOnNextFocus;
```

```
    SubmitFormPost(strHttpRoot + "/_layouts/" + L_Language_Text +
        "/CheckSave.aspx?" + strArgs + "&Source=" + GetSource());
}
```

This method updates the window to show that the file is checked out. The method also sends the file name to an .aspx page that checks out the file and opens the **Save** As dialog box.

Create a file named CheckSave.aspx in the *Local_Drive:\Program* Fi]es\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS\1033 directory with the following contents:

```
<%@ Page language="C#" EnableViewStateMac="false" %>
<%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
 Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral,
 PublicKeyToken=71e9bce111e9429c" %>
<%@ Import Namespace="Microsoft.SharePoint" %>

<script runat="server">
void Page_Load()
{
  SPWeb web = SPControl.GetContextWeb(Context);
  string strPath = Request.QueryString["FileName"];
  SPFile file=web.GetFile(strPath);

  if ((file.Exists) && (file.CheckOutStatus==Microsoft.SharePoint.SPFile.SPCheckOutStatus.None))
  {
    Response.Clear();
    file.CheckOut();
    Response.AddHeader("Content-Disposition", "attachment; filename=" + file.Name);
    Response.AddHeader ("Content-Length", file.Length.ToString());
    Response.ContentType = "application/octet-stream";
    Response.BinaryWrite(file.OpenBinary());
  }
  else
  {
    Response.Write("The file does not exist or is already checked out.");
  }
}
</script>
```

You can obtain the **PublicKeyToken** value for the current Windows SharePoint Services deployment fr the default.aspx file in the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Ser Extensions\60\TEMPLATE\LCJD(1033 in English)\STS folder, or from information provided for **Microsoft.SharePoint**assembly at Loca/_Drive:\WINDOWSJWINNT\assembly in Windows Explorer.

This page first checks to see thatthe file exists and that it is .not currently checked out. It then checks out the file and returns the contents to the browser. Setting Response.ContentType to "application/octet-stream" causes the browser to open the Save As dialog box.

Reset Internet Information Services {US), create a site based on the site definition you modified, and create a document library to test the new functionality.

With some minor changes, it is possible to make this menu appear only in document libraries based on a specified custom list definition, rather than in all document libraries.

Create a custom list.definition for document libraries (s~e Creating a List Definition), and take note of the value that you assign to the Type attribute of the ListTemplate element (in this example, 555). Modify the AddCheckinCheckoutMenuitem function in Custom_ows.jsas follows:

```
function AddCheckinCheckoutMenuItem(m, ctx, url)
{
  if (currentItemCheckedOutUserId == null)
    currentItemCheckedOutUserId = itemTable.COUId;

  if (currentItemCheckedOutUserId != "")
  {
    strDisplayText = L_Checkin_Text;
    strAction = "NavigateToCheckinAspx('" + ctx.HttpRoot + "', 'FileName=" + url + "')";
    strImagePath = ctx.imagesPath + "checkin.gif";
    CAMOpt(m, strDisplayText, strAction, strImagePath);
  }
  else
  {
    strDisplayText = L_Checkout_Text;
    strAction = "NavigateToCheckinAspx('" + ctx.HttpRoot + "', 'FileName=" + url + "&Checkout=true')";
    strImagePath = ctx.imagesPath + "checkout.gif";
    CAMOpt(m, strDisplayText, strAction, strImagePath);

    if (ctx.listTemplate == 555)
    {
      strAction = "CheckOutAndSave('" + ctx.HttpRoot + "', 'FileName=" + url + "&Checkout=true')";
      CAMOpt(m, "Check Out & Save", strAction, strImagePath);
      CAMSep(m);
    }
  }
}
```

Notice that the value you assign to the ID attribute for the custom list template is used in the line if (ctx.listTemplate== 555), which is where the ID of the list definition is checked. The Check Out & Save command will appear only on shortcut menus for document libraries that are based on this list definition.

3.19.8.12 Customizing the Message Text for Alerts·

You can modify the content and forınat of e-mail alerts about changes made to list items in Microsöft Windows SharePoint Services by customizing XML file~.th.at are installed in the *Local_Drive\Progra:* m Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\L033\XMLdirectory. These.·~les include Notifltem.xml, NotifListHdr.xml, NotifSiteFtr.xml, and NotifSiteHdr.xml, each of them defining a specific part of the overall message that is sent. Each of. these files implements a subset of Collaborative Application Markup Language (CAML) to construct the resulting HTML.

Changes that you make to files described in this topic may be overwritten when.you install updates or service packs for Windows SharePoint Services, or. when you upgrade an installation to the next product version.

Table 3.5 The following table shows the parts of an e-mail alert for immediate notification of changes to a list item and which part each XML file constructs.

)XML file	!!Displayed content	IIHTML content
NotifSiteHdr.*xmlllh~:!/Server _Name/sitesi;*	Alert result *Site Name* *Site _Display _Name*	*<!--[Begin Source: "NotifSiteHdr.xml"]-->* *<HTML dir=ltr>* *<HEAD>* *<TITLE></TITLE>* *<STYLE type="text/css"><!--* *ms-notif-descriptiontext {* *color: black;* *font-family: verdana;* *font-size: 8pt;* *}* *ms-notif-subsmanageheader {* *background-color: #F2F2F2;* *font-family: verdana;* *font-size: 8pt;* *text-align: left;* *text-decoration: none;* *font-weight: bold;* *vertical-align: top;* *}* *ms-notif-titlearea {*

	font-family: verdana;
	font-size: 9pt;
	}
	.ms-notif-pagetitle {
	color: black;
	font-family: arial;
	font-size: 14pt;
	font-weight: normal;
	}
	.ms-notif-sectionline {
	background-color: #2254b1;
	}
	--></STYLE>
	</HEAD>
	<BODY marginheight="0" marginwidth="0" topmargin="0" leftmargin="0" text="#000000"
	link="#1B55FB" vlink="#BB1CFF" alink="#FF1C2C">
	<TABLE height="100%" width="100%" cellpadding="0" cellspacing="0" border="0">
	<TR>
	<TD valign="top">
	<TABLE width="100%">
	<TR><TD class="ms-notif-titlearea" colspan="2" height="1">Alert result</TD></TR>
	<TR><TD class="ms-notif-titlearea" colspan="2" height="1">
	<A href="http://Server_Name/sites/Site_Name">http://Server_Name/sites/Site_Name - Site_Name</A>
	</TD></TR>

NotifListHdr.xmlllList _*Display_ Name* Summary	<!--[Begin Source: "NotifListHdr.xml"]--> <TR><TD> </TD></TR> <TR><TD class="ms-notif-subsmanageheader" colspan="2" height="1">

		<A href="http://Server_Name/sites/Site_Name/Doc_Lib">Doc_Lib</A> Summ </TD></TR>
Notifltem.xml	'File_Name was modified by User_Name at 10/12/2003 8:01 PM.	<!--[Begin Source: "Notifltem.xml"]--> <TR> <TD class="ms-notif-descriptiontext"> <A href="http://Server_Name/sites/Site_Name/Doc_Lib/File_Name">File_Name</A> modified by User_Name at 10/12/2003 8:01 PM. </TD> </TR>
NotifSiteFtr.xml	Go to My Alerts to edit, lldelete, or view your alerts.	<!--[Begin Source: "NotifSiteFtr.xml"]--> <TR><TD> </TD></TR> <TR><TD class="ms-notif-descriptiontext" colspan="2" height="1"> Go to <A href="http://Server_Name/sites/Site_Name/_layouts/1033/MySubs.aspx">My Alerts</A> to edit, delete, or view your alerts. </TD></TR> </TABLE> </TD> </TR> </TABLE> </BODY> </HTML>

To create a custom message for an alert, make customizations to the appropriate XML file as indicated in the previous table.

To construct the HTML for conveying the message, use the CAML GetVar element to return run-time values and the SetVar element to define variables. These elements can be used to return or set values in conjunction with the following CAML control statement elements, which are the only CAML elements supported in the XML files that define message contents:

- ItEqual
- Exprl
- Expr2
- Then
- Else

- HTML
- Switch
- Expr
- Case
- Default

Table 3.6 The following table shows all the variables that are available for returning information about the item that triggered the alert. No other properties of the item are available at run time beyond those listed in the table. These variables can be assigned to the Name attribute of the GetVar element to return current values that can be used to customize a message.

[Name	]füescription
Alert Frequency	lıThe time interval for sending an alert. Possible values include 0 (immediate), 1 (daily),, 2 (kl)
EventType	The type of event. Possible values include 1 (item added), 2 (item modified), 4 (item deleted), 16 (discussio.nadd····ed), 2 (discus. sion ı odified), 64 (discussion deleted), 128 (discussion closed), and 256 (discussion activated).
JıtemNaföe	IIThe title of the item.
lıtemUrl	IIThe absolute URL for the itenı.
IListNam.e	The. rı.arne of the list.
JListUrl	l!The absolute URL for the list.
JModifiedBy	IITiıe display name of!he u~erwho modified the item.
IMySubsUrl	JIThe absolute URL for the My Alerts on this Site page in Site Settings.
js·L· ıte anguage	The locale iden.· tifier (LCID) fôrthe.Ia11giıageused on the site...For exa-ı;:ple, 1ô:33 is the LCID for U.S. English.
ISiteName	IIThe title of the site.
ISiteUrl	!IThe absolute URL for the site.
JTimeLastModified	l!Thetime at which the item was lastgı()clıfiecl.

Example

This. programming taskshows how to use CAML to •add(c~,şf9~}extto alert messages based on whether an item is added, modified, or deleted. The example defines a different CSS style depending on the type of event.

The following code can be added to NotifSiteHdr.xml alongside other code blocks that use the SetVar element to define variables.

```
<SetVar Name="MyColor" Scope="Request">
 <IfEqual>
  <Expr1>
   <GetVar Name="AlertFrequency" />
  </Expr1>
  <Expr2>
   <HTML>0</HTML>
  </Expr2>
  <Then>
   <Switch>
```

```
    <Expr>
     <GetVar Name="EventType" />
    </Expr>
    <Default />
    <Case Value="0">
     <HTML>green</HTML>
    </Case>
    <Case Value="2">
     <HTML>blue</HTML>
    </Case>
    <Case Value="4">
     <HTML>red</HTML>
    </Case>
   </Switch>
  </Then>
  <Else />
 </IfEqual>
</SetVar>
```

The example uses the **GetVar** element to return the frequency of the alert. If the frequency is set to (immediate), then a **Switch** element is executed that determines the color to use in the style ctetirtitl\ Depending on the type of event, each Case element specifies a different color.

Next, add a style definition fo NotifSiteHdr.xml that uses the. **GetVar** element to return the value of MyColor variable and that specifies the font color and font size for the style.

```
<HTML><![CDATA[<!--[ Begin Source: "NotifSiteHdr.htm" ]-->
 <STYLE type="text/css"><!--

  .MyStyle {
   color: ]]></HTML>
   <GetVar Name="MyColor"/>
   <HTML><![CDATA[;
   font-family: verdana;
   font-size: 14pt;
   }

  .ms-notif-descriptiontext {
   color: black;
   font-family: ]]></HTML>
   .
   .
   .
```

Add the following code to NotitListHdr.xml, which determines .the text to display when items are added, changed, or deleted.

```
<SetVar Name="MyText" Scope="Request">
 <Switch>
  <Expr>
   <GetVar Name="EventType" />
  </Expr>
  <Default />
  <Case Value="0">
   <HTML>
    <![CDATA[ Item Added! ]]>
   </HTML>
  </Case>
  <Case Value="2">
   <HTML>
    <![CDATA[ Item Changed! ]]>
   </HTML>
  </Case>
  <Case Value="4">
   <HTML>
    <![CDATA[ Item Deleted! ]]>
   </HTML>
  </Case>
 </Switch>
</SetVar>
```

Depending on the: type pf event, each. **Case** element specifies different text.

To display the text in alert messages, add a row in NotifListHdr.xml that applies the MyStyle definition and that uses the **GetVar** element to return.the value of the MyText variable.

```
<HTML>
 <![CDATA[ <!--[ Begin Source: "NotifListHdr.xml" ]-->
  <TR><TD> </TD></TR>
  <TR><TD class="MyStyle" colspan="2" height="1"> ]]>
</HTML>
<GetVar Name="MyText" HTMLEncode="TRUE" />
<HTML>
 <![CDATA[ </TD></TR>
   <TR><TD class="ms-notif-subsmanageheader" colspan="2" height="1"> ]]>
</HTML>
```

Restart the server for changes to take effect in your alert messages.

Users who subscribe to alerts for all changes that occur in a list will receive an e-mail message that includes a heading indicating the type of event that occurred when an item is added, modified, or deleted.

3.19.8.13 Extending Help

Microsoft® Windows® SharePöint™ Services provides you with the ability to extend the online help system to add help information about your implementation of Windows SharePoint Services or procedures required by your organization. To modify or extend the help system, You must be a member of the local Administrators group on the Web server.

3.19.8.13.1 About the Windows SharePoint Services help system

Windows SharePoint Services help system provides assistance to site users in these ways:

- Assistance text in the user interface. The text in the user interface provides the first level of assistance to Windows SharePoint Services users. If you add a new page to the site, there should be a paragraph at the top explaining the purpose of the page. If you add user interface controls, text on the pages should guide the user through using the-controls.
- "More information on" links to help from user interface text. When you don't have room on the page for all you need to explain to users, use a "More information on <subject>" link in the user interface to point to a custom help topic. In some cases in Windows SharePoint Services, the actual link text is "Show me more information."
- Context-sensitive helpiJınk on the top l~nk bar. In most cases, the Help link on the top link bar brings up a help topic that explains how to use the page. If a page requires more than one hyperlink, a summary topic contains links to all topics that apply to the page.

The "More information on" and context-sensitive links work by using scripted links in the user interface and an XML mapping file. For. more about this subject, see "About mapping to help topics from the user interface."

Note that the user interface does not have links to all help topics in the Windows SharePoint Services help system. There are many concept, procedural, and troubleshooting topics that are accessible only through the table of contents and index.

3.19.8.13.2 Location of the help files on the server

Windows SharePoint Services help components reside in folders containing the actual help content and the XML mapping file. One copy of the help files is installed per server, so users from all sites hosted by the server see the same help files.

In the file system, the default location of the US English language help files is (where C: represents the drive where the Windows SharePoint Services files are installed):

033\STS\

The content folder has a Web address of:

http://<servername>/_vti_bin

There are twosubfolders inthe \STS folder, \HTML and \IMAGES. The HTML folder contains help files in .HTM content as well as supporting .CSS and .JS files. The IMAGES folder contains graphics files for the help system.

By default, the XML mappingfiles reside in the following folder:

C:\Program rver Extensions\60\TEMPLATE\XML\HELP\STS.XML

There is one XML mapping file for all languages.

The folder "1033" indicates that this is the US English version of the help. If you have multiple language packs installed on the server, there will be other directories.

Table 3.6 The following table lists the available language packs and their corresponding identification number.

!Langt1age	**!IFolder/**
!English	/11033
!Japanese	I~
jqerman	ıı I 031
!French	lllö36
/Spanish	1/3082
jıtalian	1/1040
!Dutch	‖ 1043
!Swedish	!IJ053
/Danish	/jt030
!Finnish	/11035
!Norwegian	/11044
!Porttıguese	i1~07ö
!BrazilianPortuguese	/jl046
!Polish	/11045
/Turkish	lll055
/Czech	/Jl0'.f9
!Hungarian	111038
!Greek	ıı1o32
/Russian	‖ 049
/Korean	j/1042
/SimplifiedChinese	/12052
/TraditionalChinese	111028
/Arabic	III§TIJ
/Hebrew	/11037
/Thai	/IfüD

3.19.8.13.3 About mapping to help topics from the user interface

All help links use scripted code to pass a unique keyword to an ASPx .page that redirects the link to a help topic that is specified in an XML mapping file (STS.XML). This allows developers to remap these links without making changes to the Windows SharePoint Services user interface.

To create your own mapping, make your own XML mapping file with a unique name, and place it in the same directory as STS.XML. The precedence for determining which help file appears is as follows:

Windows SharePoint Services searches for a keyword match in all custom XML mapping files. XML mapping files are scanned alphabetically by name.

If no keyword match is found in the custom XML mapping files, Windows SharePoint Services searches keyword match in STS.XML.

If no keyword match is found, Windows SharePoint Services scans alphabetically through all custom XML mapping files for the first valid mapping to the "helphome" keyword.

If no valid mapping for "helphome" is found, Windows Sharel'oint Services uses the "helphome" mapping i STS.XML.

If the "helphome" mapping in STS.XML is invalid, the topic WSTOC.HTM appears.

If a custom XML mapping file contains a mapping that fails .because the path to the topic specified incorrect or the file doesn't exist, the Windows SharePoint Services equivalent help topic appears (ifit exists)

For performance and stability reasons, do not edit the STS.XML file. Create your own copy of the file an edit it as needed. In your copy of the XML mapping file, remove any of the default mappings that you are n overwriting. You must then place your renamed file in the same folder as STS.XML.

Keyword mappings use the syntax:

```
<helpmap>
 <key>KEY_NAME</key>
 <url>PATH/TOPIC_NAME.HTM</url>
</helpmap>
```

"KEY_NAME" represents the keyword included in the help link and "PATH/TOPIC_NAME.HTM represents the path and name of the target help topic under the current language-specific folder. For U English language files, the default location of this folder is:

```
C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI\HELP\1033
```

Because this mapping is relative to each localized folder, any mapping changes apply to all language versio of the help files. So any mapping and content changes that you make to one language version of the help fi must be repeated for each language installed on the server.

For example, the mapping for the help link on the Windows SharePoint Services home page is:

```
<helpmap>
 <key>wssmain</key>
 <url>sts/html/cshmpg.htm</url>
</helpmap>
```

3.19.8.13,4 Adding "More information on" .Iinks

To create a "More infonnation on <subject>" link on your page, use the syntax:

For example:

ure.</A>

You must then add the keyword and create a mapping in your XML mapping file as described in the previous section.

3.19.8.13.5 Mapping the ıuairı help link on a page

The help link in a Windows .Şharepöirit Services.topJink bar points to the topic mapped to the keyword "NavBarHelpHome." By default, this keyword is ınappftto WSTOC.HTM. To change this setting for a single page, insert a help override key into the <BODY> section of your page using the syntax:

```
<script>
var navBarHelpOverrideKey = "KEY_NAME";
</script>
```

For example, the override key used on the site home page is:

```
<script>
var navBarHelpOverrideKey = "wssmain";
</script>
```

3.19.8.13.6 Adding and customizing help topics

To prevent your customizations from getting overwiitten due to updated help files being installed with a service pack, you should create a copy of the Windows SharePoint Services help files in a separate folder on the server drive and. make your changes there. You can then use a custom XML mapping file to point to your version of the help. Complete these steps:

Copy all the contents of the HTML and IMAGES folder into a new folder at the same level in the server's file structure. For example:

C:\Program Files\Common \60\ISAPI\HELP\1033\CUSTOM_HELP\

Place your own topic files in the same directory.

Do not edit the .JS files in the directory, Doing so can cfause Windows SharePoint Services help features to fail for all site users on the server.

Edit the table of contents file (WSTOC.HTM) and index (stsindex.htm) to point to the combination of Windows SharePoint Services and custom help topics that you want your users to see. The table contents uses expando code. For information about using expandos, see "Adding an expando to a topic" later in this topic.

Make a copy of the XML mapping file (STS.XML), rename it (for example CUSTOM_HELP.XML), and place it in the same folder with STS.XML. Windows Sharel'oint Services scans the directory additional mapping files alphabetically, and uses the first keyword match that it finds. STS.XML always scanned last.

Edit your copy of the XML mapping file to change any mappings that you want to point to your help folder. For example, to change the target topic for the help link on the site home page, you would change:

```
<helpmap>
 <key>wssmain</key>
 <url>sts/html/cshmpg.htm</url>
  </helpmap>
```

to:

```
<helpmap>
  <key>wssmain</key>
  <url>CUSTOM_HELP/html/cshmpg.htm</url>
</helpmap>
```

3.19.8.13.7 Adding an expandoto a topic

Expandos are sections of text.rthat expand and collapse when the user clicks ,on them. They are used extensively on the help home page (WSTOC.HTM) and throughout the help system. To use expandos in your help topics you'll need the following:

- The code in the header of the help topic to call the expand/collapse functions in the sExpCollapse.js file.
- The code to expand or collapse all expandos near the top of the body section of any topics that use expandos. This control is needed for accessibility reasons.
- The text that appears when the expando is collapsed must be contained in a special <A> link tag.
- The text that appears when the expando is expanded must be placed within specially coded <DIV> tags.

3.19.8.13.8 Code calling sExpCollapse.js

Make sure the following code is placed in the header section of the help topic:

```
pe="textja\                                                    t>
```

3.19.8.13.9 Expand all/Collapse all control

Place the following code in the <BODY> section of the topic after the table that holds the code for the top navigation bar and before the <Hl> and <Title> tags.

```
<div id="divShowAll" STYLE="display:block;" align="right">
<a href="javascript:AlterAllDivs('block');" class="DropDown">
<img id="picHeader" border="0" src="../images/expandtri.gif" alt="Show All">Show All</a></div>
<div id="divHideAll" STYLE="display:none;" align="right">
<a href="javascript:AlterAllDivs('none');" class="DropDown">
<img id="hide" border="0" src="../images/collapsetri.gif" alt="Hide All">Hide All</a>
</div>
```

This code will already be present in any Windows SharePoint Services help topics that already have expandos. However, you will need to add it to any new topics that you create.

3.19.8.13.10 Creating an expando

Expandos use the syntax:

```
<p>
<a class="DropDown" href="javascript:ToggleDiv('divExpCollAsst_<UNIQUE_NUMBER>')">
<img src="../images/blueup.gif" border="0"
id="divExpCollAsst_<UNIQUE_NUMBER>_img">Title for expando</a>
</p>
<div class="ACECollapsed" border="0" id="divExpCollAsst_<UNIQUE_NUMBER>">
<p>Text to appear in the body of expando when it is expanded.</p>
</div>
```

Add the cod~)? yo~rtopk where you want the expando.to ~ì?Pt~f, a~d,then replace the text for the title and body oftheexpando.<UNIQUE_NUMBER> identifies the places where each expando must have the unique number iden,ü~ef~~/t]ı~t{?Pi~forthat expando. The nun1bersdo not need to be sequential, and need to be unique only within the scope of the current topic. For example:

```
<p>
<a class="DropDown" href="javascript:ToggleDiv('divExpCollAsst_2')">
<img src="../images/blueup.gif" border="0"
id="divExpCollAsst_2_img">Customizing the home page</a>
</p>
<div class="ACECollapsed" border="0" id="divExpCollAsst_2">
<p>This is the home page for your site. It is made up of Web Parts that show you views of the
lists and libraries in your site. You can add more Web Parts to this page, change the layout of
this page, or change the settings for the Web Parts. The links below tell you how.</p>
<ul>
<li><a HREF="dwpdem.asp" id="SP01025935" lcid="">Overview of Web Part Pages</a></li>
<li><a HREF="dmwp.asp" id="SP01003446" lcid="">Modify Web Parts</a></li>
</ul>
</div>
```

3.19.8.14 Working with web.config Files

In a Microsoft® Windows® SharePoint™ Services deployment, web.config files are contained in the following folders within the file system:

- *Local_Drive:\Inetpub\wwwroot* - The web.config file of the top-level content root folder that defines configuration settings for the virtual server ofthe .default Web site in the Windows SharePoint Services deployment. In addition, the content root folder includes a vwpresoıırces folder that contains a web.config file used in Web Part resources.
- *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\wpresources -A web.config file that is used in Web Part resources for the Global Assembly Cache (GAC).
- *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\CONFIG - The web.config file and other .config files that together define configuration settings for extending other virtual servers.
- *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI — The web.config file that defines configuration settings for the/_vti_bin virtual directory.
- *Local _Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS- The web.config file that defines configuration settings for the *I*_layouts virtual directory.
- *Local _Drive:\Program* Files\Common Files'Microsoft Shared\We,b Server Extensions\60\TEMPLATE\ADMIN\Loca/e*JD* - The web.config file that defines configuration settings for pages used in SharePpintCentral Administration.

When a virtual server is extended with Windows Sharel'oirıt Services, a top-level web.config file is placed within the content root folder of the extended virtual server, which defines configuration settings for the server such as HTTP handling for Web Parts. Another web.config file used for Web Part resources is placed in a \wpresources folder within the same content root folder.

The web.config files in the \wpresources folders should not be modified. The settings of these files disallow pages or items that can be compiled. If these settings are changed, the security state for the compiled code will be very different from the security state for code that runs from theLocal_Drive:\Inetpub\wwwroot directory.

Typical customizations involving web.config files include such contexts as the following:

Adding Custom Configuration Settings for Extending Virtual Servers.

Modifying Configuration Settings for an Application to Coexist with Windows SharePoint Services.

Registering a Web Part Assembly as a Safe Control.

Changes that you make to web.config may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

3.19.8.14.1 Adding Custom Configuration Settings for Extending Virtual Servers

The \60\CONFIG folder contains .config and .xml files that are used together to create the web.config file for a virtual server when.it is extended with Microsoft® Windows® SharePoint™ Services. Before copying the web.config file from the \60\CONFIG folder to the root folder of the virtual server, Windows SharePoint Services searches the \60\CONFIG folder for any .xml file with a name in the format webconfig.* .xml and merges its contents with the web.config file before writing the resulting web.config file to the root path of the virtual server. The actions defined in the .xml file are applied to the configuration settings of the virtual server.,A major advantage to using an .xml file to supplement the web.config file is that customizations are not lost.when Windows SharePoint Services is upgraded and the web.config file is overwritten.

When saved as *webconfig.myName.xml* in the \60\CONFIG directory, the following example adds a safe control and replaces the run-time filter for the resulting web.config file that is created when a virtual server is extended.

```
<actions>
  <add path="configuration/SharePoint/SafeControls">
    <SafeControl
      Assembly="System.Web, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      Namespace="System.Web.UI.WebControls"
      TypeName="*"
      Safe="True"/>
  </add>
  <remove path="configuration/SharePoint/RuntimeFilter/>
  <add path="configuration/SharePoint">
    <RuntimeFilter
      Assembly="Company.Product, Version=1.0.1000.0, Culture=neutral, PublickKeyToken=1111111111"
      Class="MyRuntTimeFilter",
      BuilderUrl="MyBuilderUrl"/>
  </add>
</actions>
```

The example adds a new **SafeControl** child element on the path configuration/SharePoint/SafeControls, removes the **RuntimeFilter** element from the configuration/SharePoint/RuntimeFilterpath, and adds a new **RuntimeFilter** element on the configuration/SharePointpath.

These actions are applied to configuration settings when one of the **ExtendVirtualServer** methods or the **ExtendVirtualServerlnWebFarm** method of the **SPGlobalAdmin** class is called. If there are several *webconfig.myName.xm1*files, the order in which they apply is not guaranteed.

Changes that you make to web.config may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installationto the next product version.

For general information about the web.config files used in a Windows SharePoint Services deployment, see Working with web.config Files.

3.19.8.14.2 Modifying Configuration Settings for an Application to Coexist with Windows SharePoint Services

To make an application or Web application coexist with Microsoft® Windows® SharePoint™ Services, customize the web.config file of the application.

Exclude the path of the application from Windows SharePoint Services by performing the following steps:

On the Virtual Server Settings page in SharePoint Central Administration, click Define managed paths.

In the Add a New Path section of the Define Manage Paths page, type the URL to exclude in the Path box, select Excluded path as the option for Type, and click OK.

Clear out the ASP.NET handler used in Windows SharePoint Services and specify the .default ASP.NET handler for all pages by opening the web.config file of the application and adding lines as follows:

```
<httpHandlers>
 <clear />
 <add verb="*" path="*.aspx" type="System.Web.UI.PageHandlerFactory, System.Web, Version=1.0.5000.0, Culture=neutral,
   PublicKeyToken=b03f5f7f11d50a3a" />
   </httpHandlers>
```

Because Windows SharePoint Services maintains a restrictive trust policy, you may need to adjust the trust level by adding a line like the following to the web.config file of the application.

For information on implementing code access security, see Security Overview for Microsoft SharePoint Products and Technologies.

You may need to enable. th;).s~ssion module by adding the. following lines to the web.config file of the application. In the top~lev~t web.config file for Windows SharePoint Services, the default value for the enableSesstionState attribute is false.

```
<httpModules>
 <add name="Session" type="System.Web.SessionState.SessionStateModule"/>
```

If you receive a parser error message saying that the Session module is already in the application and cannot be re-added, remove the httpModules section.

Changes that you make to web.config may be overwritten when you install updates or service packs for Windows SharePoint Services; or when you upgrade an installation to the next product version.

For general information about the web.config files used in a Windows SharePoint Services deployment, see Working with web.config Files.

3.19.8.14.3 Registering a Web Part Assembly as a Safe Control

To register a Web Part as a safe control for use in Microsoft® Windows® SharePoint™ Services, add a SafeControl element that specifies the Web Part in the top-level web.config file of the deployment or virtual server. For information on how to register a Web Part assembly, see Creating a Basic Web Part.

Changes that you make to we'li.ppıı:fi.g may be overwritten when you install updates or service packs for Windows SharePoint Services, or when you upgrade an installation to the next product version.

For general information about the web.config files used in a Windows SharePoint Services deployment, see Working with web.config Files.

3.19.9 Programming Web Services for Microsoft Windows SharePoint Services

This section contains the following programming tasks:

- Returning Lists
- Returning Items from a List
- Updating List Items
- Adding Users to a Cross-Site Group
- Creating a Web Service for Remote Check-In and Check-Out

3.19.9.1 Returning Lists

This programing task shows how to create a simple Windows Form that uses the GetListCollection method of the Lists Web service to retrieve the collection of lists and display their names.

To create a new Window Application project

1. Start Microsoft Visual Studio .NET.
2. On the File menu, point to New, and then click Project.
3. hı the New Project dialog box, click Visual C# Projects, and then select the Windows Application template.
4. Type "ShowLists" as the name and specify the location for the project files, and then click OK.

To add a reference to the Lists Web Service

In Solution Explorer, right-click Web References, and then click Add Web Reference.

In the Add Web Reference dialog box, enter the URL to the Lists Web service on the server running Microsoft Windows SharePoint Services: http://servername/_vti_bin/lists.asmx and then click Add Reference.

To add code to display the collection of lists

Open Form! in Design view, display the Toolbox, and then drag a Label control and a Button control on to the form.

Right-click the Labell control, and then click Properties and delete the value Labell from the Text property of the control.

Double-click the Buttonl control to display the Code Editor, and add the following lines of code to the Click event handler:

```
private void button1_Click(object sender, System.EventArgs e)
{
  // Declare and initialize a variable for the Lists Web Service.
  servername.Lists myservice = new servername.Lists();

  // Authenticate the current user by passing their default
  // credentials to the Web Service from the system credential cache.
  myservice.Credentials = System.Net.CredentialCache.DefaultCredentials;

  // Declare an XmlNode object and initialize it with the XML
  // response from the GetListCollection method.
  System.Xml.XmlNode node = myservice.GetListCollection();

  // Loop through XML response and parse out the value of the
  // Title attribute for each list.
  foreach(System.Xml.XmlNode xmlnode in node) {
    label1.Text+=xmlnode.Attributes["Title"].Value + "\n";
  }
}
```

On the Debug menu, click Start to test the form.

Click Button 1 to display the lists in your SharePoint site.

3.19.9.2 Returning Iteıns>from a List

This programming task shows how to create a Microsoft Windows Form that uses the **GetListItems** method of the Lists XML Web Service to return items from a list.

To create a new Windows Application project

Start Microsoft Visual Studio .NET.

On the File menu, point to New, and then click Project.

In the New Project dialog box, click Visual C# Projects or Visual Basic Projects, and then select the Windows Application template.

Type "ShowListItems" as the name and specify the location for the project files, and then click **OK**.

To add a reference to the Lists Web Service

In Solution Explorer, right-click Web References, and then click Add Web Reference.

In the Add Web Reference dialog box, type the URL for the Lists Web Service on the server running Microsoft Windows SharePoint Services:

http://Server _Name/_ vti_bin/Lists.asmx

and then click Add Reference.

To add code to display the collection oflist items

Open Fonnl in Design view, display the Toolbox, and then drag a Label control and a Button control onto the form,

Right-click the Labell control, and then click Properties and delete the value Labell from the Text property of the control.

Double-click the Buttonl control to display the code editor.

If you are working in C#, add the following Jines of code tothe Click event handler:

```
// Declare and initialize a variable for the Lists Web Service.
Server_Name.Lists listService = new Server_Name.Lists();

/* Authenticate the current user by passing their default
credentials to the Web Service from the system credential cache.*/
listService.Credentials = System.Net.CredentialCache.DefaultCredentials;

// Set the Url property of the service for the path to a subsite.
listService.Url = "http://Server_Name/Subsite_Name/_vti_bin/Lists.asmx";

// Instantiate an XmlDocument object
System.Xml.XmlDocument xmlDoc = new System.Xml.XmlDocument();

/* Assign values to the string parameters of the GetListItems method,
using GUIDs for the listName and viewName variables*/
string listName = "{17991794-81BB-494F-9910-CFBF1093A7CF}";
string viewName = "{7137FFF8-48FF-4C69-8C76-0E3BBD1EA7F9}";
string rowLimit = "150";

/*Use the CreateElement method of the document object to create elements
for the parameters that use XML*/
System.Xml.XmlElement query = xmlDoc.CreateElement("Query");
System.Xml.XmlElement viewFields = xmlDoc.CreateElement("ViewFields");
System.Xml.XmlElement queryOptions = xmlDoc.CreateElement("QueryOptions");

/*To specify values for the parameter elements (optional), assign CAML fragments
to the InnerXml property of each element*/
query.InnerXml = "<Where><Gt><FieldRef Name=\"ID\" />" +
"<Value Type=\"Counter\">3</Value></Gt></Where>";
```

```
viewFields.InnerXml = "<FieldRef Name=\"Title\" />";
queryOptions.InnerXml = "";

/* Declare an XmlNode object and initialize it with the XML response from
the GetListItems method.*/
System.Xml.XmlNode nodeListItems = listService.GetListItems(listName, viewName,query,viewFields,rowLimit,queryOptions);

// Loop through each node in the XML response and display each item.
foreach (System.Xml.XmlNode listItem in nodeListItems)
{
  label1.Text += listItem.OuterXml;
}
```

If you are working in Visual Basic .NET, add the following lines to the ***Click*** *event handler in order to return all items:*

```
Dim serviceLists As New Server_Name.Lists()

serviceLists.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim listName As String = "{17991794-81BB-494F-9910-CFBF1093A7CF}"
Dim viewName As String
Dim query As System.Xml.XmlNode
Dim viewFields As System.Xml.XmlNode
Dim rowLimit As String
Dim queryOptions As System.Xml.XmlNode

Dim responseNode As System.Xml.XmlNode
Dim itemNode As System.Xml.XmlNode

responseNode = serviceLists.GetListItems(listName, viewName, query, viewFields, rowLimit, queryOptions)

For Each itemNode In responseNode

  Label1.Text = Label1.Text + itemNode.InnerXml

Next
```

You .must provide your own values for the listName, viewName, and rowl.imit variables that are used in the example.

On the **Debug** menu, click **Start** to test the form. Click **Buttonl** to display the list items from the specified list.

3.19.9.3 Updating List Items

This programming task shows how to use the **UpdateListitems** method of the Lists Web service to update items in a list.

To create a new Windows Application project

Start Visual Studio .NET.

On the File menu, point to New, and then click Project.

In the New Project dialog box, click Visual C# Projects or Visual Basic Projects, and then select the Windows Application template.

Type "Updatel.istltems" as the name, specify the location for the project files, and then click OK.

To add a reference to the Lists Web Service

1. In Solution Explorer, right-click Web References, and then click Add Web Reference.
2. In the Add Web Reference dialog box, enter the.URL for the.Lists .Web Service on the server running Microsoft Windows SharePoint Services:

   http://Server_ Name!_ vti_bin/Lists.asmx

   and then click Add Reference.

To add code to update list items

Use. the XmlElement class to contain tıiatch ele1119nt in. Collaborative Application Markup Language (CAML), and then use the UpdateListltems method to update the items.

1. Open Forml in Design view, display the Toolbox, and then drag a Button control onto the form.
2. Double-click the Button! control to display the code editor, and then add the following lines of code to the Click event handler:

```
// Declare and initialize a variable for the Lists Web Service.
Server_Name.Lists listService = new Server_Name.Lists();

/* Authenticate the current user by passing their default
credentials to the Web Service from the system credential cache.*/
listService.Credentials = System.Net.CredentialCache.DefaultCredentials;

// Set the Url property of the service for the path to a subsite.
listService.Url = "http://Server_Name/Site_Name/_vti_bin/Lists.asmx";

// Create an XmlDocument object and construct a Batch element and its attributes.
System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
System.Xml.XmlElement batchElement = doc.CreateElement("Batch");
batchElement.SetAttribute("OnError","Continue");
batchElement.SetAttribute("ListVersion","1");
batchElement.SetAttribute("ViewName","{7137FFF8-48FF-4C69-8C76-0E3BBD1EA7F9}");

/* Specify methods for the batch post using CAML. In each method include
the ID of the item to update and the value to place in the specified column.*/
```

```
batchElement.InnerXml = "<Method ID='1' Cmd='Update'><Field Name='ID'>6</Field><Field
Name='Title'>sixth</Field></Method><Method ID='2' Cmd='Update'><Field Name='ID'>7</Field><Field
Name='Title'>seventh</Field></Method>";

// Update list items.
listService.UpdateListItems("{17991794-81BB-494F-9910-CFBF1093A7CF}", batchElement);
```

3. On the Debug menu, click Start to test the form. Click Buttonl to display the list items from the specified list.

3.19.9.4 Adding Users to a Cross-Site Group

To add multiple users .to a cross-site group in. Microsoft Windows Sharel'oint Services, use either 'the AddUserToGroup method, when adding new or existing individual users, or the AddUserCollectionToGroup method, when adding a collection of existing users.

To create a Windows Application project

Start Microsoft Visual Studio .NET.

On the File menu, point to New, and then click Project.

In the New Project· dialog box, click Visual C# Projects or Visual Basic Projects, and then select the Windows Application template.

Type "AddUsers" as the name füd specify the location for the project files, and then click OK.

To add a refere:nce to' thetJserGroup Web Service

In Solution Explorer, right-clickWeb References, and then click Add Web Reference.

In the Add Web Reference dialog box, enter the URL of the Lists Web Service on the server runnin Windows SharePoint Services:

http://Server _Name/_ vti_bin/UserGroup.asmx

and then click Add Reference.

To add users to a group from a site or subsite

Use the GetUserCollectionFromSite method or the GetUserCollectionFromWeb method to return all th users from a site or subsite. Then iterate through the collection of users and add each one to a specified grou by using the AddUserCollectionToGroup method.

1. Open Forml in Design view, display the Toolbox, and then drag a Button control onto the form.
2. Double-click the button to display the code editor, and add the following lines of code to the Clic event handler:

```
// Declare and initialize a variable for the UserGroup Web Service.
Server_Name.UserGroup userGroup = new Server_Name.UserGroup();

/* Authenticate the current user by passing their default
```

```
credentials to the Web Service from the system credential cache and, if adding users
from a subsite, set the Url property for the service.*/
userGroup.Credentials = System.Net.CredentialCache.DefaultCredentials;
userGroup.Url = "http://Server_Name/Subsite_Name/_vti_bin/UserGroup.asmx";

/* Declare an XmlNode object and initialize it with the XML response from either
the GetUserCollectionFromWeb method or the GetUserCollectionFromSite method*/
System.Xml.XmlNode usersSite = userGroup.GetUserCollectionFromWeb();

/* Declare another XmlNode object and initialize it with the first child node
returned from the above method.*/
System.Xml.XmlNode userNode = usersSite.FirstChild;

/*Pass the first child node and its contents as the XMLNode object parameter
for the method.*/
userGroup.AddUserCollectionToGroup("Cross-site_Group_Name",userNode);
```

3. On the Debug menu, click Start to test the form. Click the button to add the users to the specified group.

To add a single user to a cross-site .group and display.the group members

Use the Addflser'I'oGroup method to add a new .or. existinŞüser to a cross-site group, and use the GetUserCollectionFromGroup method to return information about allthe users in a cross-site .groµp.

1. After creating a Windows Application project and a.ddiтg; a. Web reference as described inı-the previous example, add five TextBox controls, a Label co:ıtfol, and a Button control to the form,

2. Double-click the button to display the code editor, and add the following lines of code to the Click event handler:

```
/* Declare and initialize a variable for the UserGroup Web Service, and authenticate the current user by passing their default
credentials to the Web Service from the system credential cache*/
Server_Name.UserGroup userGroup = new Server_Name.UserGroup();
userGroup.Credentials = System.Net.CredentialCache.DefaultCredentials;

// Gather data from the text boxes.
string groupName = textBox1.Text;
string userName = textBox2.Text;
string userLoginName = textBox3.Text;
string userEmail = textBox4.Text;
string userNotes = textBox5.Text;

//Add the specified user to the group.
userGroup.AddUserToGroup(groupName, userName, userLoginName, userEmail, userNotes);

/* Declare an XmlNode object and initialize it with the XML response from the
```

```
GetUserCollectionFromGroup method, declare a second XmlNode object and
initialize it with the first child of the first node returned, and then declare and initialize
an XmlNodeList object with the child nodes of the second node.*/
System.Xml.XmlNode usersNode1 = userGroup.GetUserCollectionFromGroup(groupName);
System.Xml.XmlNode usersNode2 = usersNode1.FirstChild;
System.Xml.XmlNodeList userNodes = usersNode2.ChildNodes;

/* Iterate through the collection of user nodes and parse out the Name, LoginName, Email,
and Notes attribute values for each item and then display the values returned.*/
foreach (System.Xml.XmlNode user in userNodes)
{
  string name = user.Attributes["Name"].Value;
  string loginName = user.Attributes["LoginName"].Value;
  string email = user.Attributes["Email"].Value;
  string notes = user.Attributes["Notes"].Value;

  label1.Text += "\n\n" + name + " : " + loginName + " : " + email + " : " + notes;
}
```

3. On the Debug menu, click Startto test the form. Click the button to add the user to the spec group and to display the group members.

To add users to a group'fröm.information in an XML file

Load the collection of user dafa .füto a Dataset object, and then .use the AddUserToGroup method to through the collection and add each user to the group.

After creating a Windows apphsation and adding a Web reference as described in the previous example, TextBox control and a Button control to the form.

Double-click the button to display the code editor, and add the following lines of code to the **Click** handler:

```
/* Declare and initialize a variable for the UserGroup Web Service, and authenticate the current user by passing their default
credentials to the Web Service from the system credential cache*/
Server_Name.UserGroup userGroup = new Server_Name.UserGroup();
userGroup.Credentials = System.Net.CredentialCache.DefaultCredentials;

/* Instantiate a DataSet control, declare and initialize a string specifying the full path to the
XML file to use as a source*/
DataSet dataSet = new DataSet();
string xmlPath = "Full_Path_To_XML_File";

// Read the data from the XML file into the dataset object.
dataSet.ReadXml(xmlPath);
```

```
// Declare a variable for the group name typed in the text box.
string groupName = textBox1.Text;

/* Iterate through each row in the data set object, assign their values to variables
for each parameter, and add each user to the specified group. */
foreach (DataRow row in dataSet.Tables[0].Rows)
{
  string userName = row["name"].ToString();
  string userLoginName = row["loginname"].ToString();
  string userEmail = row["email"].ToString();
  string userNotes = row["notes"].ToString();

  userGroup.AddUserToGroup(groupName, userName, userLoginName, userEmail, userNotes);
    }
```

On the **Debug** menu, click **Start** to test the form. Click the button to add the user to the specified group and to display the group members.

3.19.9.5 Removing a Meeting from a Meeting '\Vorkşpace

Meetings you create through an events list are not canceled or deleted when the associated eVerı.fis deleted, and it is not possible to cancel or delete a meeting from a Meeting Workspace site through the user interface. This programming task shows how you can remove a ıneetipg\froın a Meeting Workspace by creating a console application that uses methods of the Lists and Meetings;"?{ebgervices.

1. Create a console application in Microsoft Visual Studio® .NET as described in Creating a Console Application.

2. Add Web references for both the Lists and Meetings Web services as follows:

   Right-click Web References in Solution Explorer and click Add Web Reference.

   In the URL box Of the A.dd Web Refereuce dialog box, type the URL/or the .Tfleb service, including the site to service applies and the virtual directory for Web services, such as http://Server/sites!Site _Namel_ vti_bin/Lists.asmx.

   Click Add Reference.

   Repeat this procedure to add a reference to Meetings.asmx for the Meetings Web service.

3. Add a **using** directive **(Imports** in Visual Basic .NET)forthe **System.Xml** namespace to Classl .cs.

4. Add the following code example to the **Main** method; which starts by instantiating the Lists and Meetings services, authenticating the current user for use of both services, and then gathering input from the user that specifies which meeting to delete.

```
// Create neccessary objects and set credentials
Web_Reference_Folder_Name.Meetings meetObj = new Web_Reference_Folder_Name.Meetings();
meetObj.Credentials = System.Net.CredentialCache.DefaultCredentials;

Web_Reference_Folder_Name.Lists listObj = new Web_Reference_Folder_Name.Lists();
```

```
listObj.Credentials = System.Net.CredentialCache.DefaultCredentials;

// Get Meeting Workspace URL
Console.Write("Enter the URL of the Meeting Workspace: ");
string baseURL = Console.ReadLine();

// Set URL properties for both objects
listObj.Url = baseURL + "/_vti_bin/lists.asmx";
meetObj.Url = baseURL + "/_vti_bin/meetings.asmx";

// Get meeting InstanceID
Console.Write("Enter the InstanceID of the meeting: ");
string InstanceID = Console.ReadLine();

XmlDocument xmlDoc = new XmlDocument();

XmlNode ndQuery = xmlDoc.CreateNode(XmlNodeType.Element,"Query", "");
XmlNode ndViewFields = xmlDoc.CreateNode(XmlNodeType.Element,"ViewFields","");
XmlNode ndQueryOptions = xmlDoc.CreateNode(XmlNodeType.Element,"QueryOptions","");

// Create Query to filter based on the InstanceID provided
ndQuery.InnerXml = @"<Where><Eq><FieldRef Name=""ID""/><Value Type=""Counter"">" +
  InstanceID + "</Value></Eq></Where>";

try
{
  // Call GetListItems Web service to get meeting UID
  XmlNode ndResult = listObj.GetListItems("Meeting Series", "", ndQuery, ndViewFields, "0", ndQueryOptions);
  XmlNode MainNode = ndResult.ChildNodes.Item(1);
  MainNode = MainNode.ChildNodes.Item(1);
  XmlNode eventUID = MainNode.Attributes.GetNamedItem("ows_EventUID");
  string UID = eventUID.InnerText;

  // Call RemoveMeeting Web service to cancel meeting
  // NOTE: You might need to increase the sequence number
  // depending on how many updates have already been made to the meeting.
  meetObj.RemoveMeeting(0, UID, 0, DateTime.Now, true);

  Console.WriteLine("Meeting canceled successfully.");
}

catch (System.Web.Services.Protocols.SoapException ex)
{
```

```
Console.WriteLine("Message:\n" + ex.Message + "\nDetail:\n" + ex.Detail.InnerText +
    "\nStackTrace:\n" + ex.StackTrace);
}
```

The example creates an XmlDocument object, which is used to create nodes for constructing. the parameters passed to the GetListItems method. The GetListItems method returns the item from the site's Meeting Series list whose ID is equal .to the instance ID specified by the user. Collaborative Application Markup Language (CAML) is used to construct the query. The ows_EventUID attribute that is returned through the GetListItems method contains the GUID identifying the Meeting Workspace site to be deleted, which is passed to the RemoveMeeting method.

5. On theDebug menu, click Start to test the application.

3.19.10 Creating a Web Service for Remote Check-In and Check-Out

This programming task provides an overview of how to create a Custom Web service that runs in the context of Microsoft® Windows® SharePoint™ Services. The task steps through the process of creating a Web service for checking documents in and out of a document library by using the server-side object model.

3.19.10.1 Basic steps for creating a Web service

1. Create a virtual server on the server running Windows SharePoint Services but on a different port. You must be a local administrator on the server.
2. Create a Web service project'on the new virtual server.
3. Generate and edit a static discovery file and a Web Services Description Language (WSDL)file.
4. Deploy the Web service files to the_vti_bin directory.
5. Create a Windows Application to consume the Web service.

3.19.10.2 Crea.tiiga virtual server on a differentport

1. On <t~e Server that is running Windows SharePoint Services, create a folder in the *Local__.l)l"ivıp:Xl11,e1:pub\wwwroot* directory in which to create.the Web service project.
2. On the taskbar, dick Start; point to Administr::ıtiVeTôols, and then click Internet Information Services (IIS) Manager.
3. In US Manager, expand the *Server _Name* (local cÔıiJ)uter)node of the server where you want to create the virtual server and select Web Sites.
4. On the Action menu, point to New and click Web Site, which launches the Web Site Creation Wizard.
5. Click Next, and on the Web Site Description page type a name in the Description box .forthe Web site that will host the Web service. Click Next.
6. In the Enter the IP address to use for this Web site box of the IP Address and Port Settings page , select the IP address that you want to use, or use the default value (All Unassigned).
7. In the TCP port this Web site should use box, type a port number that is not already in use and click Next.

You do not need to assign a host header because Windows SharePoint Services handles hosting.

8. On the Web Site Home Directory page, browse'to 6t type the file system path for the folder created in step 1 and click Next.

If you do not want to permit anonymous access to your virtual server, clear the Allow anonymous access to this Web site check box.

9. On the Web Site Access Permissions page, select the permissions that you want to use, and then click Next.

Note For best results in most cases, accept the default permissions. By default, the Read permission and the **Run** Scripts (such as ASP) permission are selected. Windows SharePoint Services automatically adds the Execute (such as ISAPI applications or CGI) permission to the appropriate folders.

10. Click Finish to exit the wizard.

3.19.10.3 Creating a Web service on the new virtual server

To create a custom Web service that runs within the context of a Windows SharePoint Services deployment, perform the following steps:

In Microsoft Visual Studio® .NET, on the File menu, point to New, and then click Project.

In the New Project dialog box, select Visual Basic Projects or Visual C# Projects in the Project **Types** box, depending on which language you prefer.

In the Templates box, select.ASP.NET Web Service.

In the Location box, type the following path, using the port number specified for the new virtual server and CheckinOut as the name of the project:

http://Server_Name:Port_#!CheckinOut

Click OK.

Creating a Web service in· Visual Studio .NET from a remote computer involves an extra step in the procedure required for creating a Web service in Visual Studio .NET from the server.

At this point, if.you are working from a remote computer, a Web Access Failed dialog bo~ appears. saying that the file path .does not correspond to the URL previously specified in the New Proje ct dialog box. Perform the following steps:

Under What would you like to do?, ensure that Retry using a different file share path is sellected and type the following patl{fü the Location box, which includes the server name, the file sy.stem path specified for the projectfolder in step 1 of the procedure for creating a virtual server, and the project name specified in step 5 of this procedure:

*\\Server_ Name\LocatDrive$\Inetpub \wwwroot\Proj ect_F*older\CheckinOut

Click OK.

Completing this step creates an ASP.NET Web service project.

Add a reference to the **Windows SharePoint Services** assemblyby performing the following steps:

- In **Solution Explorer,** right-click the **References** node and click **Add Reference** on the shortcut menu.

  If you are using Visual Studio .NET on the server, on the .NET tab of the **Add Reference** dialog box, select **Windows SharePoint Services** in the list of components. Click **Select,** and then click **OK.**

  If you are using Visual Studio .NET on a remote computer, perform the following steps:

  - Click **Browse,** and in the **Select Component** dialog box, navigate to the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI folder on the server running Windows SharePoint Services. You may instead prefer to first copy Microsoft.Sharef'oint.dll from this folder to a local drive on the remote computer and then open this local copy of the DLL in the **Select Component** dialog box.
  - Select Microsoft.SharePoint.dll,click **Open,** and then click **OK.**

2. Right-click the Servicel.asmx file for the project in 'Sotution **Explorer,** click **Rename,** and type **CheckinOut.asmx.**
3. Right-click **CheckinOut.asmx in Solution Explorer,** aıdthen click **View Code.**
4. Add the following directives to Checklnüut.asmx.cs:

```
[Visual Basic .NET]

Imports Microsoft.SharePoint
Imports Microsoft.SharePoint.WebControls
[C#]

using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
```

5. Change the class and constructor names from **Servicel .tc>CheckinOut.**
6. Add the following methods to the **CheckinOut** class:

```
[Visual Basic .NET]

Public Function CheckInDocument(inFile As String, comment As String) As String

  Dim site As SPWeb = SPControl.GetContextWeb(Context)
  Dim file As SPFile = site.GetFile(inFile)

  If file.Exists Then

    If SPFile.SPCheckOutStatus.None = file.CheckOutStatus Then
```

```
        Return "File not checked out."

    Else

        Try

            file.CheckIn(comment)
            Return file.Name + " successfully checked in."

            Catch ex As System.Exception

                Return ex.Message + " :: " + ex.Source

        End Try

    End If

Else

    Return "File does not exist."

End If

End Function 'CheckInDocument

Public Function CheckOutDocument(outFile As String) As String

Dim site As SPWeb = SPControl.GetContextWeb(Context)
Dim file As SPFile = site.GetFile(outFile)

If file.Exists Then

    If SPFile.SPCheckOutStatus.None = file.CheckOutStatus Then

        Try

            file.CheckOut()
            Return "Check-out status: " + file.CheckOutStatus.ToString()

            Catch ex As System.Exception

                Return ex.Message + " :: " + ex.Source
```

```
        End Try

    Else

        Return "File already checked out."

    End If

  Else

    Return "File does not exist."

  End If

End Function 'CheckOutDocument
[C#]

[WebMethod]
public string CheckInDocument(string inFile, string comment)
{
SPWeb site = SPControl.GetContextWeb(Context);
SPFile file = site.GetFile(inFile);

if (file.Exists)
{
if (SPFile.SPCheckOutStatus.None == file.CheckOutStatus)
{
return "File not checked out.";
}

else
{
try
{
file.CheckIn(comment);
return file.Name + " successfully checked in.";
}

catch (System.Exception ex)
{
return ex.Message + " :: " + ex.Source;
}
```

```
        End Try

    Else

        Return "File already checked out."

    End If

  Else

    Return "File does not exist."

  End If

End Function 'CheckOutDocument
```

[C#]

```
[WebMethod]
public string CheckInDocument(string inFile, string comment)
{
SPWeb site = SPControl.GetContextWeb(Context);
SPFile file = site.GetFile(inFile);

if (file.Exists)
{
if (SPFile.SPCheckOutStatus.None == file.CheckOutStatus)
{
return "File not checked out.";
}

else
{
try
{
file.CheckIn(comment);
return file.Name + " successfully checked in.";
}

catch (System.Exception ex)
{
return ex.Message + ":: " + ex.Source;
}
```

```
}
}

else
{
return "File does not exist.";
}
}

[WebMethod]
public string CheckOutDocument(string outFile)
{
SPWeb site = SPControl.GetContextWeb(Context);
SPFile file = site.GetFile(outFile);

if (file.Exists)
{
if (SPFile.SPCheckOutStatus.None == file.CheckOutStatus)
{
try
{
file.CheckOut();
return "Check-out status: " + file.CheckOutStatus.ToString();
}

catch (System.Exception ex)
{
return ex.Message + " :: " + ex.Source;
}
}

else
{
return "File already checked out.";
}
}

else
{
return "File does not exist.";
}
}
```

7. Press *FS* to compile the project.

3.19.10.4 Generating **and** modifying static discovery **and** WSDLfiles

This procedure uses the command line utility that is installed With Visual Studio .NET to generate the static discovery (.disco) file and the WSDL (.wsdl) file.

On the taskbar, click Start, point to All Programs, point to Microsoft Visual Studio .NET, point to Visual Studio .NET Tools, and then click Visual Studio .NET Command Prompt to open a command prompt.

At the prompt type the following command to generate ChecklnOut.disco and ChecklnOut.wsdl files in the *Local _Drive:* \Documents and *Settings\User* folder:

Open the new .disco and .wsdl files and replace the following XML processing instructions in the files:

"?>

with the following page directives:

```
<%@ Page Language="C#" Inherits="System.Web.UI.Page"%>

<%@ Assembly Name="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
<%@ Import Namespace="Microsoft.SharePoint.Utilities" %> <%@ Import Namespace="Microsoft.SharePoint" %>

<% Response.ContentType = "text/xml"; %>
```

Modify the following lines in the .disco file:

```
<contractRef                                                    ref="http://Server_Name:Port_#/ChecklnOut/ChecklnOut.asmx?wsdl"
    docRef="http://Server_Name:Port#/ChecklnOut/ChecklnOut.asmx" xmlns="http://schemas.xmlsoap.org/disco/scl/" />

<soap              address="http://Server_Name:Port#/CheckInOut/CheckInOut.asmx"          xmlns:q1="http://tempuri.org/"
    binding="q1:CheckInOutSoap" xmlns="http://schemas.xmlsoap.org/disco/soap/" />
```

to be as follows:

```
<contractRef ref=<% SPEncode.WriteHtmlEncodeWithQuote(Response, SPWeb.OriginalBaseUrl(Request) + "?wsdl", ""); %>
docRef=<%    SPEncode.WriteHtmlEncodeWithQuote(Response,    SPWeb.OriginalBaseUrl(Request),    ""); %>
xmlns="http://schemas.xmlsoap.org/disco/scl/" />

<soap   address=<%   SPEncode.WriteHtmlEncodeWithQuote(Response,   SPWeb.OriginalBaseUrl(Request),   ""); %>
xmlns:q1="http://tempuri.org/" binding="q1:CheckInOutSoap" xmlns="http://schemas.xmlsoap.org/disco/soap/" />
```

Modify the following line in the .wsdl file:

```
<soap:addr
```

to be as follows:

```
""); %> />
```

Save the files as ChecklnOutdisco.aspx and ChecklnOutwsdl.aspxrespectively.

To include the ChecklnOut service in the list of defaultWindows SharePoint Services Web services that is displayed in the Add Web Reference browser of Visual Studio .NET, perform the following steps:

In Notepad, open the spdisco.aspx file of the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPIfolder.

Add the following lines to the end of the file within the discovery element and save the file:

```
<contractRef ref=<% SPEncode.WriteHtmlEncodeWithQuote(Response, spWeb.Url + "/_vti_bin/CheckInOut.asmx?wsdl", ""); %>
docRef=<% SPEncode.WriteHtmlEncodeWithQuote(Response, spWeb.Url + "/_vti_bin/CheckInOut.asmx", ""); %>
xmlns="http://schemas.xmlsoap.org/disco/scl/" />

<soap address=<% SPEncode.WriteHtmlEncodeWithQuote(Response, spWeb.Url + "/_vti_bin/CheckInOut.asmx", ""); %>
xmlns:q1="http://schemas.microsoft.com/sharepoint/soap/directory/" binding="q1:CheckInOutSoap"
xmlns="http://schemas.xmlsoap.org/disco/soap/" />
```

The text appearing before "Soap" in the binding attribute of the soap element (in this example, binding="ql :ChecklnOutSoap") specifies the name of the class that is defined in the Web service.

3.19.10.5 Copy the Web service files to the _vti_bin directory

The_ vti_bin virtual directory maps physically to the *Local_Drive:\Program* Files\Common Files\Microsoft Shared\Web Server Extensions\ISAPI directory, which contains the default Web service files used in Windows SharePoint Services. Copy the new ChecklnOutwsdl.aspx and ChecklnOutdisco.aspx files, as well as the ChecklnOuta.sn1xfile ofthe. Web service project, to the ISAPI folder. Also<topy the corresponding assembly (ChecklnOut.dll) from the bin folder of the project to the bin subfolder within the JŞAPifolder.

From the _vti_bin directory, a Web service offers its functionality to the site that is specified when adding a Web reference for the service.

3.19.11 Creating a Windows Applicatfo:ıto coüstnne the Web service

1. On anôthe:rcô111puter; open Visual Studio .NET.
2. *0n* the File menu, point to New, and then clickProjett.
3. In the New Project dialog box, click Visual C# Projects or Visual Basic Projects, and then select the Windows Application template.
4. Type a name for the application in the Name box, specify a location for the project files in the Location box, and then click OK
5. In Solution Explorer, right-click Web References, and then click Add Web Reference.
6. In the address bar of the Add Web Reference browser, type the URL for the site to which to apply the service, as follows, and press ENTER:

   http://Server_Name/[sites/][Site_*Name/]*

7. In the Available references window, find the CheckinOut service in the list of Web services and click View Documentation.

   Skipping this step results in adding a reference to all the Web services that are displayed.

8. Click Add Reference to download the service contract for the Web service.

9. Open **Forml** in Design view, display the **Toolbox,** arid then drag two text boxes and two buttons onto the form. The first text box is used to specify the URL of a document when checking it in or out, and the second text box is used for typing a comment when checking in a file.

10. Double-click the **Buttonl** control on **Forml** to display the code-behind file in the code editor, and add the following code for checking out documents to the **buttonl_Click** handler, *replacingWeb_Reference__Folder_Name* with the folder name used for the reference to the **CheckInOut** service as displayed in **Solution Explorer** under **Web References.**

```
[Visual Basic .NET]

Dim svcDocLib As New Web_Reference_Folder_Name.CheckInOut()
svcDocLib.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim str As String = svcDocLib.CheckOutDocument(textBox1.Text)
MessageBox.Show(str)
[C#]

Web_Reference_Folder_Name.CheckInOut svcDocLib = new Web_Reference_Folder_Name.CheckInOut();
svcDocLib.Credentials = System.Net.CredentialCache.DefaultCredentials;

string str = svcDocLib.CheckOutDocument(textBox1.Text);
MessageBox.Show(str);
```

11. Double-click the **Button2** control on **Form1** to display the code-behind file in the code editor, and add the following code for checking in documents **to the button2_Click** handler, replacing*Web_Reference F'9lder_Name* with the folder name used **for the reference to the CheckInOut** service as displayed in **S9lution Explorer** under **Web References.**

```
[Visual Basic .NET]

Dim svcDocLib As New Web_Reference_Folder_Name.CheckInOut()
svcDocLib.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim str As String = svcDocLib.CheckInDocument(textBox1.Text, textBox2.Text)
MessageBox.Show(str)
[C#]

Web_Reference_Folder_Name.CheckInOut svcDocLib = new Web_Reference_Folder_Name.CheckInOut();
svcDocLib.Credentials = System.Net.CredentialCache.DefaultCredentials;

string str = svcDocLib.CheckInDocument(textBox1.Text, textBox2.Text);
MessageBox.Show(str);
Add the following directives to the top of the code-behind file for Form1.cs.
[Visual Basic .NET]

Imports System.Net
```

[C#]

```
using System.Net;
```

12. Press F5 to compile and run the project.

To check out a document from a document library, type the site-relative URL of .the file *(Folder_Name/File_Name)* in the first text box and click the first button. To check the document in, type the site-relative URL in the first text box, type a comment in the second text box, and then click the second button.

3.19.12 Prögramfuing with the Microsoft.SharePoint.Meetings Namespace

A Meeting Workspace site is a Web site for centralizing all the information and materials for one or more meetings. The following topics describe the programming essentials you need to build .NET Framework applicatiôtıs(Jlltôp bf Microsoft Windows SharePoint Services, from identifying existing Meeting Workspace sites on aserveF1o.creating and updating Meeting Workspace sites from your code. The topics provide conceptual information about key programming concepts, as well as code examples.

- Using the Meetings Web Service
- Using the Wiıı.dows SharePôirifServices Object Model
- Additıg a Recurring Event tô Lists ôn Multiple Sites
- Supporting Form Libraries in Meeting Workspace Sites

3.19.12.1 Using the Meetings Web Service

The Meetings Web service helps you to create and manage Meeting Workspace sites. This topic describes how you can use the Web services to perfonn the following tasks:

Identify existing Meeting Workspace sites

Create new Meeting Workspace sites and add meetings

Delete Meeting Workspace sites

Update meeting information on a Meeting Workspace site

3.19.12.1.1 Identifying existing Meeting Workspace sites

The following Microsoft Visual Basic. NET code example lists the Meeting Workspace sites that exist on the server.

ServerURLTextBox is an interface element that is on a form in the Visual Basic .NETproject.

[Visual Basic .NET}

```
Dim ws As New mywss001.Meetings()
Dim myCache As New System.Net.CredentialCache()

Private Sub ListMWS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListMWS.Click
ws.Credentials = myCache.DefaultCredentials()
ws.Url = ServerURLTextBox.Text
If (ws.Url.EndsWith("/")) Then
 ws.Url = ws.Url.Remove(ws.Url.Length - 1, 1)
End If
ws.Url = ws.Url + "/_vti_bin/meetings.asmx"
Dim GetMeetingWorkspacesResponse As System.Xml.XmlNode
If (ws.Url <> "") Then
 GetMeetingWorkspacesResponse = ws.GetMeetingWorkspaces(True)
End If
Dim OuterXml As String
OuterXml = GetMeetingWorkspacesResponse.OuterXml()
MsgBox("OuterXml")
End Sub
```

3.19.12.1.2 Creating a new Meeting Workspace site and Adding a meeting to the site

The following Visual Basic .NET code example creates a Meeting Workspace site and adds a meeting to it.

Server URL TextBox, Meetings uh jectTextBox, MeetingLocation'f extBc:,x, <" J).'I'~'I'f\ll'fTextBox, DTENDTextBox, and CreateWorkspaceButton are all interface elements that areon a form in the Visual Basic .NETproject.

```
[Visual Basic .NET]
Dim ws As New mywss001.Meetings()
Dim tz As New mywss001.TimeZoneInf()
Dim myCache As New System.Net.CredentialCache()
Dim UID As Integer
Dim Sequence As UInt32

Private Sub CreateWorkspaceButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CreateWorkspaceButton.Click
ws.Credentials = myCache.DefaultCredentials()
ws.Url = ServerURLTextBox.Text
If (ws.Url.EndsWith("/")) Then
 ws.Url = ws.Url.Remove(ws.Url.Length - 1, 1)
End If
ws.Url = ws.Url + "/_vti_bin/meetings.asmx"
Dim CreateWorkspaceResponse As System.Xml.XmlNode
If (ws.Url <> "") Then
 CreateWorkspaceResponse = ws.CreateWorkspace(MeetingSubjectTextBox.Text, "MPS#0", System.UInt32.Parse("1033"), tz)
```

```
End If
Dim OuterXml As String
OuterXml = CreateWorkspaceResponse.OuterXml()
Dim MWSURL As String
Dim Start As Integer
Dim Finish As Integer
Start = OuterXml.IndexOf("""")
Finish = OuterXml.IndexOf("""", Start + 1)
MWSURL = OuterXml.Substring(Start + 1, Finish - Start - 1)
Dim MyRand As New System.Random()
UID = MyRand.Next(100, 10000)
Sequence.ToString("0")
ws.Url = MWSURL + "/_vti_bin/meetings.asmx"
ws.AddMeeting("", UID.ToString, Sequence, "2003-03-27T15:00:00-08:00", MeetingSubjectTextBox.Text,
MeetingLocationTextBox.Text, DTSTARTTextBox.Text, DTENDTextBox.Text, False)
MWSURLLink.Text = MWSURL
End Sub
```

3.19.12.1.3 Deleting a Meeting Workspace site

The following Visual Basic .NET code example deletes a specified Meeting Workspace site.

MWSURLLink contains the URL of the Meeting Workspace site.

[Visual Basic .NET]

```
Private Sub DeleteWorkspaceButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
DeleteWorkspaceButton.Click
ws.Credentials = myCache.DefaultCredentials()
ws.Url = MWSURLLink.Text + "/_vti_bin/meetings.asmx"
ws.DeleteWorkspace()
End Sub
```

3.19.12.1.4 Updating meeting information on a Meeting Workspace site

The following Visual Basic .NET code example updates a meeting that exists on a Meeting Workspace site.

MWSURLLink, MeetingSubjectTextBox, MeetingLocationTextBox, DTSTARTTextBox, DTENDTextBox, and CreateWorkspaceButton are all interface elements that are on a form in the Visual Basic .NET project.

[Visual Basic .NET]

```
Private Sub UpdateMeetingButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
UpdateMeetingButton.Click
ws.Credentials = myCache.DefaultCredentials()
ws.Url = MWSURLLink.Text + "/_vti_bin/meetings.asmx"
Sequence.ToString("0")
```

```
ws.UpdateMeeting(UID, Sequence, "2003-03-27T15:00:00-08:00", MeetingSubjectTextBox.Text, MeetingLocationTextBox.Text,
DTSTARTTextBox.Text, DTENDTextBox.Text, False)
End Sub
```

3.19.12.2 Using the Windows SharePoint Services Object Model

The following code examples show how to use the Microsoft® Windows® SharePoint™ Services object model to work with Meeting Workspace sites.

3.19.12.3 Identifying existing Meeting Workspace sites

The following code example prints the names of the Meeting Workspace sites that exist on the top-level site on the server.

```
[C#}
SPGlobalAdmin globAdmin = new SPGlobalAdmin();
System.Uri uri = new System.Uri("http://server_name");
SPVirtualServer vServer = globAdmin.OpenVirtualServer(uri);
SPSite TargetSite = vServer.Sites["http://server_name"];
SPWeb RootWeb = TargetSite.OpenWeb("/");
SPWebCollection spRootWebChildren = RootWeb.Webs;
for (int i =0; i<spRootWebChildren.Count,i++)
{
if(spRootWebChildren[i].WebTemplateId == (int) SPWebTemplate.WebTemplate.Meetings)
{
 Console.WriteLine(spRootWebChildren[i].Name.ToString());
}
}
}
```

3.19.12.4 Deleting exisfö:ığ Meeting Workspace sites

The following code example deletes the Meeting Workspace site with the name "testmws" from the top-level site on the server.

[C#]

```
SPGlobalAdmin globAdmin = new SPGlobalAdmin();
System.Uri uri = new System.Uri("http://server_name");
SPVirtualServer vServer = globAdmin.OpenVirtualServer(uri);
SPSite TargetSite = vServer.Sites["http://server_name"];
SPWeb RootWeb = TargetSite.OpenWeb("/");
SPWebCollection spRootWebChildren = RootWeb.Webs;
for (int i =0; i<spRootWebChildren.Count;i++) {
if(spRootWebChildren[i].WebTemplateId == (int) SPWebTemplate.WebTemplate.Meetings)
{
if(spRootWebChildren[i].Name=="testmws")
```

```
spRootWebChildren.Delete("testmws");
}
}
}
```

3.19.12.5 Supporting Form Libraries in Meeting Workspace Sites

By default, you cannot add form libraries to Meeting Workspace sites. However, you can configure Meeting Workspace sites to support form libraries by performing the following steps.

3.19.12.6 Configure Meeting Workspace sites to support form libraries

Add the following line from \Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\1033\STS\XJyIL\ONET.XML to.· \Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\1033\MPS\XML\ONET.XMLin the <ListTemplate> tag:

<ListTemplate Name="xmlform" DisplayName="Form Library" Type="115" BaseType="1" OnQuickLaunch="FALSE" SecurityBits="11" Description="Create a form library when you have XML-based business forms, such as status reports or purchase orders, that you want to manage. These libraries require a Microsoft Windows SharePoint Services-compatible XML editor, such as Microsoft Office InfoPath 2003." Image="/_layouts/images/itdl.gif" DocumentTemplate="1000"></ListTemplate>

This registers the list template for use with Meeting Workspace sites.

It is recommended that you create a custom site definition by copying an existing site definition, rather than modifying the original ONE'.T.XMIfile installed with Windows SharePoint Services. Changes that you .maketo the originally installed file may be lost when you update Windows SharePoint Services such as through service packs or patches. For information on creating a custom site definition, see Creating a Site Definition from an Existing Definition.

Copy \Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\1033\STS\DOCTEMP\XMLFORM to \Program Files\Common Files\Micfösöft Shared\Web Server Extensions\60\Template\1033\MPS\DOCTEMP\.

Copy \Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\1033\STS\LISTS\XMLFORM to \Program Files\Common Files\Microsofl Shared\Web Server Extensions\60\Template\1033\MPS\LISTS\.

Run the HSResefc:öinmand:

New Meeting Wofkspacesites will now support form libraries.

3.19.12.7 Add a form library to a Meeting Workspace site

The following code example adds a form library to your Meeting Workspace site.

```
[C#]
SPVirtualServer TargetServer = new SPVirtualServer(new Uri("http://server_name"));
SPSite TargetSite = TargetServer.Sites["http://server_name"];
SPWeb RootWeb = TargetSite.OpenWeb("/");
SPWeb MWSWeb = RootWeb.Webs.Add("mwsformtest", " mwsformtest ", "Form Library MWS Test", 1033, "MPS", false, false);
MWSWeb.Lists.Add("FormLib", "FormLib", SPListTemplateType.XMLForm);
```

3.20 Reference

The Windows SharePoint Services section of the SDK contains the following references:

- Class Library
- Web Services
- Co1laborativeApplication Markup Language
- Client-Side API Reference
- Protocols
- Databases

3.20.1 Class Library

This section provides descriptions of the .NET managed namespaces that are included in Microsoft Windows SharePoint Services,

3.20.2 Nam espaces

Table 3.7 The following table showsthe namespaces of Windows SharePoint Services and provides a brief description of each.

[Namespace	][>escri_ption
Microsoft.HTML Trans.Interface	Allows you to write custom applicatio~s that can create an HTMIr version of documents. stored on } ~eıver Ju~ning Windows SharePoint Services for users ""R~. d.<J ngt havı:: the required client application or viewer installed on theirco_p:ıpıiters.
IMicrosoft.SharePoint	;~~;f!~so:1::s~nd members for working with a top-level site and its
Microsoft.SharePoint.Administr~tion	Us.ed for managing the server or server farm in a deployment of Wındows SharePoint Services.
Microsoft.SharePoint.Dsp	Pr?vides th.e .base class for the data retrieval service adapters used in Mıcrosoft Wındows SharePoint Services.
Microsoft.SharePoint.Dsp.OleDb	Provides the data retrieval service adapter for performing queries against OLE DB data sources.
Microsoft.SharePoint.Dsp.SoapPT	Provides the data retrieval service adapter for performing pass-through queries against arbitrary Web services.
Microsoft.SharePoint.Dsp.Sts	Provides the data retrieval service adapter for performing queries against sites, lists, and document libraries in Microsoft Windows SharePoint Services.
Microsoft.SharePoint.Dsp.XmlUrl	Provides the data retrieval service adapter for performiirg queries against arbitrary XML data sources.
Microsoft.SharePoint.Meetings	Provides types and members that can be used to customize Meeting Workspace sites.
Microsoft.SharePoint.Security	Provides a set of code access permission and attribıte classes designed to protect a specific set ofresources and operatiqns, such as access to the Windows SharePoinfServices object model; the ability to do unsafe saving on HTTP Gets, and enabling .pgint-to-point WebPart connections.
/Microsoft.SharePoint.SoapServer	Jjçontains classes that implement the Windows Sharel'oint Services

	Web Service .andWebservices for working with Web. Part pages and! Web Parts. In most cases, the members of these classes are not] designed to be called from the server but remotely from client, applications.
Microsoft.SharePoint.Utilities	Used for various purposes, including encoding strings and processing user: information.
Microsoft.SharePoint. W ebControls	Provides server controls that are used on site and list pages in ai SharePoint site. Its major class is SPControl, from which other controls in this namespace are derived.

3.21 WhıdôwsSbarePöint Services Web Service

The Windows SharePoint Services Web Service provides methodsthat can be used to work remotely with a deployment of Microsoft Windows SharePoint Services. For information on using the methods of a· Web service.

Table 3.8 Services

!service	JJDescription
Administration	fJ~b!!~tl:e!~~~~e~~~g~~:~ing a deployment otWindows SharePOintServices, such as
]Alerts	jjJ>rövi<ieslnetlıödsor workingwi.th alertrlorJisfitems iti a S}ıarePöitıtsite.
Document Workspace	Exposes the Document Workspace Web service and its eleven methods for managing Document Workspace sites and the data they contain.
]Lists	IJ?rovidesmethods for working with lists and list data.
JMeetings	IIProvidesmethods that enable you to create and manage Meeting Woı-kspac:~ sites.
]Permissions	IIPı:cıvicles methods for working with t~e peı:ıfü~siÜ~s for a site or list.
/site Data	~fJJ~:s ~~:~~;~i~~~te:~~:. metadata 6ı->Wst data from sites or. lists in Microsoft
Sites	IPr9vides. a method for returning information about the site templates for a site collection.
!Users and Groups	j~ö\li<fes methods for working with users, site,gr,oups,and cross-site gr,oup~
!Versions	J~()vides methods for workiıg with file versions.
!Views	J!Provi<les methods for working with views of lists.
Web Part Pa es	Pro~ides the methods to send information tô and retrieve information from XML Web servıces.
jWebs	IIProvidesmethods for working with sites and subsites.

3.22 .NET Support and Security in Microsoft Windows SharePoint Services

Microsoft Windows SharePoint Services provides supportfor.NET Web development. The .NET-managed object models serve as a platform for customizing Sharel'oint sites and for integrating custom Web applications developed upon the .NET Framework.

Windows SharePoint Services supports .NET development in the following ways:

- Uses ASP.NET, instead ofISAPI, for base page execution.
- Includes the Web Part infrastructure, which provides not only the views of list data used on every page of a SharePoint site, but also a dynamic means for customizing a site through.Web parts and Web Part Pages.
- Provides a server-side, managed-code object model for code that is executed on the server running Windows SharePoint Services and allows for programmatic access to site and list data. This object model is accessible via ASP.NET or any other server process.
- Offers XML Web services for access from remote computers and applications, including SOAP interfaces with methods foı:accessing data on a SharePoint site.

3.22.1 Security

The following precautions imP.töve security on servers running Windows SharePôint Services. These precautions affect where or how c9de is executed on a SharePoint site:

- For content stored in Windows SharePoint Services, only a registered set of custom server controls operates ona Web page.
- Inline script is not ,executed .in a default page, such as for working with lists or managing a site, although you can implement code-behind pages in which script will run.
- You must install allıexecutable code, including custom server controls, Web Parts, and code-behind classes, physicalty on the'.frô:ıit-endWeb server or, in the case of a server farm, on each ofthe front-end Web servers.
- In an ASP.NET context, you must perform updates to the database as part of a POST request. Windows SharePoint Services throws.an exception if you use a GET to make the request, which would otherwise open security risks.
- By default, Windows SharePoint Services requires that you include security validation on any ASPX page that submits a request to modify the contents of the database. For information on the two kinds of security validation that can be used, see Security Validation and Making Posts to Update Data.

- Code running on one virtual server cannot reference another virtual server unless the first server is the admi11isn-'M';t/'\'irtual server. If the code needs to operate in relation to more than one virtual server, create the application on .the administrative port (in other words, within the *Local_Drive:\Program* Files\Cpmmori Files\Microsoft Shared\Web·Server Extensions\60\TEMPLATE\ADMIN\l033 folder) or set the **RequestFromAdminPort** field of the **SPGlobalAdmin** class to **true.**

- The **Microsoft.SharePoint.Utilities** namespace provides methods for encoding strings that can be used to improve security in a Windows SharePoint Services deployment. As an exantple, suppose someone enters the :f'61lôwing code as a value for the title of a list - <script>alert();</script> ---- and you are running code like the following to display the title of every list within a site collection:

```
SPSite site = SPControl.GetContextSite(Context);
SPWebCollection allSites = site.AllWebs;
foreach (SPWeb subSite in allSites)
{
  SPListCollection allSiteLists = subSite.Lists;
  foreach (SPList subSiteList in allSiteLists)
  {
    Response.Write(subSiteList.Title + "<BR>");
  }
}
```

When you run your code, the script block in the title of the list runs and a message box is displayed. To prevent this from happening, you can use the **HtmlEncode** method of the **SPEncode** class to convert angle. brackets (<" or ">") to HTML entities so that, as a result, the script block does not run and "<script>alert();</script>"is harmlessly displayed.

To run custom code that uses types and members in the **Microsoft.SharePoint** namespaces, users and groups must be assigned the appropriate permissions, just as when interacting with a site or list by using the user interface. For more information on permissions, see Security, Users, and Groups Overview.

CONCLUSIONS

SharePoint Portal Server is a powerful portal solution that provides a single point of access to people, teams, knowledge, and applications. It helps you put information to work, connect people and spaces, and target and personalize information. SharePoint Portal Server 2003 builds on and enhances Windows SharePoint Services by providing enterprise features and services, such as global search, personalized portals, information delivery, and support for enterprise application integration and single sign-on.

SharePoint Portal Server is a reliable, scalable, and easy-to-deploy and manage platform for developing customized portal solutions. With SharePoint Portal Server, you can tie together disconnected islands of data, integrate technology with business processes, and target and personalize information for groups and individual users, so your users can be more productive and effective in their work. SharePoint Portal Server achieves this while taking advantage of your existing infrastructure investments and keeping the total cost of ownership low.

REFERENCES

[1] Microsoft Web Services"www.microsoft.com, msdn library".

[2] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSClassLibrary.asp"

[3] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/soapnsMicrosoftSharePointSoapServer2.asp"

[4] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSCAML.asp"

[5] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSClientRef.asp"

[6] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSDatabases.asp"

[7] Microsoft Web Services "http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSClientAPI.asp"