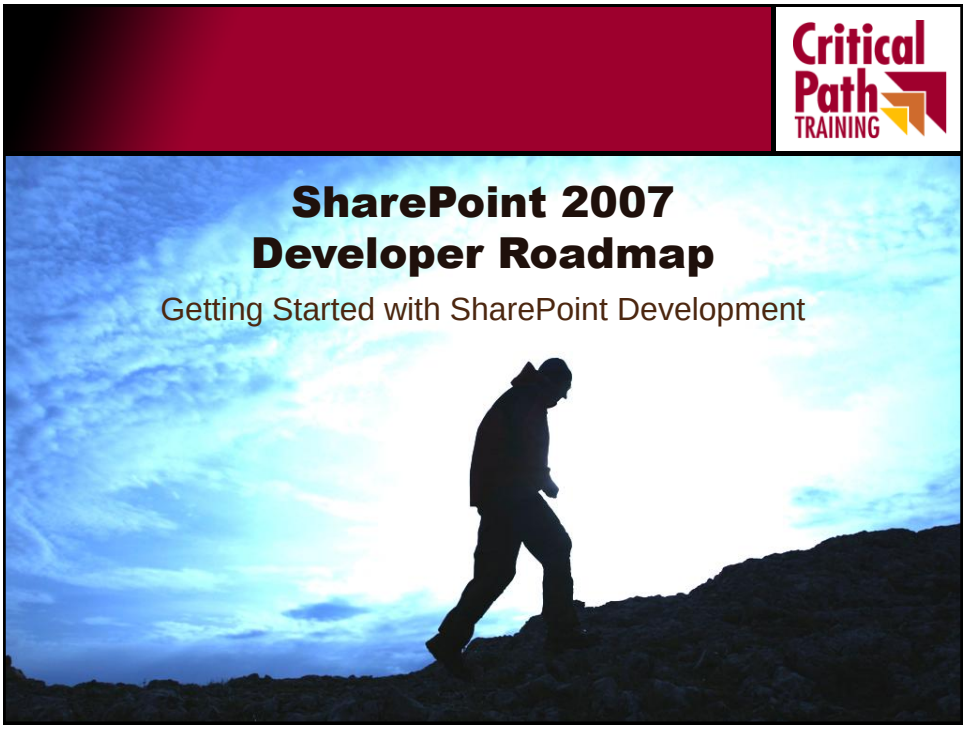
SPT401

Inside SharePoint Products and Technologies

Schedule of lectures

1. SharePoint 2007 Roadmap
2. Developing Features
3. SharePoint Architecture
4. Pages and Site Branding
5. Developing Web Parts
6. Lists and Content Types
7. InfoPath 2007 and Forms Services
8. SharePoint Workflows
9. Extending MOSS Portal and Search
10. Web Content Management
11. Business Data Catalog
12. Excel Services and Report Center
13. Application Security

Revision: v3.0

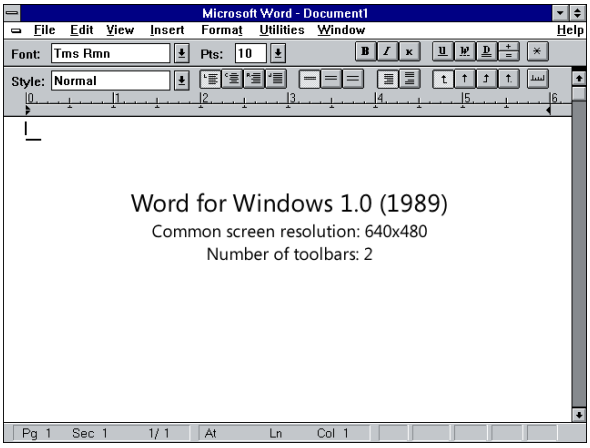


Agenda

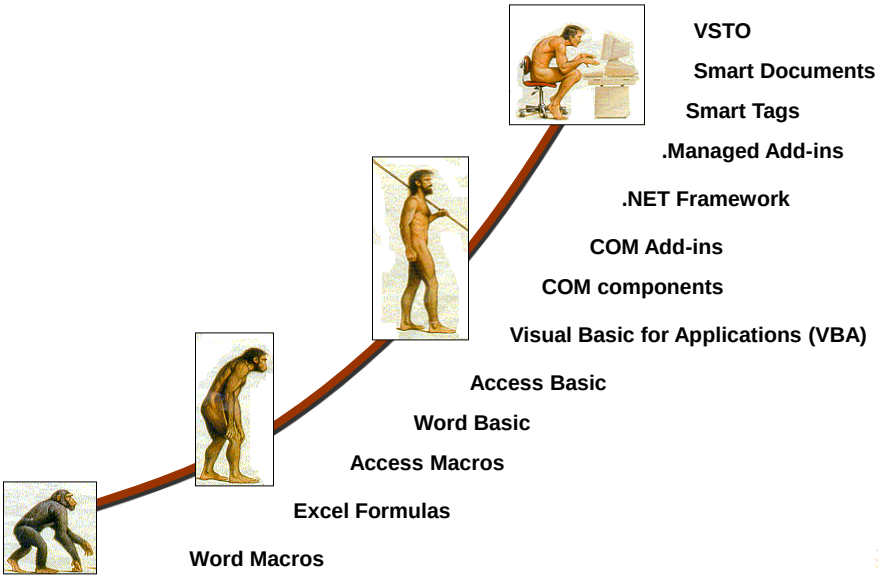
- Architectural overview of SharePoint 2007
  - Windows SharePoint Services 3.0 (WSS)
  - Microsoft Office SharePoint Server 2007 (MOSS)
- WSS as a collaboration solution
- Customizing WSS Sites
- Overview of MOSS components and services

Microsoft Office Through the Ages

- It all started off with a modest productivity tool from a medium-sized company in Redmond

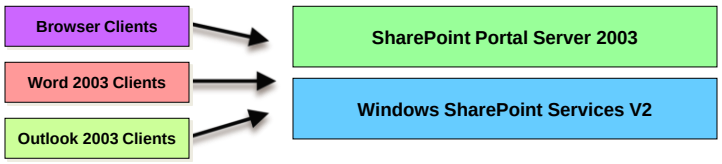


Evolution of the Office Developer

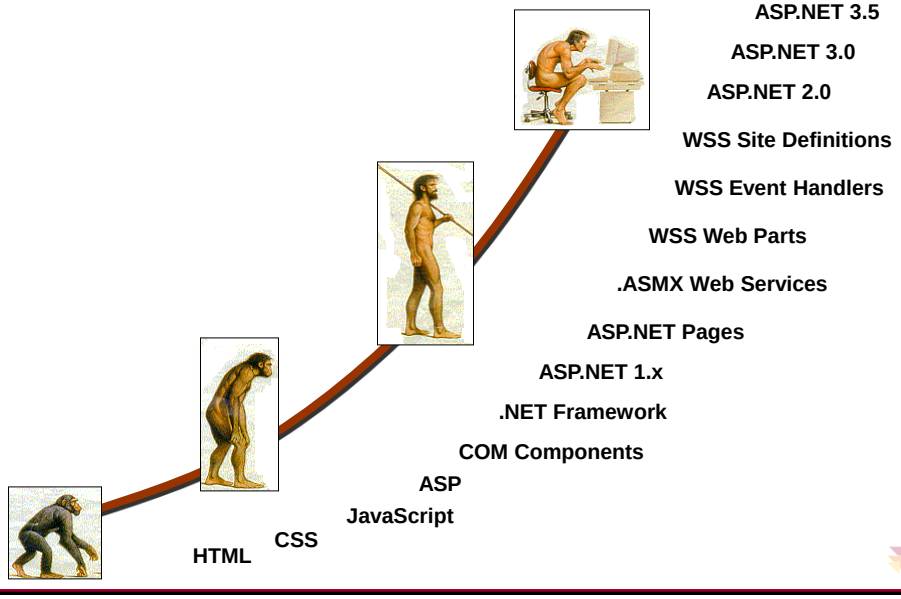


Office 2003 Server Components

- Windows SharePoint Services (WSS v2)
  - Site and Workspace Provisioning Engine
  - Accessibility from browser and Office client applications
  - Out-of-the-box Collaboration Services
- MS Office SharePoint Portal Server 2003 (SPS)
  - Aggregation and search features
  - Social networking (Profiles, Audiences, My Sites)



Evolution of the Web/WSS Developer



Student Questionnaire

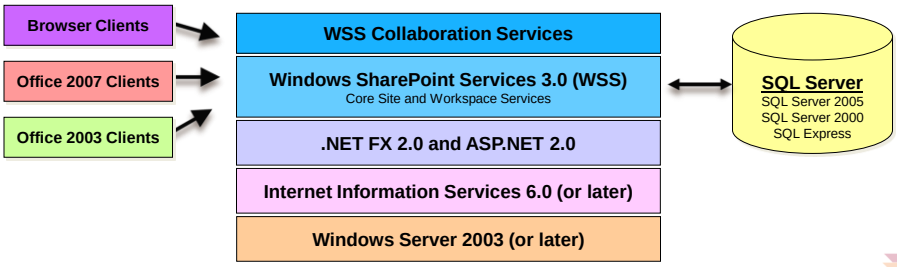
- What's Your Name?
- What Company are you with?
- How have you evolved as a Developer?
- Do you have experience with…
  - The .NET Framework and Visual Studio
  - ASP.NET (what was the latest version)
  - WSS 2.0 and SPS 2003
  - WSS 3.0 and MOSS

Introducing The Office 2007 System

- Windows SharePoint Services 3.0 (WSS)
  - Licensed as part of Windows Server 2003
  - Site provisioning engine and core workspace services
  - Out-of-box collaboration features
  - A development platform
    think of WSS as ASP.NET extensions
- Microsoft Office SharePoint Server 2007 (MOSS)
  - Licensed separately under its own SKUs
  - New components and services built on top of WSS 3.0
  - Unification of SPS 2003 and CMS 2002
  - Lots of functionality rolled in beyond SPS and CMS

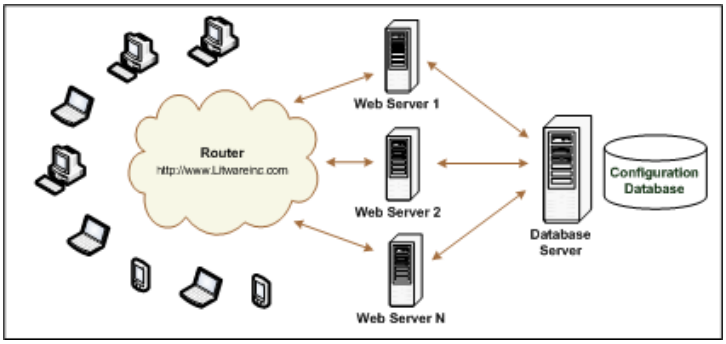
The WSS 3.0 Server-side Platform

- Windows SharePoint Services 3.0 (WSS)
  - An engine for creating/running/managing sites
  - Architecture designed to scale to 10,000s of sites
  - Platform for building Web applications and solutions
  - Collaboration services included out-of-the-box



The WSS Farm

- WSS deployment based on a farm
  - Farm requires Web server(s) and database server
  - Farm can be single server or multi-server
  - Each farm has exactly one configuration database

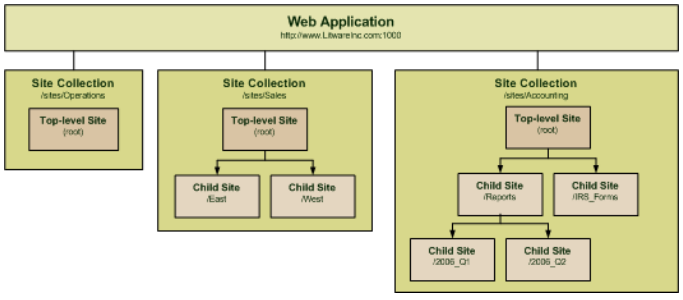


Web Applications

- Web Applications provide HTTP entry points
  - Web Applications based on IIS Web sites
  - Web Application defines one or more URL spaces
  - Web Application security configured independently

Site Collections and Sites

- Web Applications are partitioned using Site Collections
  - Site collection is scope for administrative privileges
  - Site collection always contains top-level site
  - Site collection may contain hierarchy of child sites
  - Web application can support 1000s of site collections

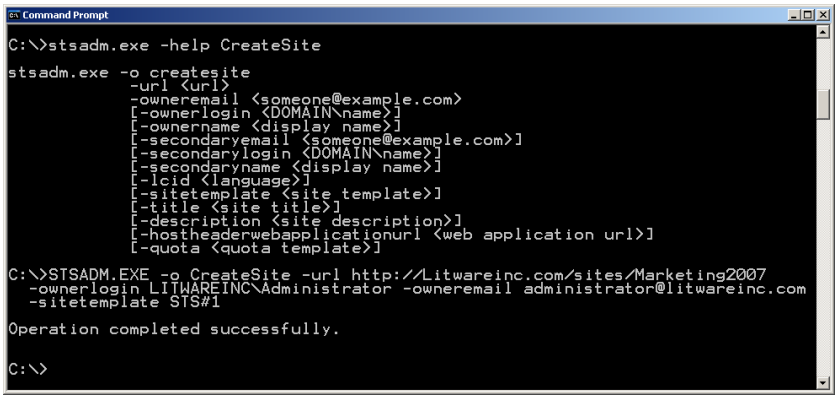


Watch Out: Inconsistent Terminology

New Term	Old Term	WSS Object Model
Site Collection	Site	SPSite
Site	Web	SPWeb

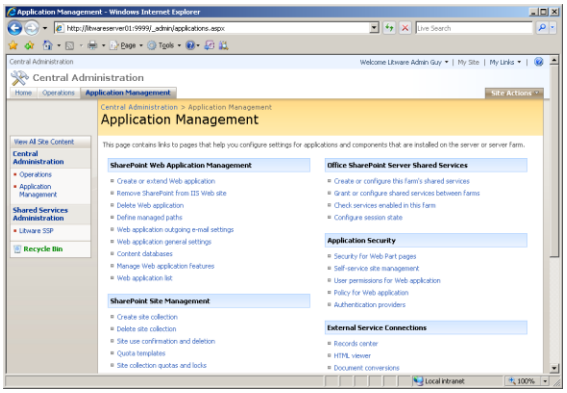
STSADM.EXE Command-line Utility

- Useful for running administrative commands
  - Can be used interactively from command line
  - Commands can be scripted using batch files



WSS Central Administration (WSS CA)

- WSS CA hosted in separate Web Application
  - Used by farm-level administrators
  - WSS CA pages have more links if MOSS is installed



Start >

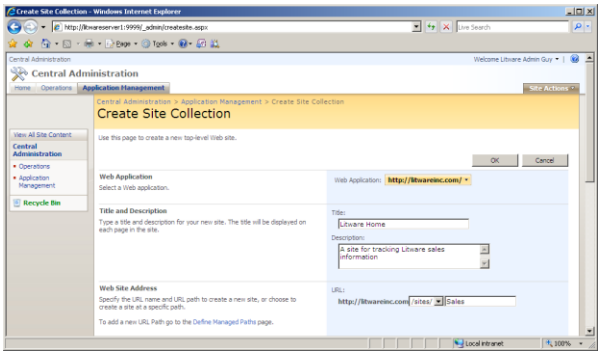
All Programs >

Administrative Tools >

SharePoint 3.0 Central Administration

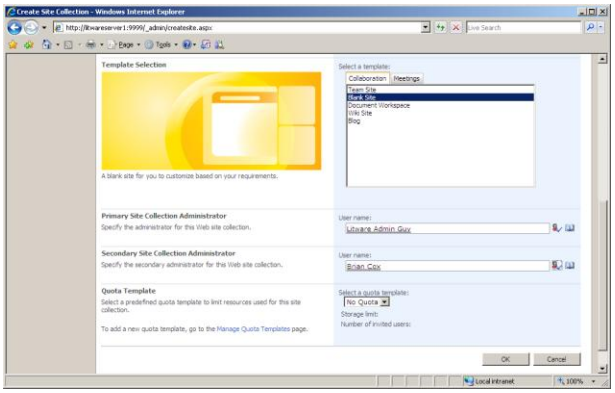
Creating New Site Collections

- Steps to provisioning new site collections
  - Go to the Application Management tab of WSS CA
  - Click the link entitled **Create site collection**
  - Fill out the input form and click OK



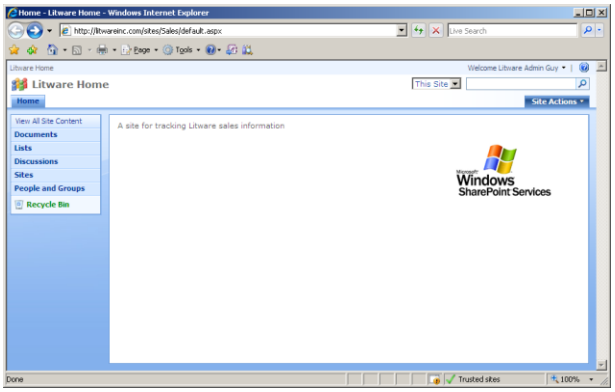
Creating New Site Collections (Part 2)

- Important site collection settings
  - Site template for top-level site
  - Site collection owner(s)



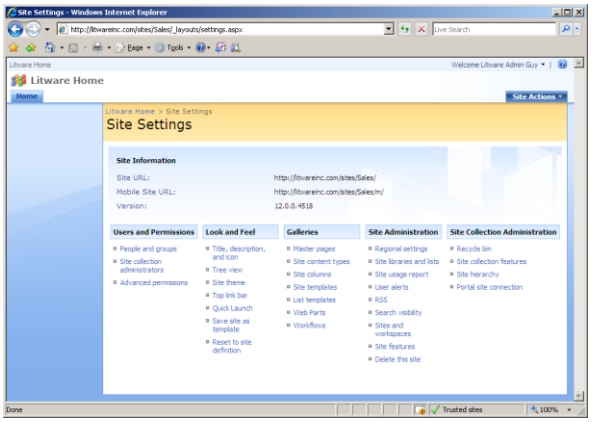
A New WSS Site

- New site collections have a top-level site
  - Site collection owner can provision site elements
  - Site collection owner can create child sites



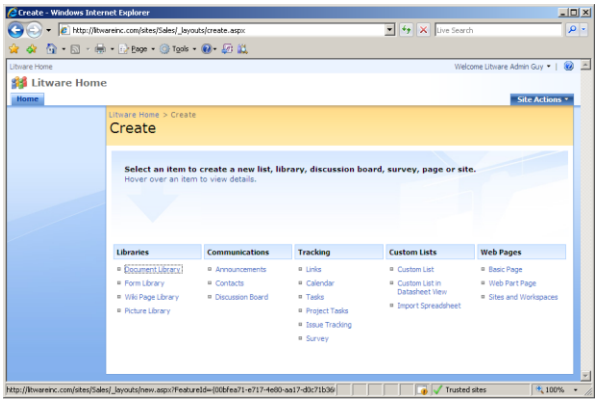
The Site Settings Page

- Site Settings accessible via Site Actions menu
  - Provides links for site and site collection administration



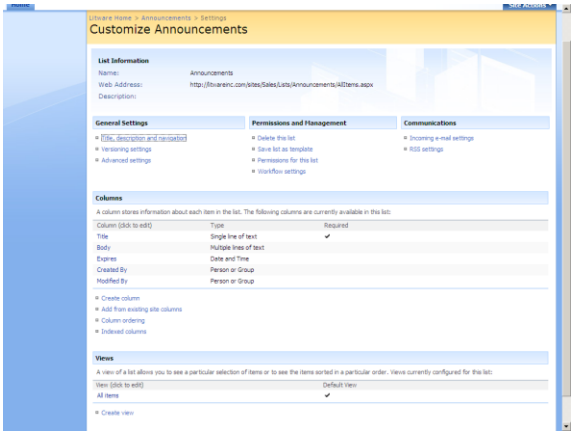
The Create Page

- Create page allows provisioning of site elements
  - WSS provides many collaboration list types out-of-box
  - You can also provision new pages and child sites



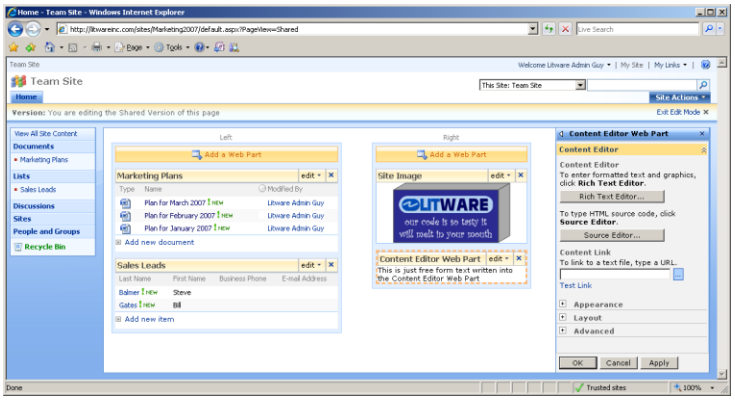
The List Settings Page

- Each List Instance provides a Settings Page
  - You can change list settings and add/remove columns



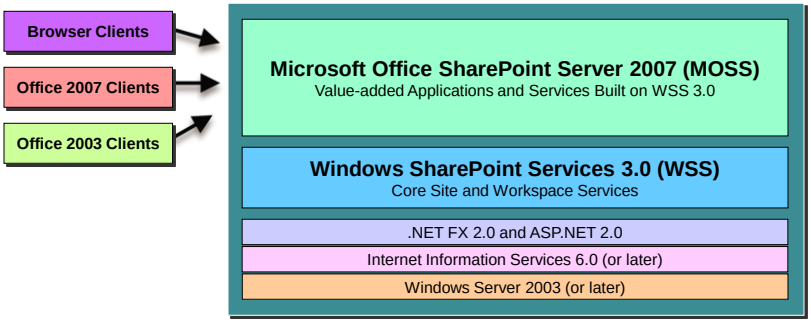
Page Customization using Web Parts

- Web Parts provide page-level customization
  - Users can add Web Parts and modify their properties
  - Web Parts support customization and personalization



Microsoft Office SharePoint Server 2007

- Microsoft Office SharePoint Server 2007 (MOSS)
  - Components and services built on WSS 3.0



MOSS Services and Components

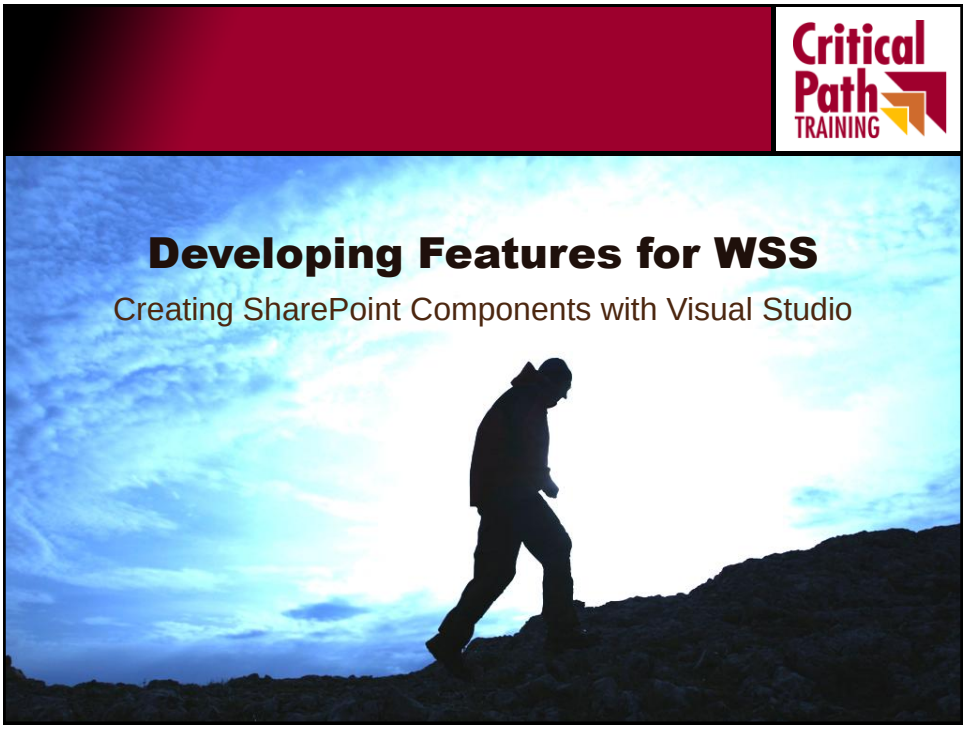
- What does MOSS Standard Edition provide?
  - Next-generation features of SPS 2003 (Portal)
  - Next-generation features of CMS 2002 (WCM)
- What does MOSS Enterprise Edition provide?
  - Forms Services
  - Business Data Catalog
  - Excel Services

Schedule of Lectures

1. Roadmap to WSS Development <<< (you are here)
2. Developing Features
3. SharePoint Architecture
4. Page Design and Provisioning
5. Master Pages and Site Branding
6. Web Part Development
7. AJAX Web Parts
8. Lists and Content Types
9. Document Libraries
10. Introduction to SharePoint Workflows
11. Solutions and Deployment
12. Application Security

Summary

- Architectural overview of SharePoint 2007
  - Windows SharePoint Services 3.0 (WSS)
  - Microsoft Office SharePoint Server 2007 (MOSS)
- WSS as a collaboration solution
- Customizing WSS Sites
- Overview of MOSS components and services



Agenda

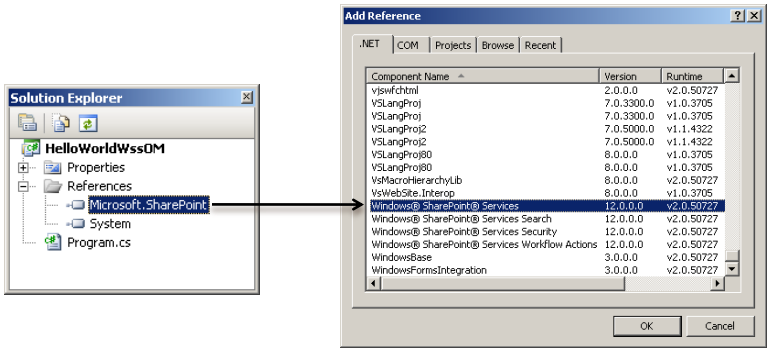
- SharePoint Customization versus Development
- The WSS system directories
- What Are Features?
- Developing a Custom Feature
- Adding Event Handlers to a Feature

Customization Versus Development

- Site Customizations
  - Changes to one particular site
  - Done using the browser or the SharePoint Designer
  - Changes recorded in content database
  - Easy to do but hard to reuse
- WSS Development
  - Creation of reusable templates/components
  - Templates/components installed on Web server
  - Development based on Visual Studio projects
  - Project source files checked into source code control
  - Projects can be moved through staging to production

Hello World: The WSS Object Model

- Create a simple Console Application
  - Add a reference to Microsoft.SharePoint.dll
  - Write the code to access a site and see it's lists



Watch Out: Inconsistent Terminology

New Term	Old Term	WSS Object Model
Site Collection	Site	SPSite
Site	Web	SPWeb

The 'Hello World' Code

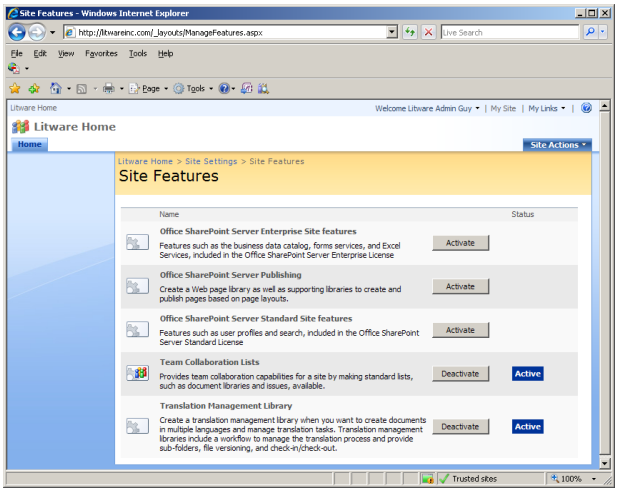
```
using System;
using Microsoft.SharePoint;
namespace Hello_WSS_OM {
  class Program {
    static void Main() {

      string sitePath = "http://litwareinc.com";
      // enter object model through site collection.
      SPSite siteCollection = new SPSite(sitePath);
      // obtain reference to top-level site.
      SPWeb site = siteCollection.RootWeb;
      // enumerate through lists of site
      foreach (SPList list in site.Lists) {
        Console.WriteLine(list.Title);
      }
      // clean up by calling Dispose.
      site.Dispose();
      siteCollection.Dispose();
    }
  }
}
```

What is a Feature?

- A building block for creating SharePoint solutions
  - A unit of design, implementation and deployment
- Features can contain elements
  - e.g. menu items, links, list types and list instances
  - Many other element types possible
- Features can contain event handlers
  - You can add any code which uses the WSS object model

User's View of Features

- Features support the concept of activation/deactivation

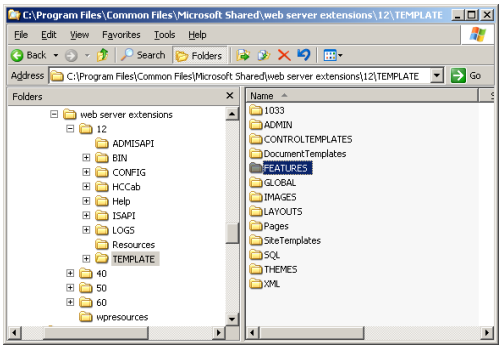


This is the site-level feature management page in a WSS farm where MOSS has been installed.

Much of the functionality of MOSS is enabled and disabled by activating and deactivating features that have been developed by the MOSS team.

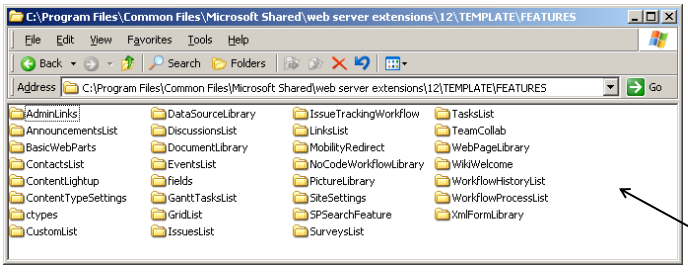
The WSS System Directories

- Developers must learn WSS system directories
  - \12\TEMPLATE
  - \12\TEMPLATE\FEATURES ← This is the one we care about in this lecture
  - \12\TEMPLATE\IMAGES
  - \12\TEMPLATE\LAYOUTS



The Features Directory

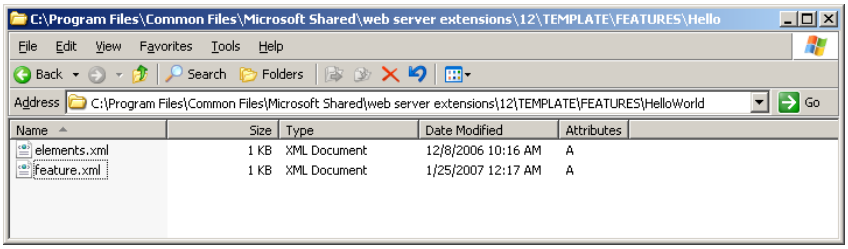
- Functionality in WSS is based on Features
  - Features are installed at farm level
  - Feature activation makes functionality available
  - WSS supports for different feature activation scopes
    (1) Site (2) Site Collection (3) Web Application (4) Farm



Your custom features will each need a directory created here

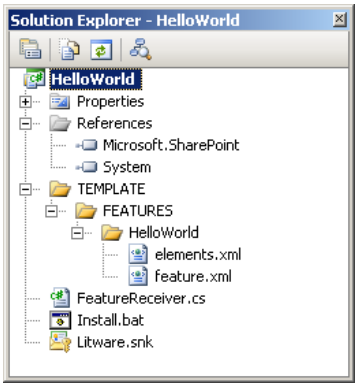
Developer's View of a Feature

- Each feature requires its own directory
  - Directory must contain feature.xml file
  - Directory often contains other file definition elements



Creating the 'Hello World' Feature

- Create a new Visual Studio Class Library project
  - Create XML files which define the feature
  - Add a FeatureActivated event handler



The feature.xml file

- The Feature.xml file serves as the feature manifest
  - Features are defined in declarative fashion using CAML

    CAML = Collaborative Application Markup Language

```
<Feature
  Id="B2CB42E2-4F0A-4380-AABA-1EF9CD526F20"
  Title="A Sample Feature: Hello World"
  Description="Hi mom, class is fun. I am doing great"
  Scope="Web"
  Hidden="FALSE"
  ImageUrl="TPG\WhitePithHelmet.gif"
  xmlns="http://schemas.microsoft.com/sharepoint/">

  <ElementManifests>
    <ElementManifest Location="elements.xml" />
  </ElementManifests>

</Feature>
```


Elements.xml

- Features include elements defined using CAML
  - This element defines a Site Actions menu item
  - There are many other types of elements

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <CustomAction
    Id="SiteActionsToolbar"
    GroupId="SiteActions"
    Location="Microsoft.SharePoint.StandardMenu"
    Sequence="100"
    Title="Hello World"
    Description="A custom menu item added using a feature"
    ImageUrl="_layouts/images/crtsite.gif" >

      <UrlAction Url="http://msdn.microsoft.com"/>

  </CustomAction>

</Elements>
```

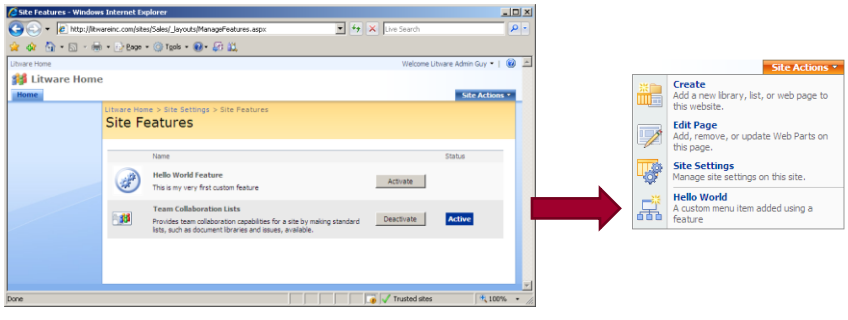
Install.bat

- Visual Studio supports post-build events
  - Can be used to run batch files to deploy components
  - Used on development machines
  - Should not be used on staging/production machines

```
@SET TEMPLATEDIR="c:\program files\common files\microsoft shared\web server extensions\12\Template"
@SET STSADM="c:\program files\common files\microsoft shared\web server extensions\12\bin\stsadm"
@SET GACUTIL="c:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\gacutil.exe"

Echo Installing HelloWorld.dll in GAC
%GACUTIL% -if bin\debug\HelloWorld.dll

Echo Copying files to TEMPLATE directory
xcopy /e /y TEMPLATE\* %TEMPLATEDIR%

Echo Installing feature
%STSADM% -o installfeature -filename  HelloWorld\feature.xml -force

Echo Restart all IIS worker processes
IISRESET

Echo Restart just the IIS worker process for a particular Application Pool
REM C:\WINDOWS\system32\inetsrv\appcmd.exe recycle APPPOOL "SharePointDefaultAppPool"
```

Feature Activation

- Steps to testing Features
  - Copy Feature files to FEATURES directory
  - Install feature with WSS
  - Activate Feature within a specific site



Feature event handlers

- FeatureReceiver class provides 4 event handlers

```
using System;
using Microsoft.SharePoint;
namespace HelloWorld {
  public class FeatureReceiver : SPFeatureReceiver {
    // no functionality required for install/uninstall events
    public override void FeatureInstalled(SPFeatureReceiverProperties properties) { }
    public override void FeatureUninstalling(SPFeatureReceiverProperties properties) { }

    public override void FeatureActivated(SPFeatureReceiverProperties properties) {
      SPWeb site = (SPWeb)properties.Feature.Parent;
      // track original site Title using SPWeb property bag
      site.Properties["OriginalTitle"] = site.Title;
      site.Properties.Update();
      // update site title
      site.Title = "Hello World";
      site.Update();
    }

    public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {
      // reset site Title back to its original value
      SPWeb site = (SPWeb)properties.Feature.Parent;
      site.Title = site.Properties["OriginalTitle"];
      site.Update();
    }
  }
}
```

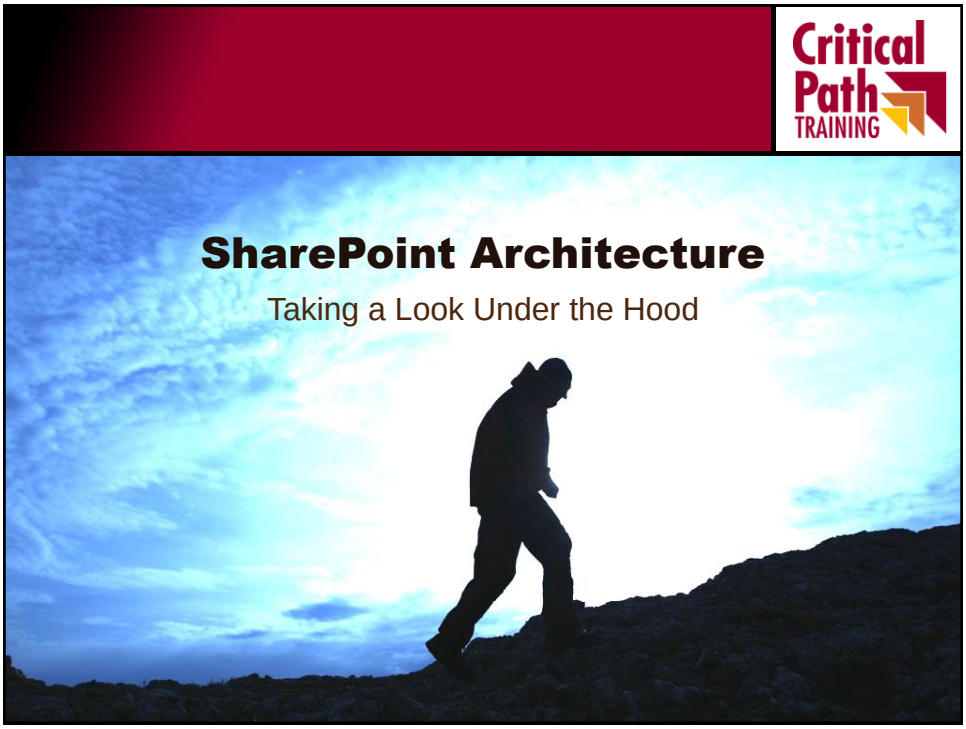
The feature.xml file revisited

- The Feature.xml file serves as the feature manifest

```
<Feature
  Id="B2CB42E2-4F0A-4380-AABA-1EF9CD526F20"
  Title="A Sample Feature: Hello World"
  Description="Hi mom, class is fun. I am doing great"
  Scope="Web"
  Hidden="FALSE"
  ImageUrl="TPG\WhitePithHelmet.gif"
  ReceiverAssembly="HelloWorld, [full 4-part assembly name]"
  ReceiverClass="HelloWorld.FeatureReceiver"
  xmlns="http://schemas.microsoft.com/sharepoint/">

  <ElementManifests>
    <ElementManifest Location="elements.xml" />
  </ElementManifests>

</Feature>
```

Summary

- SharePoint Customization versus Development
- The WSS system directories
- What Are Features?
- Developing a Custom Feature
- Adding Event Handlers to a Feature

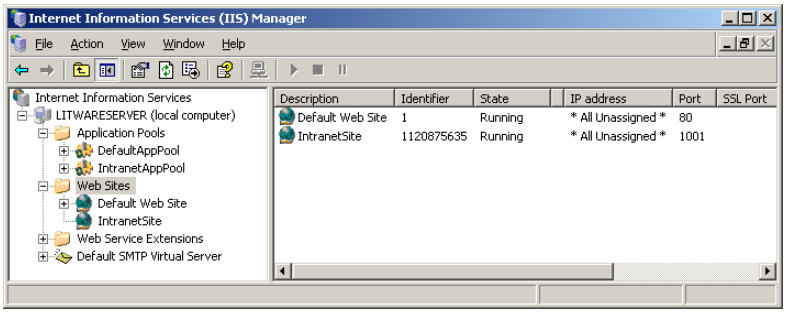


Agenda

- WSS Integration with ASP.NET 2.0
  - IIS Web sites and Web Applications
  - The farm and the configuration database
  - Web Application and Content Database
- Content Databases
- The web.config file
- Site pages versus Application pages
- Creating custom Application pages
- Deployment using Solution Packages

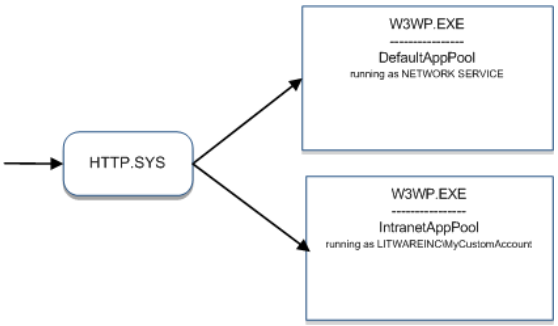
IIS Web Sites

- WSS depends on IIS Web sites for…
  - HTTP listener mechanism
  - Process management through Application Pools
  - Security and user authentication



IIS Application Pools

- IIS dispatches requests to Application Pools
  - Each Application Pool is configured to run in its own process
  - IIS lets you configure Application Pool identity
  - App Pool identity can be a local or domain account



The ASP.NET Framework

- ASP.NET is a productivity framework on top of IIS
  - Integrated with IIS via the ISAPI extension (aspnet_isapi.dll)
  - Provides abstractions such as page, request, response
  - Integrates with Visual Studio and managed code

The web.config file

- Provides configuration for ASP.NET runtime

```
<configuration>
  <system.web>

    <customErrors mode="On" />

    <httpRuntime maxRequestLength="51200" />

    <authentication mode="Windows" />

    <identity impersonate="true" />

    <authorization>
      <allow users="*" />
    </authorization>

  </system.web>
</configuration>
```

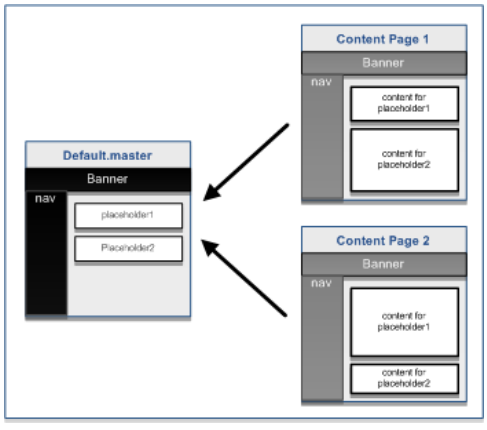
ASP.NET Pages

- ASP.NET development is typically based on pages
  - Pages are deployed as .ASPX files to Web server
  - .ASPX files are parsed and compiled on the first request
  - Compiled page class inherits from **System.Web.UI.Page**

```
<%@ Page Language="C#" %>
<script runat="server">
  protected override void OnLoad(EventArgs e)  {
    lblDisplay.Text = "Hello, ASP.NET";
  }
</script>

<html>
<body>
  <form id="frmMain" runat="server">
    <asp:Label runat="server" ID="lblDisplay" />
  </form>
</body>
</html>
```

Master Pages in ASP.NET

- ASP.NET 2.0 introduces Master Pages
  - Defines common layouts used across content pages



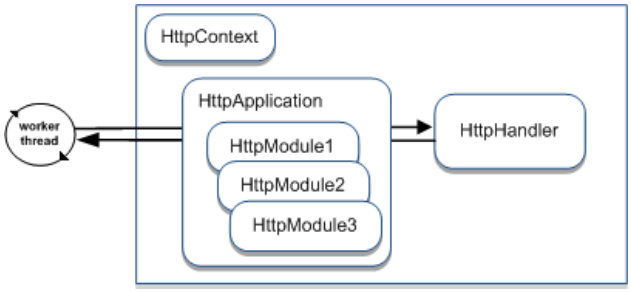
Linking Content Page to Master Page

```
<!-- default.master -->
<%@ Master %>
<html><body>
  <form id="frmMain" runat="server">
    <table width="100%">
      <tr><td> <h1>Litware Inc.</h1><hr /></td></tr>
      <tr>
        <td> <!-- Display Main Body of Page -->
          <asp:contentplaceholder id="PlaceHolderMain" runat="server" />
        </td>
      </tr>
    </table>
  </form>
</body></html>
```

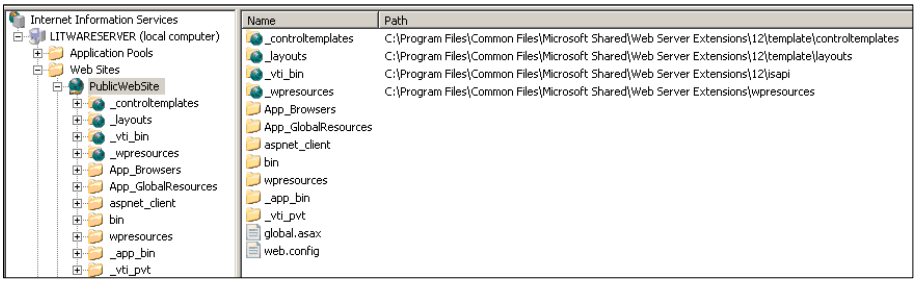
```
<!-- content page linking to default.master -->
<%@ Page Language="C#" MasterPageFile="~/default.master" Title="Page 1" %>

<asp:Content ID="Main" ContentPlaceHolderID="PlaceHolderMain">
  Unique page content goes here
</asp:Content>
```

The HTTP Pipeline of ASP.NET

- ASP.NET processing is based on the HTTP pipeline
  - HttpApplication and HttpModule act as interceptors
  - HttpHandler acts as the endpoint for request
  - All object types can be replaced with custom code
  - HttpContext object is available anywhere in pipeline



The WSS-extended Web Application



- Web Applications extend IIS and ASP.NET
  - IIS wildcard application map sends all requests to ASP.NET
  - ASP.NET extended using common objects inside HTTP pipeline
  - Web Application is configured with WSS system virtual directories

    _layouts
    _controltemplates
    _vti_bin
    _wpresources

The WSS-extended web.config file

- **WSS replaces HttpApplication object**

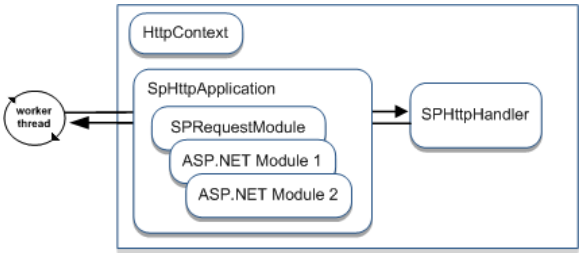
```
<!-- global.asax file at root of WSS Web Application -->
<@Application Inherits="Microsoft.SharePoint.ApplicationRuntime.SPHttpApplication" >
```

- **WSS configures pipeline with its own HttpHandler and HttpModule**

```
<!-- web.config file at root of WSS Web Application -->
<configuration>
  <system.web>
    <httpHandlers>
      <remove verb="GET,HEAD,POST" path="*" />
      <add verb="GET,HEAD,POST" path="*"
         type="Microsoft.SharePoint.ApplicationRuntime.SPHttpHandler,..." />
    </httpHandlers>
    <httpModules>
      <clear />
      <add name="SPRequest"
         type="Microsoft.SharePoint.ApplicationRuntime.SPRequestModule,..." />
      <!-- other standard ASP.NET httpModules added back in -->
    </httpModules>
  </system.web>
</configuration>
```

WSS Web Applications

- WSS extends HTTP pipeline with custom objects
  - Configuration added to every WSS Web Application
  - Modifications made to web.config file and IIS metabase



- Different and superior architecture than WSS 2.0
  - WSS 2.0 architecture based on problematic ISAPI filter

WSS Extensions to the web.config file

```
<configuration>

  <configSections>
    <sectionGroup name="SharePoint">
      <section name="SafeControls" type="..." />
      <section name="RuntimeFilter" type="..." />
      <section name="WebPartLimits" type="..." />
      <section name="WebPartCache" type="..." />
      <section name="WebPartWorkItem" type="..." />
      <section name="WebPartControls" type="..." />
      <section name="SafeMode" type="..." />
      <section name="MergedActions" type="..." />
      <section name="PeoplePickerWildcards" type="..." />
    </sectionGroup>
  </configSections>

  <SharePoint>
    <SafeMode />
    <WebPartLimits />
    <WebPartCache />
    <WebPartControls />
    <SafeControls />
    <PeoplePickerWildcards />
  </SharePoint>

</configuration>
```

Important Debugging Settings

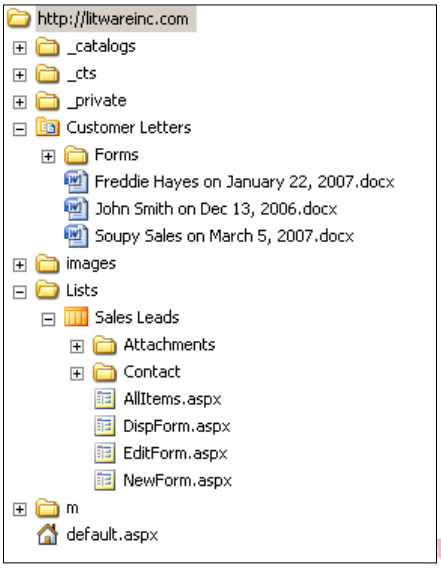
```
<configuration>
  <configSections>...
  <SharePoint>
    <SafeMode MaxControls="200" CallStack="false" DirectFileDependencies="10"
      <PageParserPaths>
      </PageParserPaths>
    </SafeMode>
    <WebPartLimits MaxZoneParts="50" PropertySize="1048576" />
    <WebPartCache Storage="CacheObject" />
    <WebPartControls DatasheetControlGuid="65BCBEE4-7728-41a0-97BE-14E1CAE36A
    <SafeControls>...
    <PeoplePickerWildcards>
      <clear />
      <add key="AspNetSqlMembershipProvider" value="%" />
    </PeoplePickerWildcards>
    <MergedActions>...
    <BlobCache location="C:\blobCache" path="\.(gif|jpg|png|css|js)$" maxSize=
    <RuntimeFilter Assembly="Microsoft.Office.Server, Version=12.0.0.0, Cultu
  </SharePoint>
  <system.web>
    <securityPolicy>...
    <httpHandlers>...
    <customErrors mode="On" />
    <httpRuntime maxRequestLength="51200" />
```

set to **true**

set to **Off**

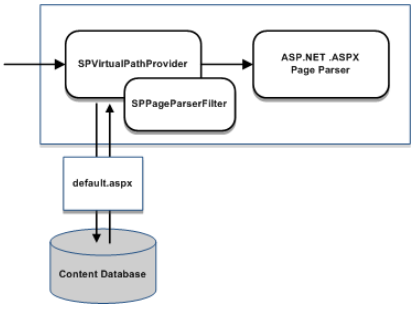
The Virtual File System of a Site

- Site is a virtual file system
  - made up of folders and files
  - Pages are files
  - Documents are files
  - Stored in content database
- How can you look at it?
  - SharePoint Designer
  - Windows Explorer (WebDav)



Processing Pages within a Site

- WSS stores.aspx files in content database
  - Retrieved using SPVirtualPathProvider object
  - Page based on page templates on Web server
  - Non-customized pages can be ghosted
  - Customized pages cannot be ghosted



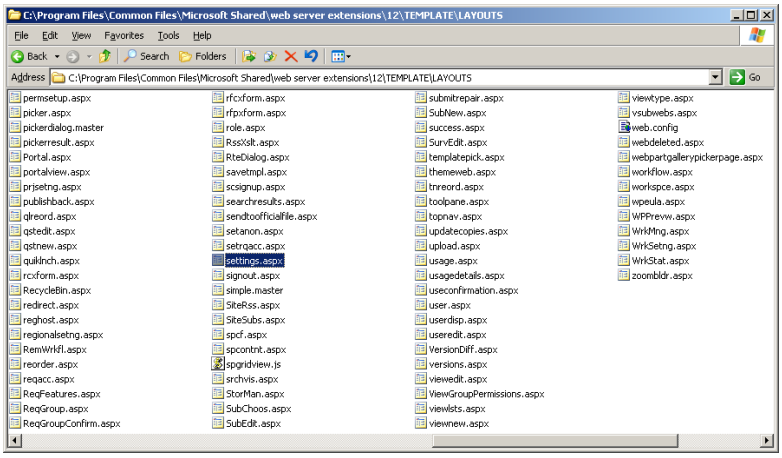
The _layouts Virtual Directory

- Files in `_layouts` directory accessible to all sites
  - _layouts provides access to common resources
  - _layouts contains files for images, CSS and JavaScript
  - _layouts contains Application Pages
- All these URLS resolve to the same page

```
http://Litwareinc.com/_layouts/settings.aspx
http://Litwareinc.com/sites/Vendors/_layouts/settings.aspx
http://Litwareinc.com:1001/sites/Accounting/_layouts/settings.aspx
```

Application Pages

- Standard Application Pages are part of WSS

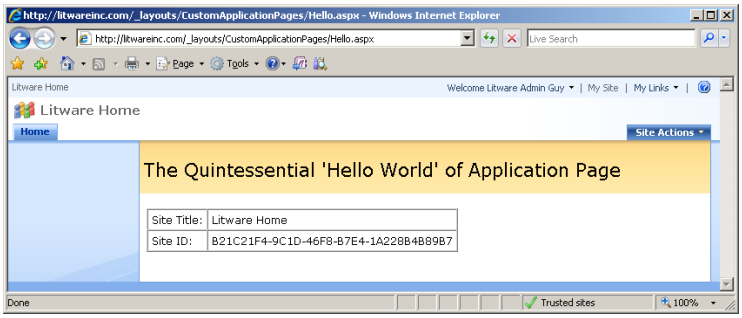


Site Pages Versus Application Pages

- Site Pages exist within virtual file system of site
  - They may or may not be ghosted
  - They support customization via Web Parts
  - They support customization via SharePoint Designer
  - Customized pages impact performance and security
- Application Pages are deployed once per farm
  - They do not support customization or Web Parts
  - They are parsed/compiled as classic ASP.NET pages
  - They run faster than Site Pages
  - They always support code behind

Creating Custom Application Pages

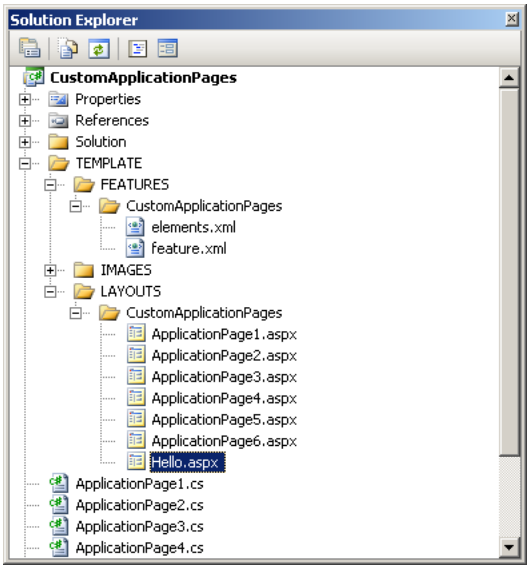
- Steps to creating a custom Application Page
  - Inherit from LayoutsPageBase
  - Link to application.master
  - Add server-side controls and code
  - Deploy to LAYOUTS directory



'Hello World' Custom Application Page

```
<%@ Assembly Name="Microsoft.SharePoint, [full 4-part name]"%>

<%@ Page Language="C#" MasterPageFile="~/_layouts/application.master"
        Inherits="Microsoft.SharePoint.WebControls.LayoutsPageBase"  %>

<%@ Import Namespace="Microsoft.SharePoint" %>

<script runat="server">
  protected override void OnLoad(EventArgs e) {
    // SPWeb site = SPContext.Current.Web;
    SPWeb site = this.Web; // base class provides access to WSS objects
    lblSiteTitle.Text = site.Title;
    lblSiteID.Text = site.ID.ToString().ToUpper();
  }
</script>

<asp:Content ID="Main" contentplaceholderid="PlaceHolderMain" runat="server">
  Site Title: <asp:Label ID="lblSiteTitle" runat="server"/><br />
  Site ID: <asp:Label ID="lblSiteID" runat="server" />
</asp:Content>

<asp:Content ID="PageTitleInTitleArea" runat="server"
             contentplaceholderid="PlaceHolderPageTitleInTitleArea" >
        The Quintessential 'Hello World' of Application Page
</asp:Content>
```

Demo: CustomApplicationPages



Adding a Feature for Navigation

- Features can be used with custom applications
  - Custom actions provide navigation menu items

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- Add Menu Command to Site Actions Dropdown -->
  <CustomAction Id="HelloApplicationPage"
                GroupId="SiteActions"
                Location="Microsoft.SharePoint.StandardMenu"
                Sequence="2000"
                Title="Hello World Application Page"
                Description="Getting up and going with inline code">

    <UrlAction Url="~site/_layouts/CustomApplicationPages/Hello.aspx"/>

  </CustomAction>
</Elements>
```

Adding an ECB Menu Item

- Custom ECB menu items can be added to lists
  - Redirect to application page
- Registration Types
  - List
  - Content Type
  - File Extension


```
<CustomAction
  Id="CustomApplicationPage4"
  RegistrationType="List"
  RegistrationId="101"
  ImageUrl="/_layouts/images/GORTL.GIF"
  Location="EditControlBlock"
  Sequence="240"
  Title="Application Page 4" >
 <UrlAction Url="~site/_layouts/CustomApplicationPages/
          ApplicationPage4.aspx?ItemId={ItemId}&amp;ListId={ListId}"/>
</CustomAction>
```

Deployment using Solution Packages

- Evolution of Web Part Packages from WSS 2.0
  - Solution Package is a CAB file with a .wsp extension
  - Solution Package contains a manifest
  - Solution Package contains files required on Web server
- What can be deployed via a Solution Package
  - Feature definitions
  - Application Pages
  - Assembly DLLs
  - And much more…

Deployment using Solution Packages

- WSS Deployment is done with Solution Packages
  - Solution Package is CAB file with .wsp extension
  - Created using DDF file and MAKECAB.EXE
  - Deployed using STSADM.EXE or WSS Central Admin

CustomApplicationPages.dll	Application Extension	
manifest.xml	XML Document	
elements.xml	XML Document	CustomApplicationPages\
feature.xml	XML Document	CustomApplicationPages\
ApplicationPage1.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
ApplicationPage2.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
ApplicationPage3.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
ApplicationPage4.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
ApplicationPage5.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
ApplicationPage6.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\
Hello.aspx	ASP.NET Server Page	LAYOUTS\CustomApplicationPages\

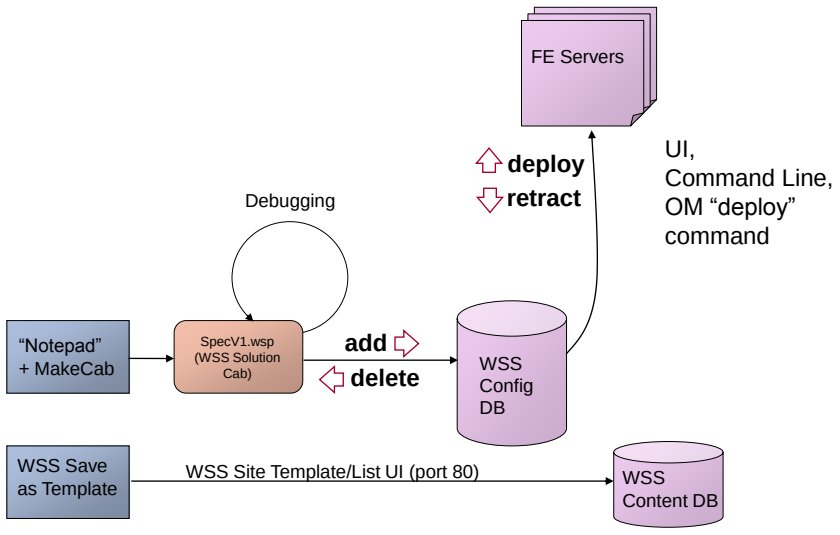
CustomApplicationPages.wsp

Solution Package Manifest

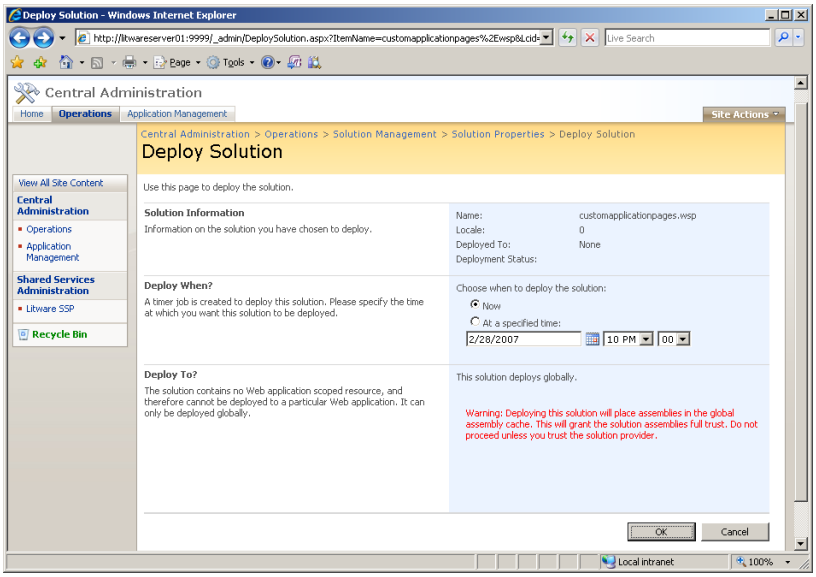
- Solution Manifest is read by WSS installer

```
<Solution SolutionId="9EFFE92B-781D-4c99-BBCC-432D248B899D"
                  xmlns="http://schemas.microsoft.com/sharepoint/">

  <FeatureManifests>
    <FeatureManifest Location="CustomApplicationPages\feature.xml" />
  </FeatureManifests>

  <TemplateFiles>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\Hello.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage1.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage2.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage3.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage4.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage5.aspx"/>
    <TemplateFile Location="LAYOUTS\CustomApplicationPages\ApplicationPage6.aspx"/>
  </TemplateFiles>

  <Assemblies>
    <Assembly Location="CustomApplicationPages.dll"
              DeploymentTarget="GlobalAssemblyCache" />
  </Assemblies>

</Solution>
```

Solution And Template Deployment

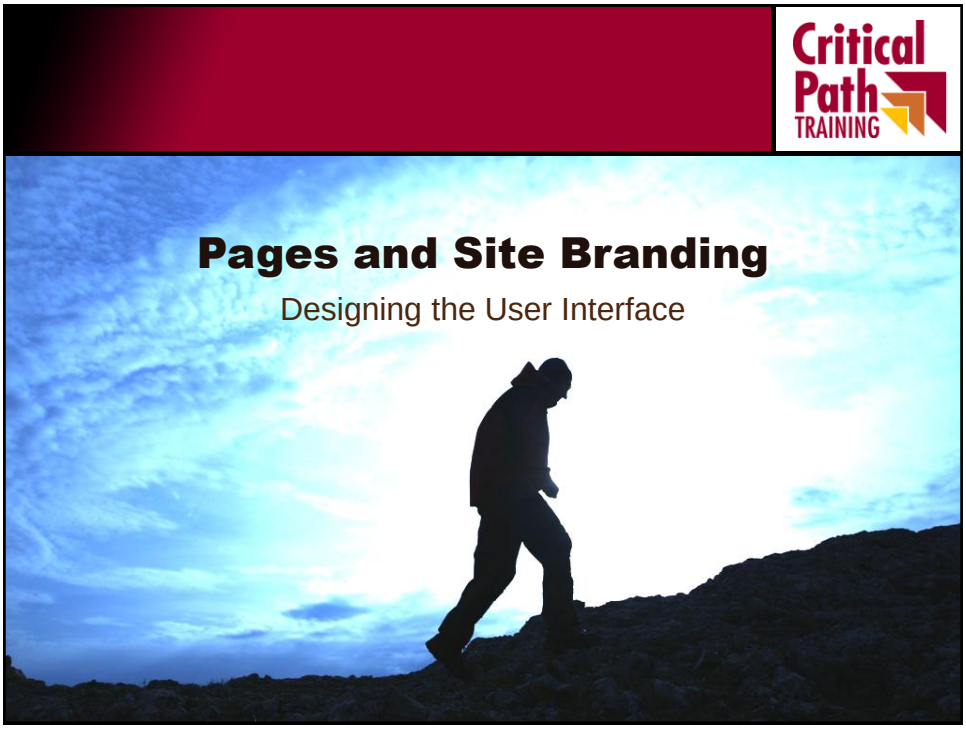


Deploying Solution Packages



Summary

- WSS Integration with ASP.NET 2.0
  - IIS Web sites and Web Applications
  - The farm and the configuration database
  - Web Application and Content Database
- Content Databases
- The web.config file
- Site pages versus Application pages
- Creating custom Application pages
- Deployment using Solution Packages

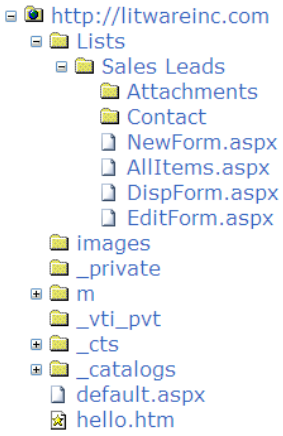


Agenda

- Page parsing and Safe Mode restrictions
- Creating custom page templates
- Designing Web Part Pages
- Master Pages
- Branding a site collection with a custom feature
- Understanding and extending core.css

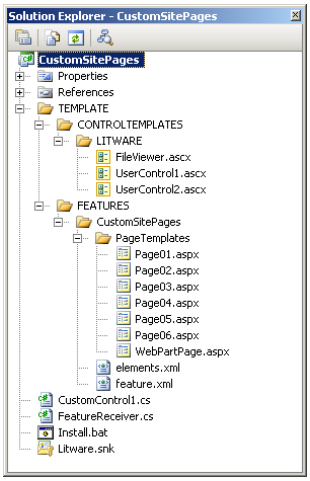
Site Page Fundamentals

- Site Pages are part of a site
  - Represented with SPFile objects
  - Structured in SPFolder objects



Demo: CustomSitePages

- Important Concepts
  - Page template vs. page instance
  - Page customization
  - SafeMode processing



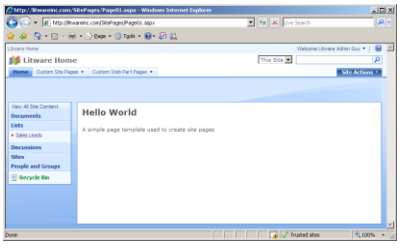
'Hello World' Page Template

- A Page Template can be added to a feature
  - MasterPageFile points to **~masterurl/default.master**
  - **progid** adds support for SharePoint Designer

```
<%@ Page MasterPageFile="~masterurl/default.master"
    meta:progid="SharePoint.WebPartPage.Document"  %>

<asp:Content runat="server" ContentPlaceHolderID="PlaceHolderMain">

  <h3>Hello World</h3>

  A simple page template used to create site pages

</asp:Content>
```

Provisioning a Page Instance

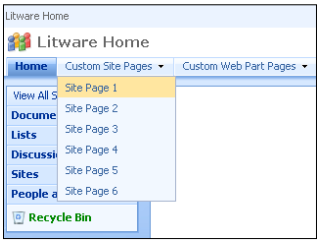
- A Module element is used to provision a page instance
  - One File element per page instance
  - Supports page ghosting




```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <Module Path="PageTemplates" Url="SitePages" >
    <File Url="Page01.aspx" Type="Ghostable" />
  </Module>

</Elements>
```

Adding Navigation Support for Pages

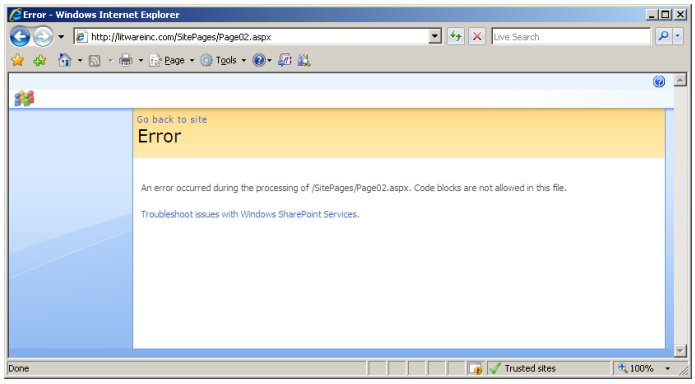
- Navigation nodes can be added
  - Can be added during feature activation
  - Can be added to top-link bar
  - Can be added to QuickLaunch
  - Nodes created as SPNavigationNode




```
public class FeatureReceiver : SPFeatureReceiver {
  public override void FeatureActivated(SPFeatureReceiverProperties properties) {
    // get a hold off current site in context of feature activation
    SPWeb site = (SPWeb)properties.Feature.Parent;
    SPNavigationNodeCollection topNav = site.Navigation.TopNavigationBar;

    // create dropdown menu for custom site pages
    SPNavigationNode DropDownMenu1 =
                    new SPNavigationNode("Custom Site Pages", "", false);
    topNav[0].Children.AddAsLast(DropDownMenu1);
    DropDownMenu1.Children.AddAsLast(
      new SPNavigationNode("Site Page 1", "SitePages/Page01.aspx"));
  }
}
```

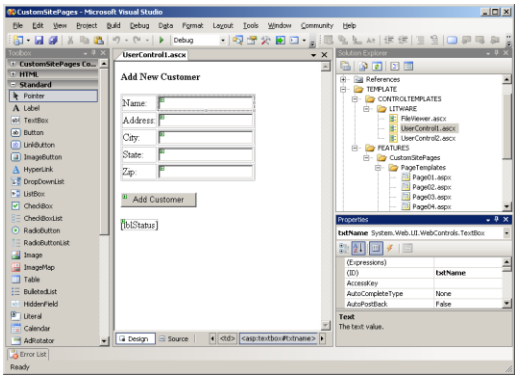
Safe Mode Processing

- Customized site pages run in SafeMode
  - They do not support inline code
  - They only support controls registered as SafeControls



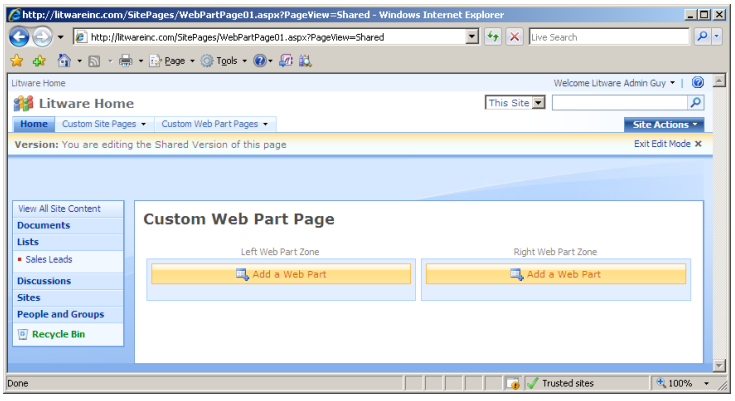
Designing Pages with Controls

- Two kinds of ASP.NET controls
  - Custom controls
  - User controls



Designing Web Part Pages

- Creating a Web Part Page template
  - Inherit from WebPartPage
  - Add one or more Web Part Zones



```
<%@ Page Language="C#" MasterPageFile="~masterurl/default.master"
         Inherits="Microsoft.SharePoint.WebPartPages.WebPartPage,[asm name]"
         meta:progid="SharePoint.WebPartPage.Document"  %>

<%@ Register Tagprefix="WebPartPages"
             Namespace="Microsoft.SharePoint.WebPartPages"
             Assembly="Microsoft.SharePoint, [asm name]" %>

<asp:Content ID="main" runat="server" ContentPlaceHolderID="PlaceHolderMain"
>

<h3>Custom Web Part Page</h3>

<table width="100%">
  <tr>
    <td valign="top" style="width:50%">
        <WebPartPages:WebPartZone ID="Left" runat="server"
                                  FrameType="TitleBarOnly"
                                  Title="Left Web Part Zone" />
    </td>
    <td valign="top" style="width:50%">
        <WebPartPages:WebPartZone ID="Right" runat="server"
                                  FrameType="TitleBarOnly"
                                  Title="Right Web Part Zone" />
    </td>
  </tr>
</table>

</asp:Content>
```

Adding Web Parts into Zones

- Web Parts can be pre-populated into zones
  - Can be done declaratively through CAML
  - Can be done programmatically through WSS OM

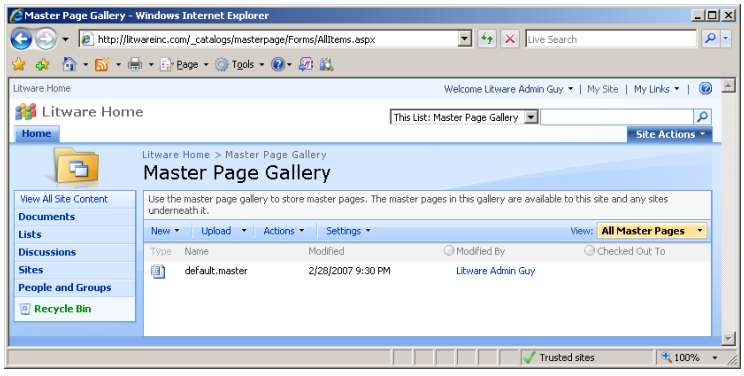
```
<File Url="WebPartPage.aspx" Name="WebPartPage03.aspx" Type="Ghostable" >
  <!-- Add a Web Part to right zone -->
  <AllUsersWebPart WebPartZoneID="Right" WebPartOrder="0">
    <![CDATA[
       <WebPart xmlns="http://schemas.microsoft.com/WebPart/v2"
                xmlns:iwp="http://schemas.microsoft.com/WebPart/v2/Image">
         <Assembly>Microsoft.SharePoint, [asm name]</Assembly>
         <TypeName>Microsoft.SharePoint.WebPartPages.ImageWebPart</TypeName>
         <FrameType>None</FrameType>
         <Title>Watch My Gears Run</Title>
         <iwp:ImageLink>/_layouts/images/GEARS_AN.GIF</iwp:ImageLink>
       </WebPart>
    ]]>
  </AllUsersWebPart>
</File>
```

Master Pages in WSS

- Application pages use application.master
  - Farm-wide master page
  - Cannot be customized on a per-site basis
- Site Pages use default.master by default
  - default.master is a page template
  - default.master instance is created in Master Page Gallery
  - default.master can be customized on a per-site basis
  - default.master can be replaced with a different template

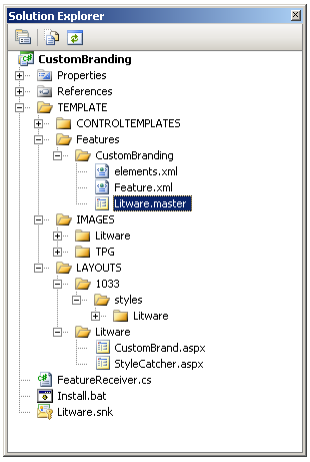
The Master Page Gallery

- Each site has a Master Page Gallery
  - Instance of default.master automatically provisioned
  - default.master can be customized on a per-site basis



Demo: CustomBranding

- Important Concepts
  - Custom Master Page Templates
  - Custom CSS File
  - Custom Site Logo



Custom Master Page Templates

- Creating a Master Page Template
  - Use default.master as a starting point
  - Make changes to suit your tastes
- Master Page templates are like site page templates
  - Support ghosting and unghosting
  - Provisioned using a File element within a Module

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="MasterPages" List="116" Url="_catalogs/masterpage">
    <File Url="Litware.master" Type="GhostableInLibrary" />
  </Module>
</Elements>
```

Master Page Elements

```
<%@Master language="C#"%>
<%@ Register Tagprefix="SharePoint"
    Namespace="Microsoft.SharePoint.WebControls"
    Assembly="Microsoft.SharePoint, …" %>

<HTML id="HTML1" runat="server">
<HEAD id="HEAD1" runat="server">

  <!-- SharePoint Utility Controls -->
  <SharePoint:CssLink ID="CssLink1" runat="server"/>
  <SharePoint:Theme ID="Theme1" runat="server"/>

  <!-- Named Placeholders -->
  <Title ID=onetidTitle>
    <asp:ContentPlaceHolder id=PlaceHolderPageTitle runat="server"/>
  </Title>
  <asp:ContentPlaceHolder id="PlaceHolderAdditionalPageHead" runat="server"/>

    <!-- Named Delegate Control -->
    <SharePoint:DelegateControl
      ID="DelegateControl1" runat="server"
      ControlId="AdditionalPageHead" AllowMultipleControls="true"/>

</HEAD>
```

Updating the MasterUrl Property

- Update MasterUrl to redirect site pages
  - A Child site can reference the Master Page in a top-level site

```
public partial class _Default : System.Web.UI.Page {
  protected void cmdApplyCustomBrand_Click(object sender, EventArgs e) {
  SPWeb site = SPContext.Current.Site.RootWeb
  string MasterUrlPath = site.ServerRelativeUrl;
  if (!MasterUrlPath.EndsWith(@"/"))
    MasterUrlPath += @"/";
  MasterUrlPath += @"_catalogs/masterpage/Litware.master";
  ApplyCustomBrand(MasterUrlPath, site);
}

protected void ApplyCustomBrand(string MasterUrlPath, SPWeb site) {
  site.MasterUrl = MasterUrlPath;
  site.Update();
  // use recusion to update all child sites in site collection
  foreach (SPWeb child in site.Webs) {
    ApplyCustomBrand(MasterUrlPath, child);
  }
}
```

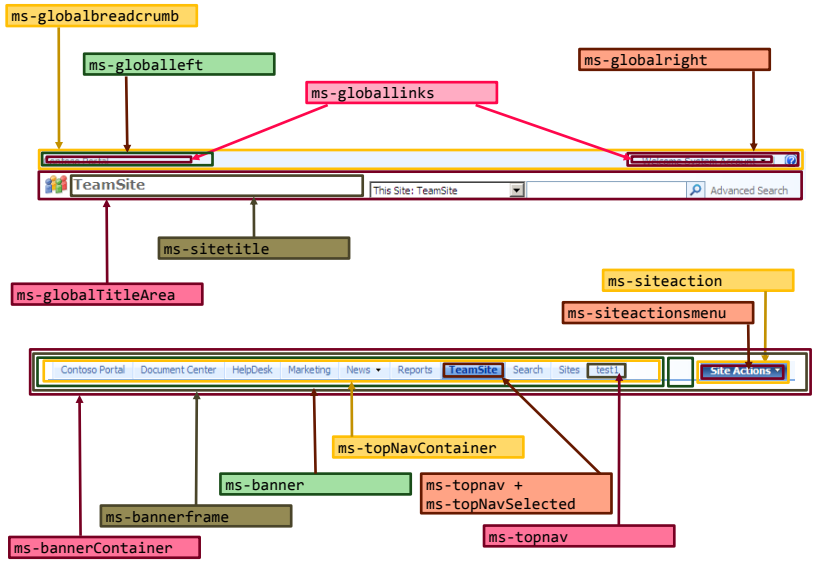
Understanding core.css

- All styles in WSS initially defined by core.css
  - Located in \TEMPLATE\LAYOUTS\1033\STYLES
  - Contains over 4000 lines of CSS class definitions
  - Classes used throughout standard WSS UI elements
- Extending core.css
  - Applying WSS styles (meant for end users)
  - Applying custom CSS files (meant for developers)

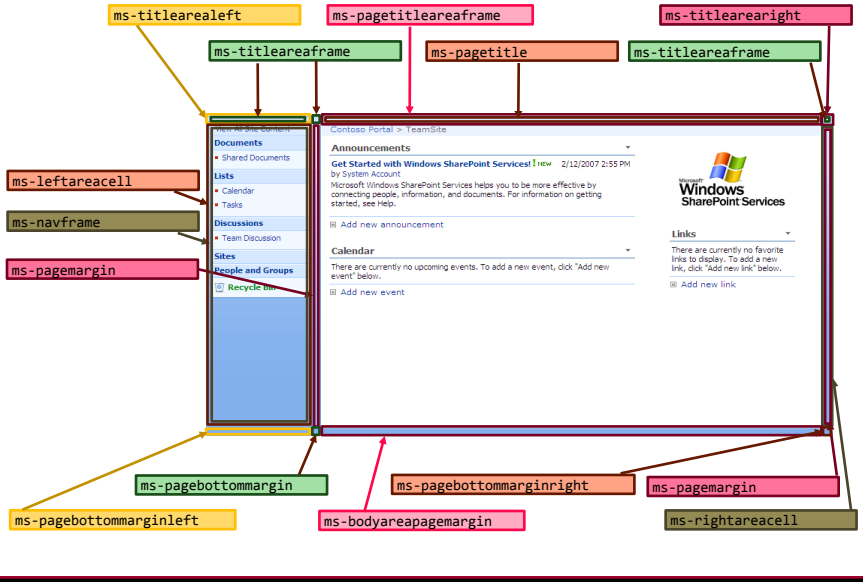
Applying Custom Branding

```
protected void cmdApplyCustomBrand_Click(object sender, EventArgs e) {

  SPWeb site = SPContext.Current.Web;

  string MasterUrlPath = site.ServerRelativeUrl;
  if (!MasterUrlPath.EndsWith(@"/"))
    MasterUrlPath += @"/";
  MasterUrlPath += @"_catalogs/masterpage/Litware.master";
  ApplyCustomBrand(MasterUrlPath, site);

  Response.Redirect(Request.RawUrl);
}

protected void ApplyCustomBrand(string MasterUrlPath, SPWeb site) {
  site.ApplyTheme("");
  site.MasterUrl = MasterUrlPath;
  site.AlternateCssUrl = "/_layouts/1033/STYLES/Litware/LitwareBrand.css";
  site.SiteLogoUrl = "/_layouts/images/Litware/LitwareFullLogo.png";
  site.Update();

  foreach (SPWeb child in site.Webs) {
    ApplyCustomBrand(MasterUrlPath, child);
  }
}
```

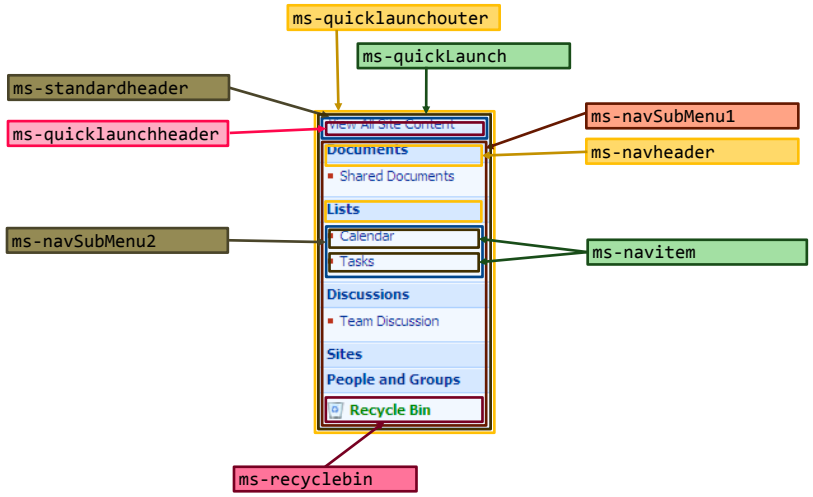
Top Navigation Areas



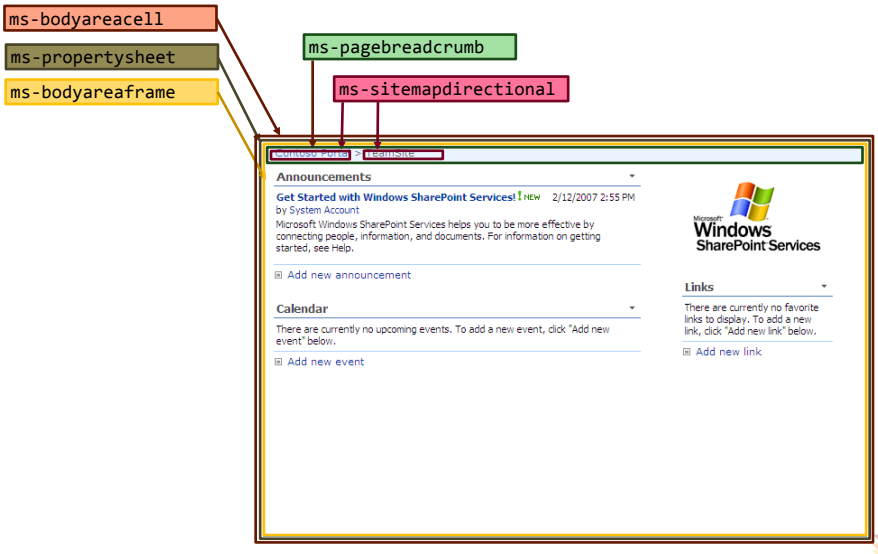
Body Structure



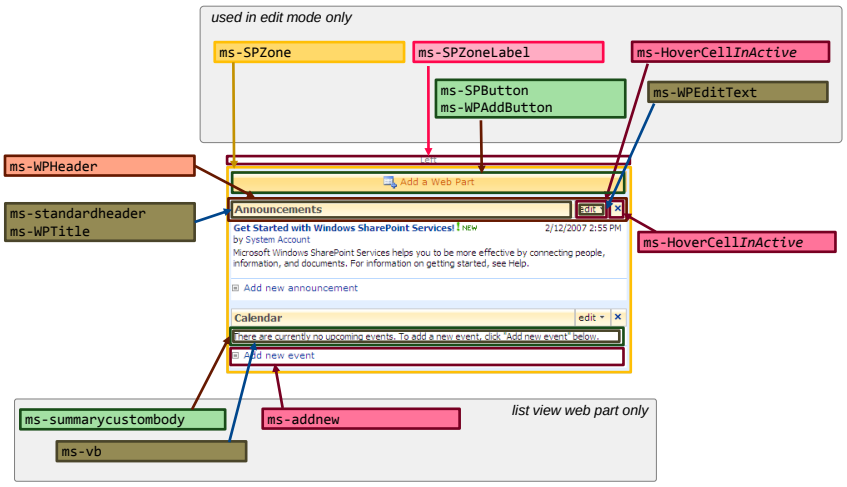
Quick Launch



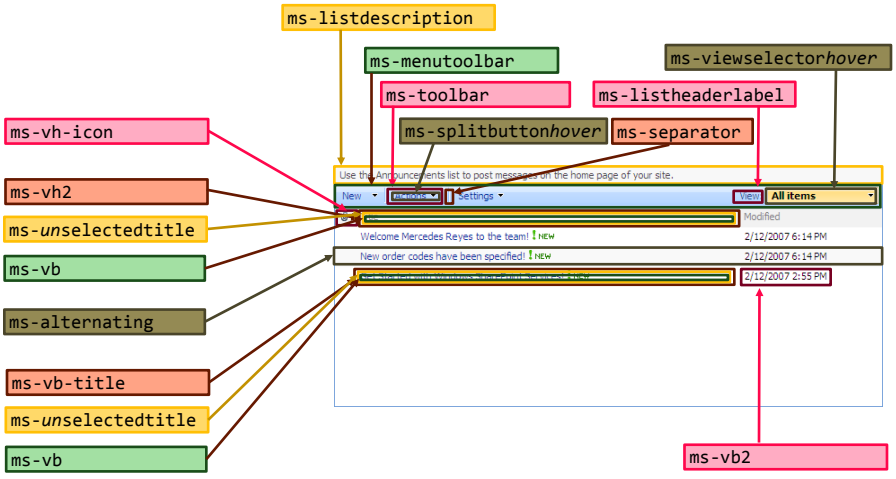
Page Body



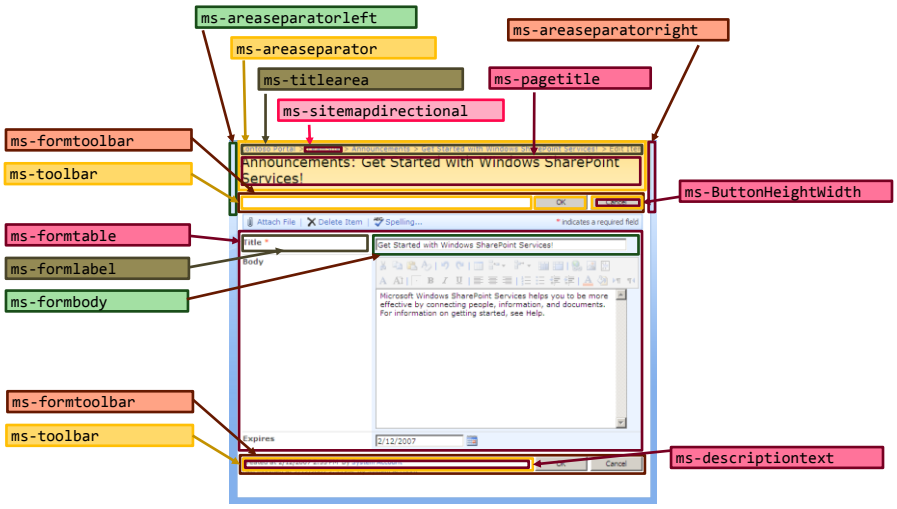
Web Parts



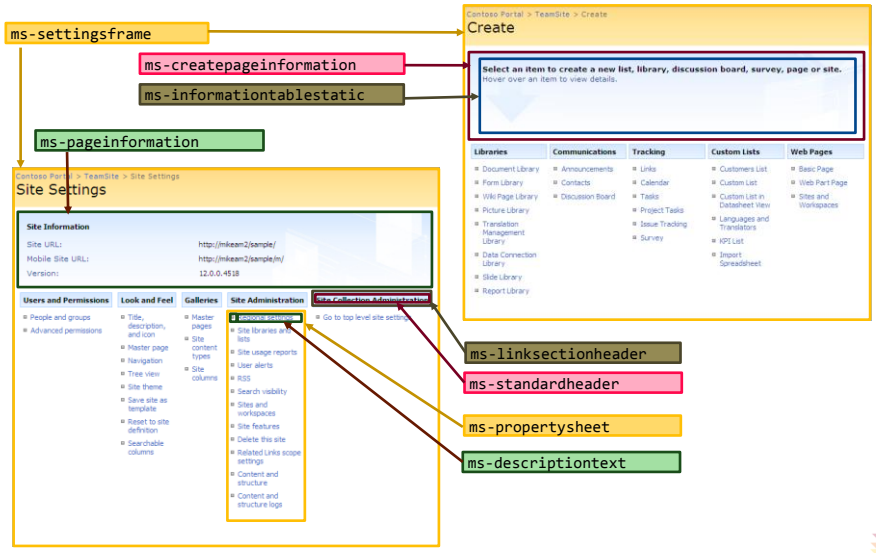
List Views



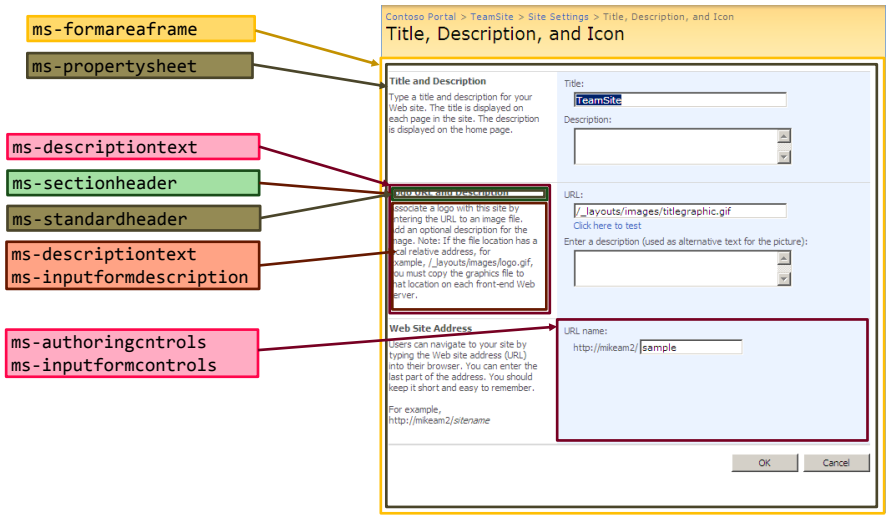
Forms



Application Pages (Landing Pages)



Application Pages (Form Pages)



Summary

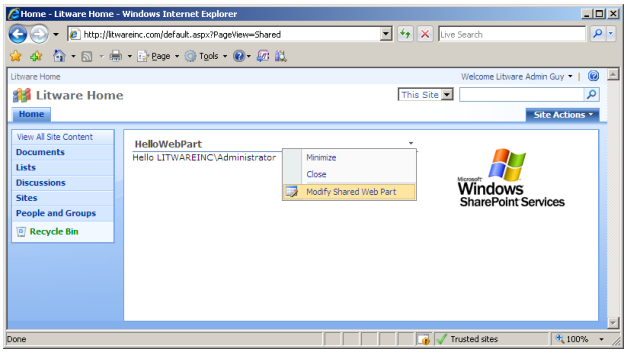
- Page parsing and Safe Mode restrictions
- Creating custom page templates
- Designing Web Part Pages
- Master Pages
- Branding a site collection with a custom feature
- Understanding and extending core.css

Agenda

- Developing ASP.NET Web Parts for WSS 3.0
- Persistent Web Part properties
- Importing Web Parts into the Web Part Gallery
- Creating a feature for deploying Web Parts
- Advanced Web Part Techniques

Web Parts

- Web Parts are used to build portal-style applications
  - Content is modular, consistent and easy to navigate
  - Configurable chrome: border and title bar
  - Web Parts support customization and personalization

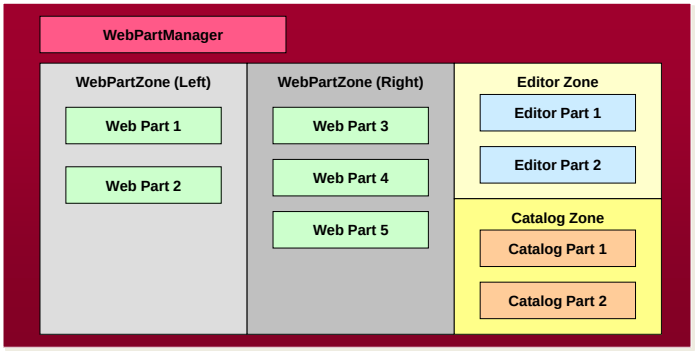


Web Part History

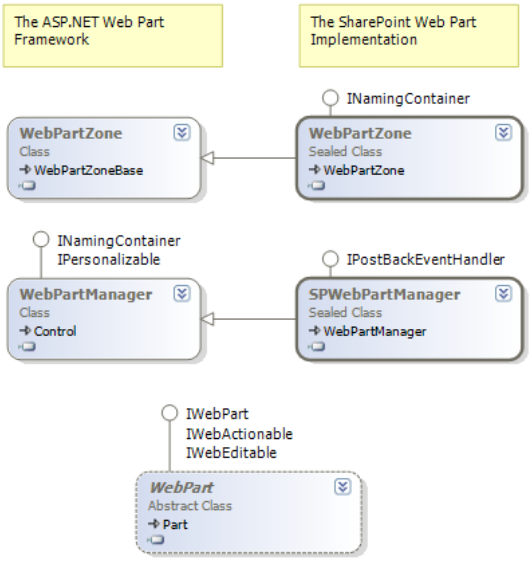
- Windows SharePoint Services 2.0 (WSS V2)
  - Designed with its own Web Part infrastructure
  - WSS serializes/stores/retrieves personalization data
- ASP.NET 2.0
  - Designed with a newer universal Web Part infrastructure
  - Serializes/stores/retrieves personalization data
  - More flexible and more extensible than WSS
  - ASP.NET 2.0 does not support WSS v2 Web Parts
- Windows SharePoint Services 2007 (WSS V3)
  - Supports WSS V2 style Web Parts
  - Supports ASP.NET 2.0 style Web Parts (preferred)

ASP.NET Web Part Page Structure

- Web Part Page in ASP.NET 2.0
  - One instance of the WebPartManager class
  - One or more Web Part Zones
  - Optionally an Editor Zone and/or a Catalog Zone

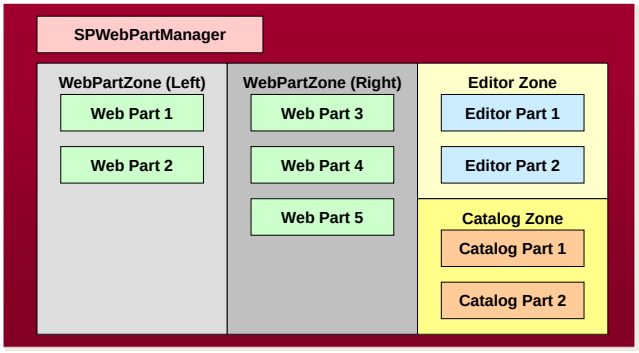


SharePoint's Web Part Implementation



WSS Web Part Page Structure

- Web Part Pages in WSS
  - Inherits from the WSS WebPartPage base class
  - Contains one SPWebPartManager control
  - Contains one or more WSS WebPartZone controls



Overview of Developing Web Parts

1. Create a new class library DLL project
   - Create class that inherits from ASP.NET Web Part class
   - Override methods as required (e.g. RenderContents)
2. Deploy Web Part DLL
   - Compile DLL into \bin directory
   - Configure DLL in web.config file SafeControl list
3. Import Web Part into a WSS site collection
   - Add Web Part class to Web Part Gallery
   - Add Web Part to zone on a Web Part Page

ASP.NET 2.0 Web Parts

- Web Parts derive from the WebPart base class
  - All Web Parts inherit common functionally

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls.WebParts;

namespace LitwareWebParts {

  public class HelloWorld : WebPart {

    protected override void RenderContents(HtmlTextWriter writer) {
      writer.Write("Hello, world");
    }
  }
}
```


Persistent Web Part Properties

- Web Parts support persistent properties
  - Customization data is seen by all users
  - Personalization data is seen only by one user

```
namespace LitwareWebParts {
  public class HelloWorld : WebPart {

    protected string _ZipCode;

    [ Personalizable(), WebBrowsable(true),
      WebDisplayName("Zip Code"),
      WebDescription("used to track user zip code")  ]
    public string ZipCode {
      get{ return _ZipCode; }
      set{ _ZipCode = value; }
    }
  //...
  }
}
```

Web Parts As A Safe Control

- Web Parts usually run on Web Part Pages
  - Web Parts must be registered as Safe in the web.config file
  - You must add entry to web.config before testing

```
<!- web.config in Web Application root directory -->

<configuration>
  <SharePoint>
    <SafeControls>
      <SafeControl Assembly="AcmeWebParts"
                   Namespace="AcmeWebParts"
                   TypeName="*"
                   Safe="True" />
    </SafeControls>
  </SharePoint>
</configuration>
```


Web Part Security Caveats

- Web Parts in \bin are subject to security restrictions
  - Security restrictions from Code Access Security (CAS)
  - You might want to turn off security during development
- You can choose between three built-in levels

```
WSS_Minimum (default for WSS V3)
WSS_Medium
Full
```

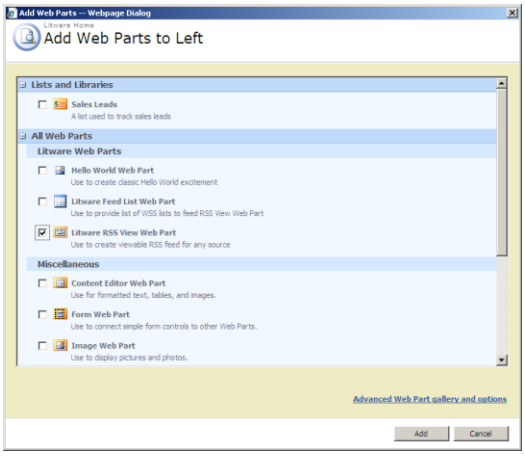
```
<!- web.config -->
<configuration>
  <system.web>
    <!-- <trust level="WSS_Minimal" originUrl="" /> -->
    <trust level="Full" originUrl="" />
  </system.web>
</configuration>
```

The Web Part Gallery (WPG)

- The WPG is scoped at the Site Collection level
  - Contains a list of Web Parts available to place on pages
  - Contains .webpart files and .dwp files

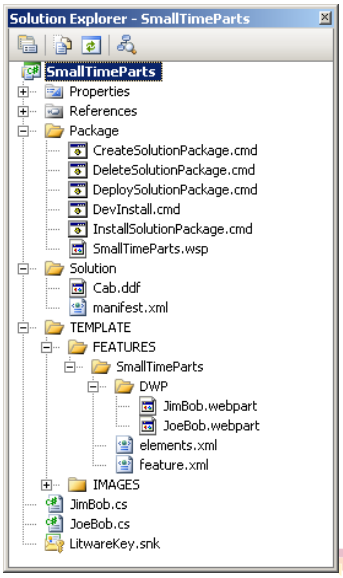
Adding Web Parts from the Gallery

- WSS provides standard dialog for adding parts



Demo: SmallTimeParts

- Important Concepts
  - .webpart files
  - Web Part Deployment Feature
  - Web Part Solution Package
  - Custom CAS Settings



Provisioning .webpart files

- .webpart files need to be included with WP deployment feature

```
<webParts>
  <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
    <metaData>
      <type name="SmallTimeParts.JimBob, SmallTimeParts, [full 4-part assembly name]" />
      <importErrorMessage>Cannot import this Web Part.</importErrorMessage>
    </metaData>
    <data>
      <properties>
        <!-- standard Web Part properties -->
        <property name="ChromeType" type="chrometype">Default</property>
        <property name="Title" type="string">Jim Bob's Web Part</property>
        <property name="Description" type="string">Some valuable description goes here</property>
      </properties>
    </data>
  </webPart>
</webParts>
```

- Modules are then used to provision .webpart files into the Web Part Gallery

```
<!-- this module goes in the feature used to deploy your Web Parts -->
<Module Name="SmallTimeParts" List="113" Url="_catalogs/wp" Path="dwp" RootWebOnly="true">

  <File Url="JimBob.webpart" Type="GhostableInLibrary" >
    <Property Name="Group" Value="A Set of Smalltime Web Parts" />
  </File>

</Module>
```

Solution Manifest for WP Deployment

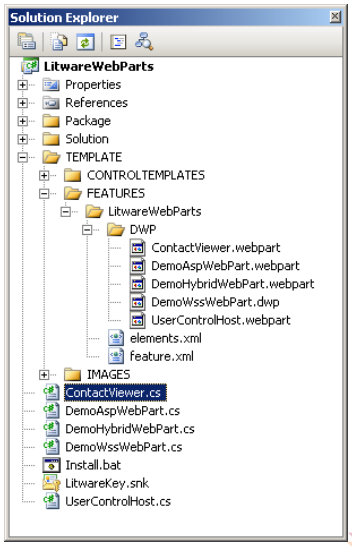
```
<Solution
  SolutionId="DEADBEEF-BADD-BADD-BADD-BADBADBADBAD"
  xmlns="http://schemas.microsoft.com/sharepoint/">

  <FeatureManifests>
    <FeatureManifest Location="SmallTimeParts\feature.xml" />
  </FeatureManifests>

  <TemplateFiles>
    <TemplateFile Location="IMAGES\TPG\compass.gif"/>
    <TemplateFile Location="IMAGES\TPG\SmallCompass.gif"/>
    <TemplateFile Location="IMAGES\TPG\SmallBinoculars.gif"/>
  </TemplateFiles>

  <Assemblies>
    <Assembly DeploymentTarget="WebApplication"   Location="SmallTimeParts.dll">
      <SafeControls>
        <SafeControl Assembly="SmallTimeParts, [full 4-part assembly name]"
                     Namespace="SmallTimeParts" TypeName="*" Safe="True"/>
      </SafeControls>
    </Assembly>
  </Assemblies>

  <CodeAccessSecurity>
    <!-- use when custom CAS policy is needed for deployment in \bin -->
  </CodeAccessSecurity>

</Solution>
```

Solution Manifest for WP Deployment

```
<Solution SolutionId="DEADBEEF-BADD-BADD-BADD-BADBADBADBAD"
          xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- other solution elements omitted for clairty -->
  <CodeAccessSecurity>

    <PolicyItem>
      <!-- create permission set for this policy -->
      <PermissionSet class="NamedPermissionSet" version="1"
                     Description="Permission set for SmallTimeParts assembly">
        <!-- add generic .NET CAS security permission -->
        <IPermission class="SecurityPermission" version="1"
                     Flags="Execution, UnmanagedCode, ControlThread" />

        <!-- add ASP.NET hosting permission -->
        <IPermission class="AspNetHostingPermission" version="1" Level="High" />

        <!-- add SharePoint permission -->
        <IPermission class="Microsoft.SharePoint.Security.SharePointPermission, [asm name]"
             version="1" ObjectModel="true" Impersonate="true" UnsafeSaveOnGet="true" />
      </PermissionSet>
       <!-- add assembly to be associated with this policy -->
      <Assemblies>
        <Assembly Name="SmallTimeParts"  />
      </Assemblies>
    </PolicyItem>

  </CodeAccessSecurity>
</Solution>
```

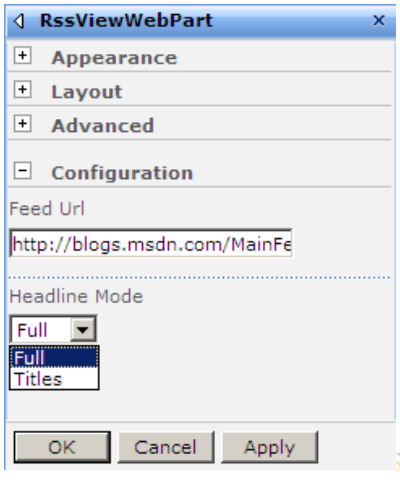
Demo: LitwareWebParts

- Important Concepts
  - Editor Parts
  - Web Part Verbs
  - Web Part Connections
  - Asynchronous Processing



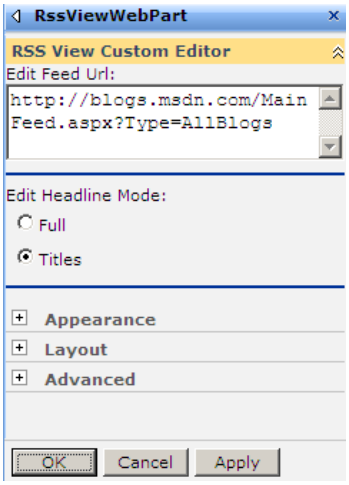
Standard Editor Parts

- WSS provides standard editor parts



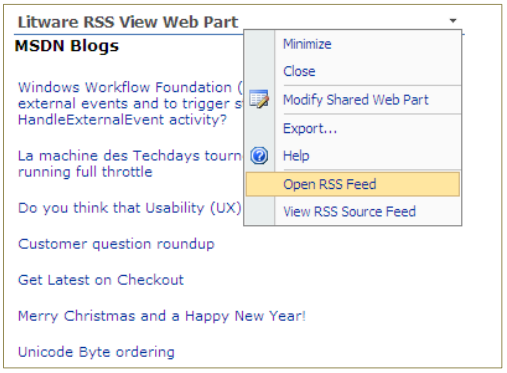
A Custom Editor Part

- Custom Editor Parts provide more control
  - Control over rendering
  - Control over validation



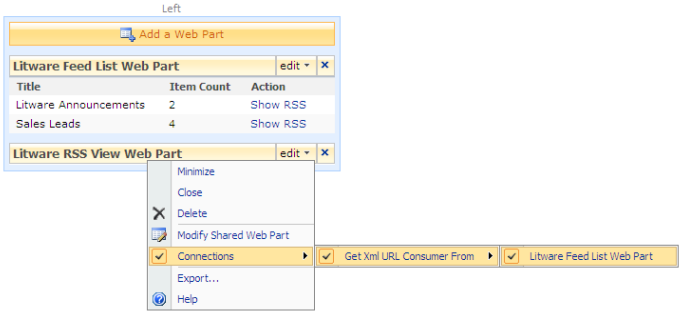
Web Part Verbs

- Used to add menu items to Web Parts
  - Supports client-side handlers through JavaScript
  - Supports server-side handlers through managed code



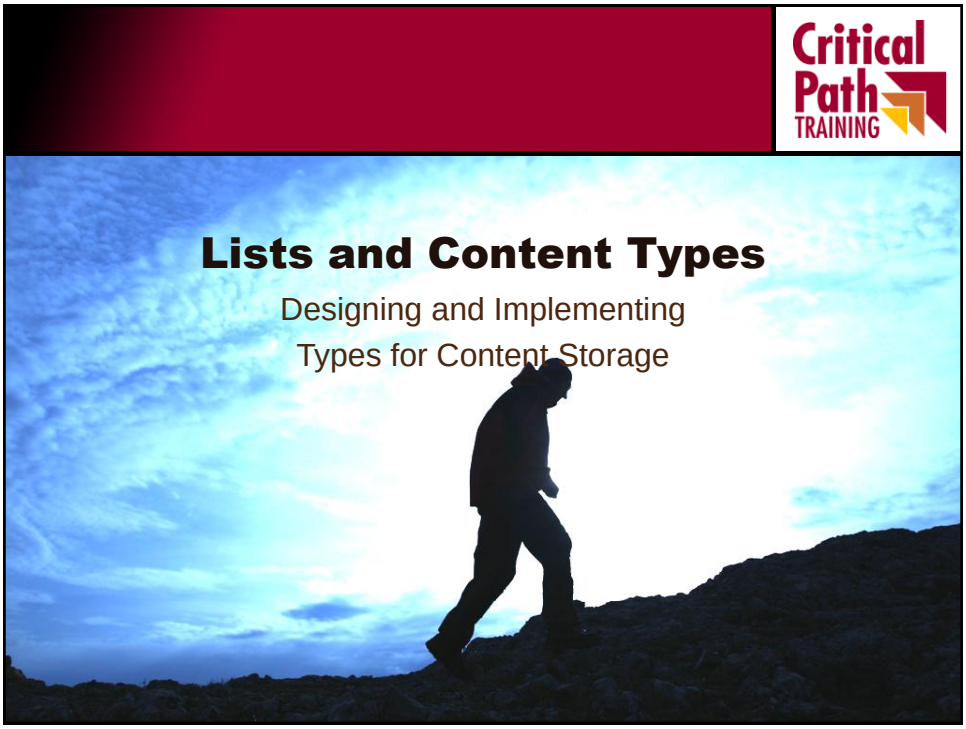
Web Part Connections

- ASP.NET provides a Web Part Connection model
  - Provider Web Parts supply data
  - Consumer Web Parts retrieve data
  - WSS provides UI elements to establish connections



Summary

- Developing ASP.NET Web Parts for WSS 3.0
- Persistent Web Part properties
- Importing Web Parts into the Web Part Gallery
- Creating a feature for deploying Web Parts
- Advanced Web Part Techniques



Agenda

- Content storage enhancements in WSS 3.0
- Querying data in lists
- WSS storage fundamentals
  - Site columns
  - Custom field types
  - Content types
- Provisioning lists and document libraries
- Event handling with receiver classes

Motivation: Content Storage in WSS

- All storage in WSS is based on the concept of lists
  - Everything is modeled in terms of rows and columns
  - The Document Library is really just a hybrid list
- WSS adds value on top of the generic list concept
  - Transparent content storage in SQL Server
  - Automatic generation of the user interface

Platform Storage Enhancements

- Parity between lists and document libraries
  - Folders are supported for lists as well as document libraries
  - Versioning is supported for list items as well as documents
  - Events are supported on lists as well as in document libraries
- List and Document Library Enhancements
  - New productivity-oriented built-in field types
  - Wide list support allowing 100s of columns (e.g. surveys)
  - Custom column indexing to improve performance
  - Cross web queries, list views and lookup fields
  - Enhanced versioning with major and minor versions
  - Lists and document libraries automatically support RSS feeds

Accessing List Data

- Updating list data

```
SPListItem newItem = list.Items.Add();
newItem["Title"] = "Litware Goes Public!";
newItem["Body"] = " We all live in exciting times.";
newItem["Expires"] = DateTime.Now + TimeSpan.FromDays(2);
newItem.Update();
```

- Enumerating through list items

```
foreach (SPListItem item in list.Items) {
  foreach (SPField field in list.Fields) {
    if (field.Hidden != true && !field.ReadOnlyField)
      Console.WriteLine("{0} = {1}", field.Title, item[field.Id]);
  }
}
```


SPQuery

- SPQuery supports CAML-based queries
  - Faster access than enumerating through all list items
  - Limited to a single list per query

```
SPQuery query = new SPQuery();
query.ViewFields = @"<FieldRef Name='Title'/><FieldRef Name='Expires'/>";
query.Query =
@"<Where>
    <Lt>
      <FieldRef Name='Expires' />
      <Value Type='DateTime'>
        <Today />
      </Value>
    </Lt>
  </Where>";

SPList list = site.Lists["Litware News"];
SPListItemCollection items = list.GetItems(query);
foreach (SPListItem expiredItem in items) {
  Console.WriteLine(expiredItem["Title"]);
}
```

SPSiteDataQuery

- SPSiteDataQuery can extend across lists/sites
  - Introduced in WSS 3.0
  - Scope can be Site, SiteCollection or Recursive

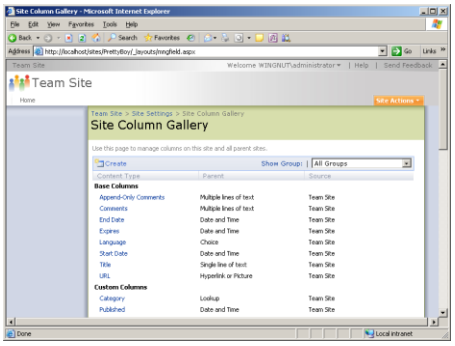
```
SPSiteDataQuery query = new SPSiteDataQuery();
query.Lists = @"<Lists ServerTemplate='104' />";
query.ViewFields = @"<FieldRef Name='Title'/><FieldRef Name='Created'/>";
query.Webs = "<Webs Scope='SiteCollection' />";
string queryText =
@"<Where>
   <Eq>
     <FieldRef Name='Created' />
     <Value Type=""DateTime"">
       <Today />
     </Value>
   </Eq>
 </Where>";
query.Query = queryText;
DataTable table = site.GetSiteData(query);
foreach (DataRow row in table.Rows) {
  Console.WriteLine(row["Title"].ToString()); }
```

Issues with Managing Content

- Problems with managing content in large companies
  - There are many document types identified in an organization, but there is no clear way to enforce standards
  - There's a need to create different types of documents and store them all in one central location
  - Content management applications should make a list of actions available to users depending on the type of content or document
- WSS provides new features to solve these problems
  - Site Columns
  - Content Types

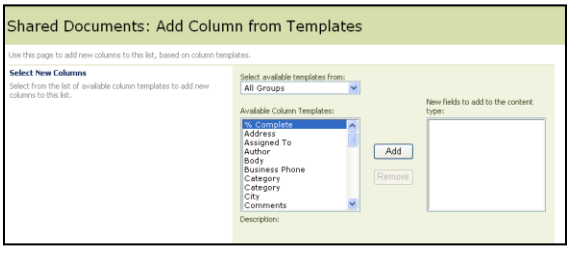
Site Columns

- Site columns are reusable column definitions
  - Site columns can be reused across multiple lists
  - Site columns are scoped to site in the Site Column Gallery
  - Site columns are visible within the site collection to all child sites

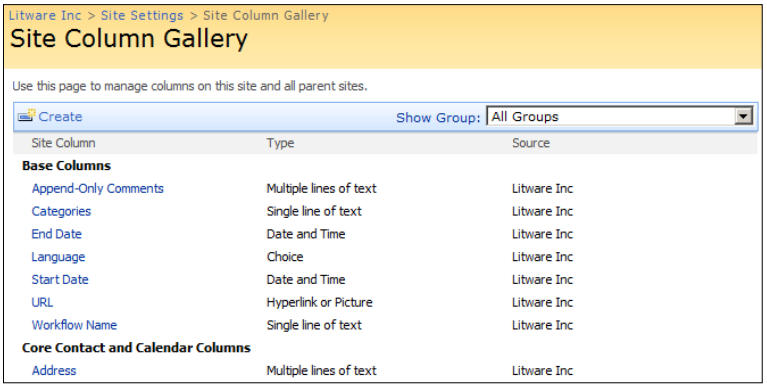


Using a Site Column in a List

- Site Columns can be used in List Definitions
  - A Site column represents a reusable, named column definition
  - Site columns are used in lists, document libraries or content types
  - Updates to a site column can optionally be pushed out to lists, document libraries and content types where it has been used



Demo: Creating Site Columns



Introduction to Content Types

- Foundation for content management in WSS v3
  - Reusable definition for list schema
  - Defines constraints and requirements for an item type
  - Created by users and developers
  - Reused and extended by users

Examples for Content Types

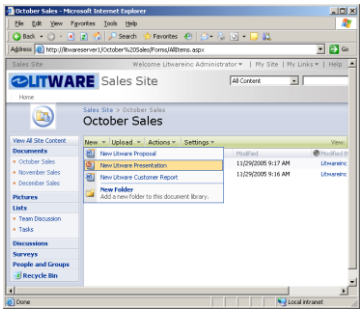
- Content type for Proposals
  - Requires string column for tracking clients
  - Requires boolean column for legal review
- Content Type for Presentations
  - Requires string column for tracking client
  - Requires boolean column for art review

Content Types

- A content type definition can include…
  - Columns to represent metadata or properties
  - A document template for creating new documents
  - Custom forms for New, Edit, and Display
  - Event handlers
  - Workflows

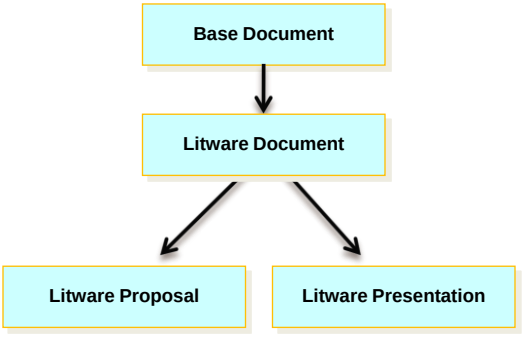
Supporting Multiple Content Types

- Lists can support multiple content types
  - Makes it possible to support heterogeneous content
  - The “New button” becomes a dropdown list
  - Input and display forms change depending on content type

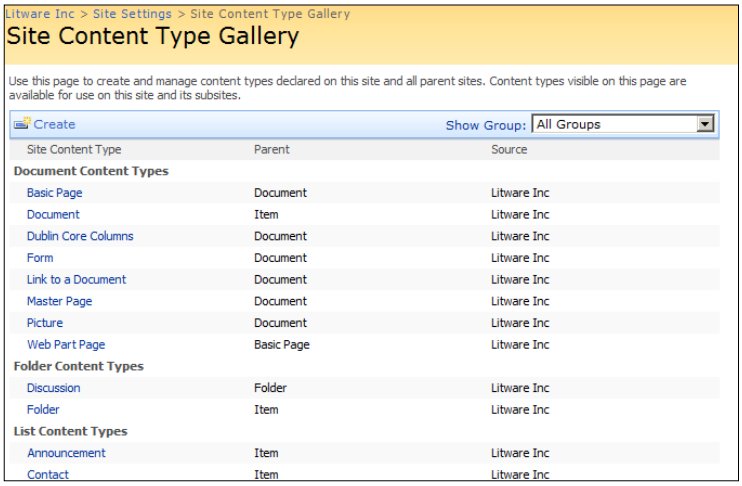


Inheriting Content Types

- Allows base definition reuse across multiple types
  - Core properties can be defined in base content types
  - The Base content type is inherited by more specific content types

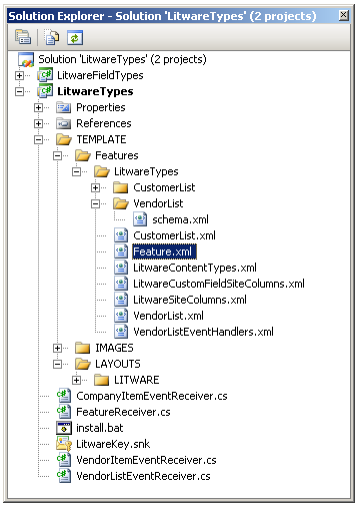


Demo: Creating Content Types



Demo: LitwareTypes

- Important Concepts
  - Defining WSS types in features using CAML
  - Defining site columns
  - Custom field types
  - Defining content types
  - Defining list schemas
  - Creating event handlers



WSS 3.0 Events

- Events architecture has significantly improved
  - Events are supported for lists, document libraries and content types
  - Events are supported for changes to list schema as well as items
  - Events are supported at site collection and site level
  - Events are supported for incoming email messages
  - Support for synchronous events and asynchronous events
  - Synchronous events occur before the fact and are cancel-able
- How do you get events to work
  - Create a custom class inheriting a WSS receiver class
    e.g. SPItemEventReceiver or SPWebEventReciever
  - Compile class into assembly DLL and install in GAC
  - Add event configuration by installing and activating a feature

Item-level Events

Define the receiver class by inheriting from SPItemEventReceiver

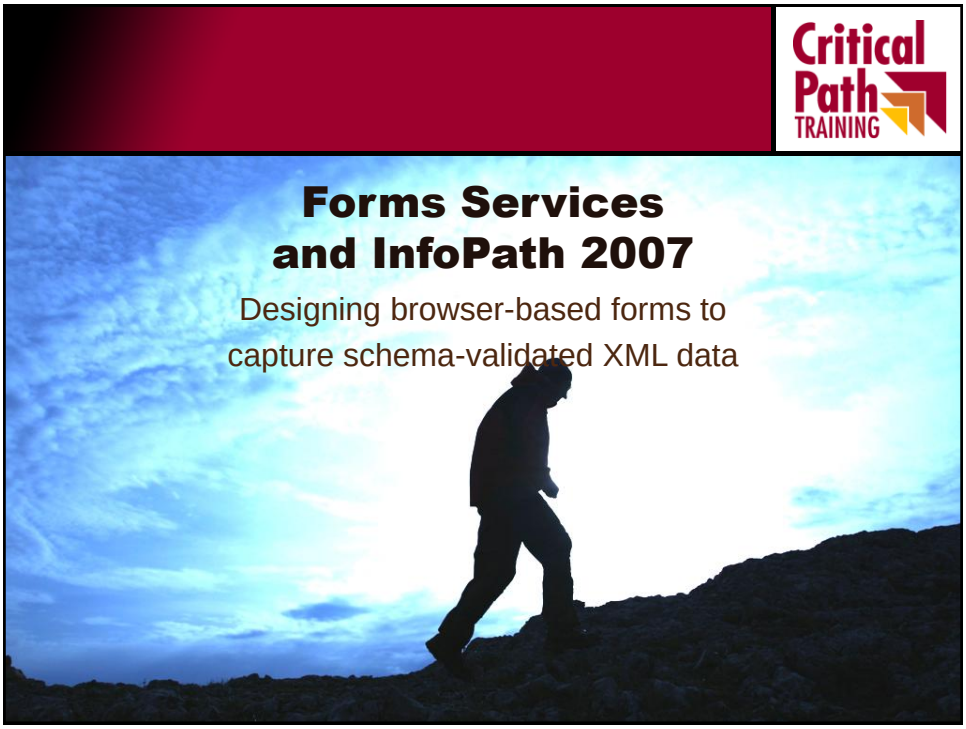
```
namespace Litware {
  public class TimesheetEventReceiver : Microsoft.SharePoint.SPItemEventReceiver  {
    public override void ItemUpdating(SPItemEventProperties properties) {
      SPWeb web = properties.OpenWeb();
      SPListItem timesheet = web.Lists[properties.ListId].GetItemById(properties.ListItemId);
      // check to make sure date is not day in future
      if (Convert.ToDateTime(timesheet["Submitted On"]).CompareTo(DateTime.Today) > 0) {
        properties.ErrorMessage = "You cannot enter future timesheets";
        properties.Cancel = true;
        return;
      }
    }
  }
}
```

Register receiver class through either OM code or feature element

```
SPList list = web.Lists["Timesheets"];
list.EventReceivers.Add(SPEventReceiverType.ItemAdding,
         "LitwareAssembly, [asm name]",
         "Litware.TimesheetEventReceiver");
```

Summary

- Content storage enhancements in WSS 3.0
- Querying data in lists
- WSS storage fundamentals
  - Site columns
  - Custom field types
  - Content types
- Provisioning lists and document libraries
- Event handling with receiver classes

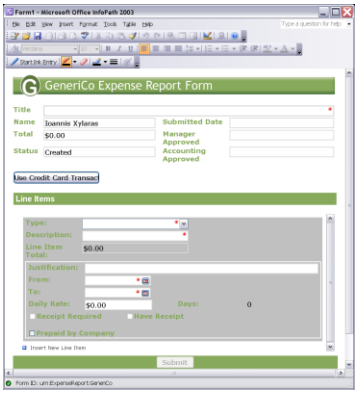


Agenda

- Background in InfoPath 2003
- The InfoPath Forms Designer
- Integration with WSS forms libraries
- Forms Services Architecture
- Designing server-side forms with InfoPath 2007
- Visual Studio Tools for Applications(VSTA)

The Role of InfoPath in Office 2003

- InfoPath was introduced with Office 2003
  - Platform for next generation of electronic forms
- InfoPath Forms
  - Captures XML data
  - Based on XML Schema
  - Requires little/no code



Challenges with InfoPath 2003

- Companies like InfoPath 2003, but…
  - They want better support for offline scenarios
  - They want to greater reach (browser-based clients)
  - They want a better code-behind model
- InfoPath 2007 introduces several improvements
  - Improved offline support through wizard
  - Forms Services extended InfoPath forms to browser
  - IT People Responsible for the Deployment
  - Code-behind using Visual Studio Tools Applications

Inside an InfoPath Form

- InfoPath for is a CAB file with .XSN extension
  - Contains manifest with form metadata (XSF)
  - Contains an XML Schema (XSD)
  - Contains XSL transforms to for view rendering
  - Contains XML files with data

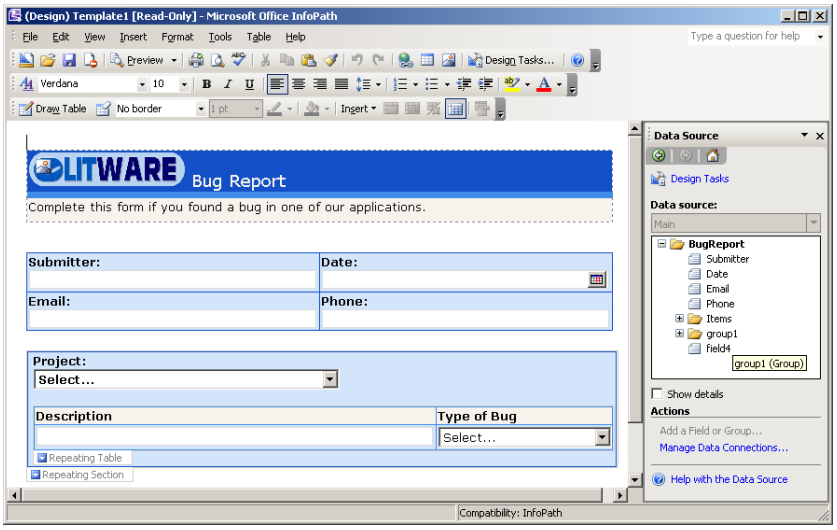
Security – Trust Levels

Restricted
- Deployed via email, no auto-updates
- No data connections, no managed code
- Not applicable for browser forms

Domain
- Deployed to SharePoint library, browser forms
- Connect to own server only, no code for browser forms
- Use trusted Data Connection Library for cross-domain

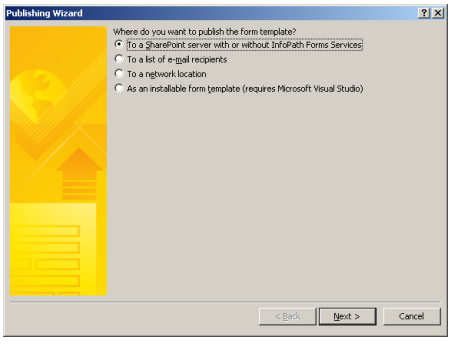
Full Trust
- Installed, Digitally Signed, or .NET Code Group
- Must be admin-deployed for browser forms
- Connect to any server, managed code in browser forms

Demo: The InfoPath Forms Designer



InfoPath Integration with WSS

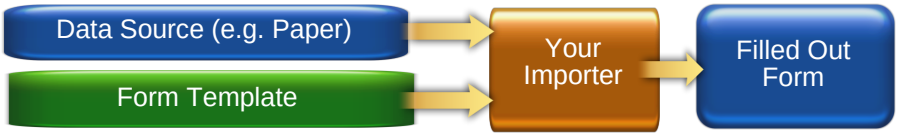
- Forms Libraries
  - A document library with a .XSN document template
  - Introduced with InfoPath 2003 and WSS 2.0
  - Create by users through InfoPath Publishing command



Template Importing

- Built-in support for Word, Excel documents
- Extensible framework
  - Options and progress only
  - IFormTemplateConverter
- Use in combination with the Design Checker

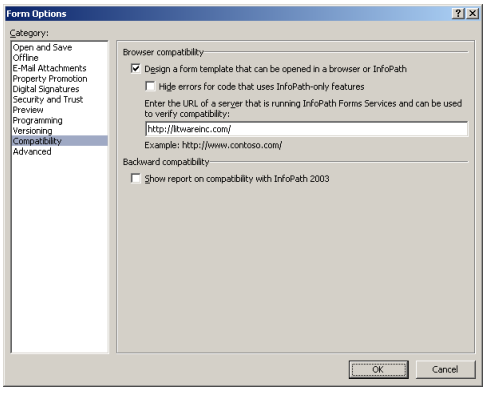
Data Importing



- No OOB solutions for this
- Extensible framework
  - Any custom UI
  - IInfoPathDataImporter

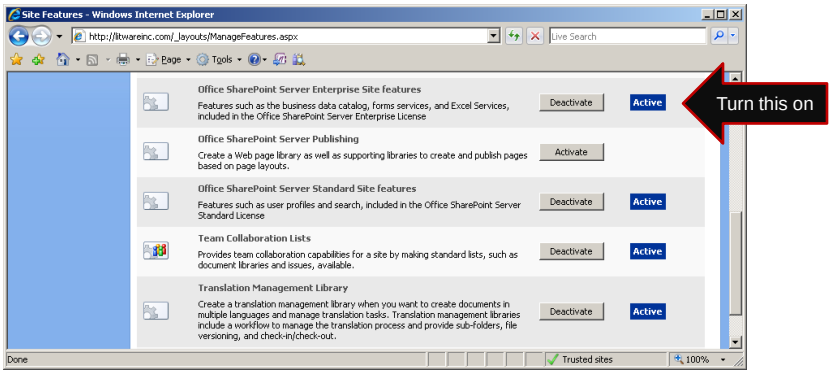
Browser-based Forms

- Forms Services provides HTML rendering
  - Forms must be designed using InfoPath 2007
  - Forms must be designed to be browser compatible

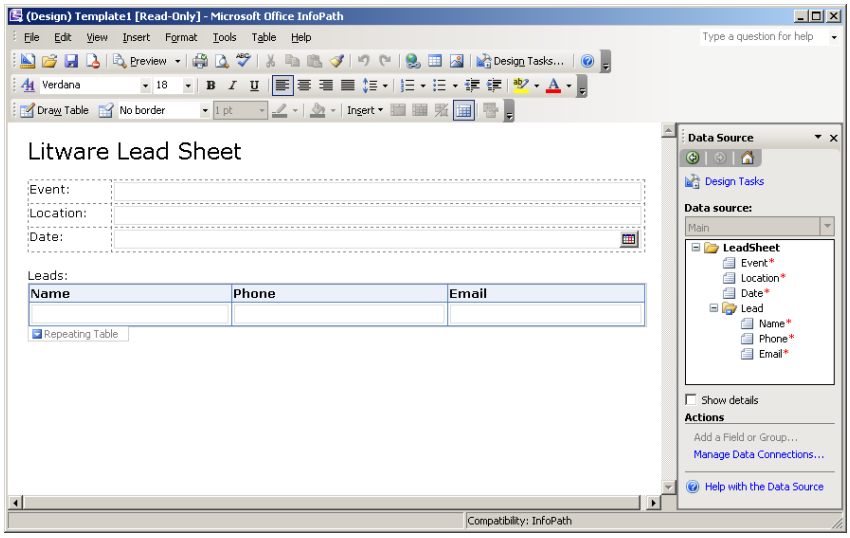


Sites and Browser-based Publishing

- Activate MOSS Standard and Enterprise features
  - Should be done for target site and site collection

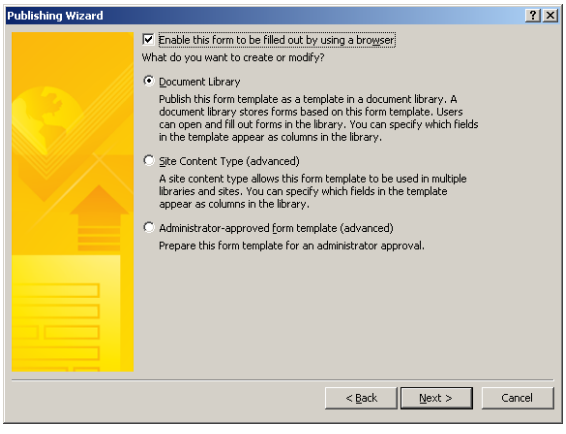


Designing a Browser-based Form



Publishing a Browser-based Form

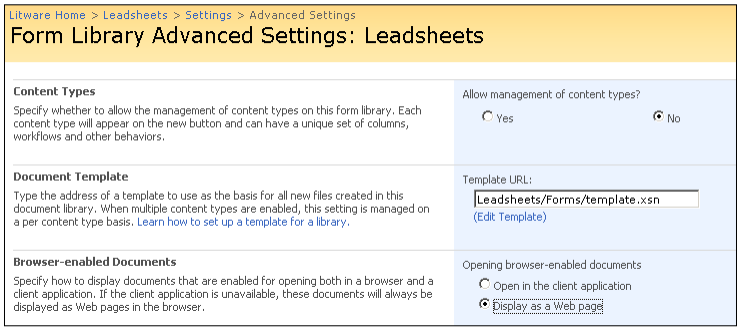
- Saved up to WSS Forms Library
  - MOSS uses document template .xsn file for rendering



Forms Library Settings

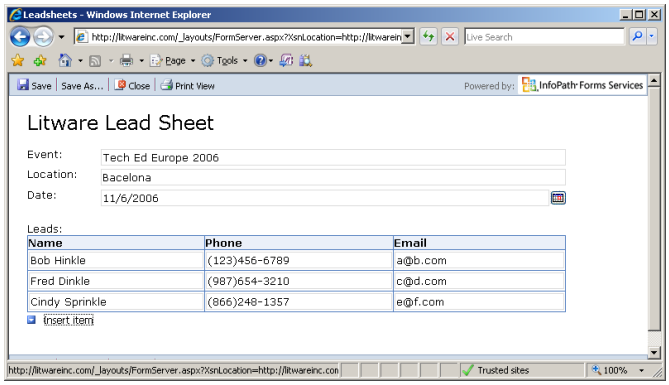
- Important Forms Library settings
  - template.xsn is the editable InfoPath form template
  - Open browser-enabled documents

    `The default is to open with InfoPath rich client if possible`



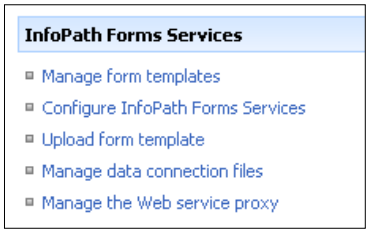
Browser-based Rendering

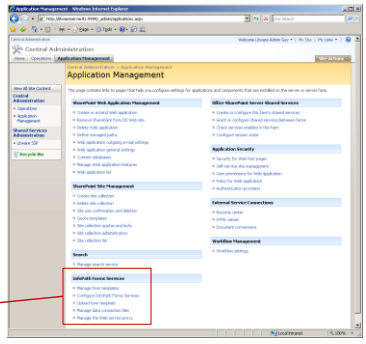
- Browser-based rendering for wide reach
  - Based on DHTML and JavaScript
  - Tested with IE, FireFox, Netscape & hand-held devices



Forms Services Administration

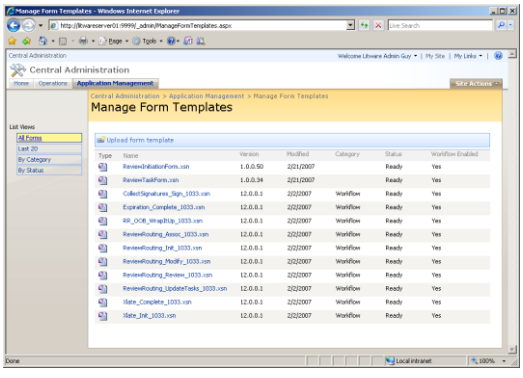
- Part of WSS Central Administration
  - Used to upload/manage forms and data connections





Administrator Uploaded Form Templates

- Some forms must be uploaded by administrator
  - Forms with code and/or forms with data connections
  - Benefit: deployed at farm scope not at site scope

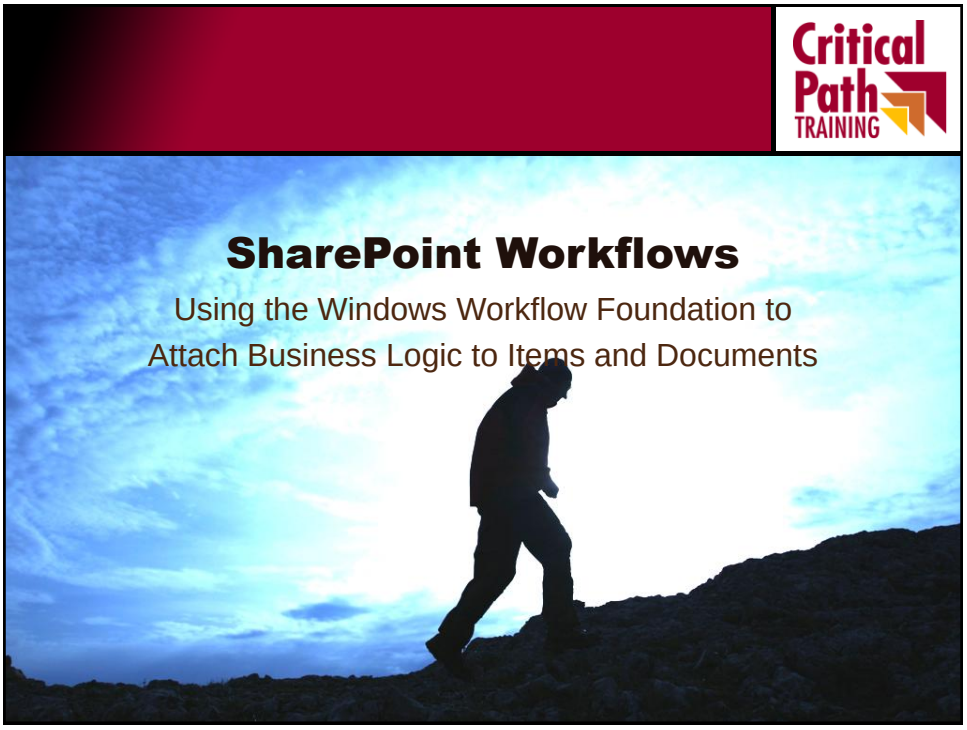


Supporting the .NET Developer

- Visual Studio Tools for Applications (VSTA)
  - Provided with InfoPath 2007 out-of-box
  - Lowers the bar for forms with managed code
  - Compatible with Visual Studio Tools for Office
- Visual Studio Tools for Office (VSTO)
  - Embedded designer for professional developers
  - One IDE for all your projects: Workflow, etc.
  - Integrated toolbox, project wizard, etc.

Summary

- Background in InfoPath 2003
- The InfoPath Forms Designer
- Integration with WSS forms libraries
- Forms Services Architecture
- Designing server-side forms with InfoPath 2007
- Visual Studio Tools for Applications(VSTA)



Agenda

- Windows Workflow Foundation (WF) Primer
- Creating WF programs in Visual Studio
- Creating workflow templates for WSS
- Workflow associations and workflow instances
- Creating and waiting on WSS tasks
- Integrating workflow input forms

Reactive Programs

- Automating a business process
  - Often requires program with episodic behavior
  - Program waits around and then reacts to some event
- How would you automate document approval?
  - In a Windows Forms application…
  - In an ASP.NET Application

Windows Workflow Foundation (WF)

- What is the Windows Workflow Foundation?
  - Development platform for building reactive programs
  - Set of development tools integrated with Visual Studio
  - Runtime components that ship with.NET FX 3.0
- Windows Workflow Foundation concepts
  - WF program
  - Workflow instance
  - Activities

Activities

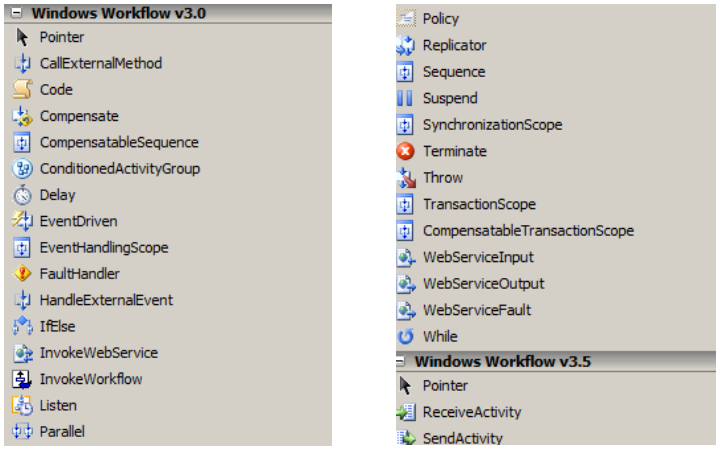
- An activity is…
  - atomic set instructions used complete a unit of work
  - reusable component used to compose WF programs
- Activities are like a controls in forms development
  - You drag and drop them onto a design surface
  - You modify their properties through property sheet
  - You generate event handlers and write code inside
- Activities are different than controls
  - Activities are resumable

Composite Activities

- Composite Activities can contain children
  - Composite activity controls execution of children
  - Composite activity can encapsulate control-of-flow
  - Examples: IfElse, While, Sequence, Parallel, Replicator
- WF program is itself a composite activity
  - WF program models a tree of activities

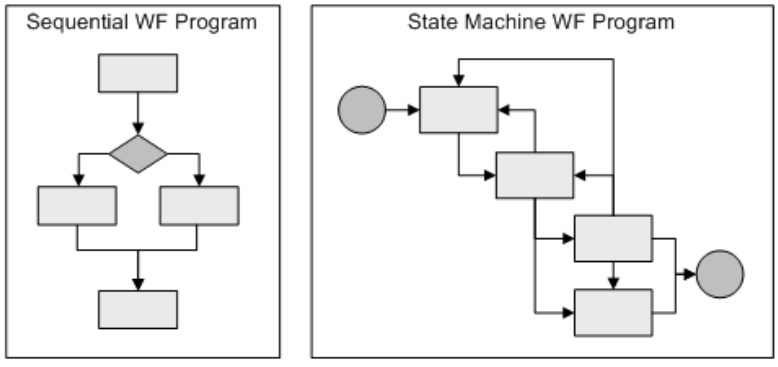
WF Base Activity Library

- Standard WF activities provide basic building blocks

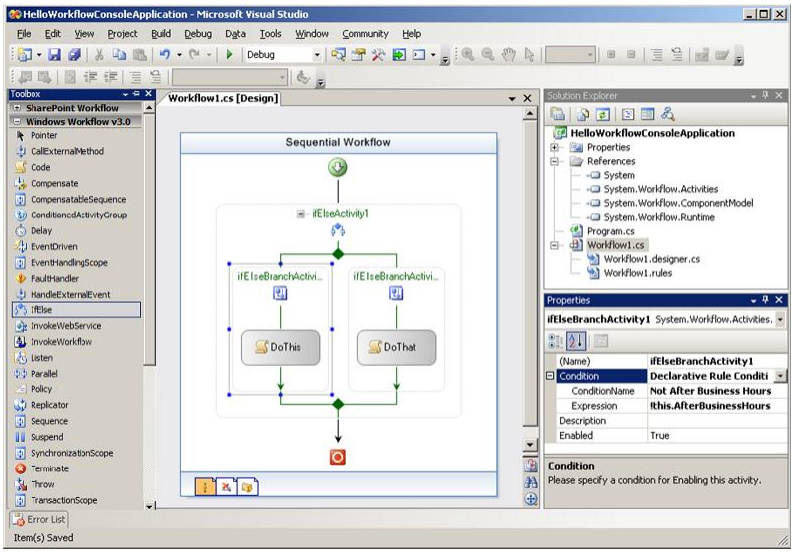


WF Program Types

- WF provides two main styles of WF programs
  - Sequential WF program modeled as flow chart
  - State machine WF program models using states



Visual Studio Workflow Designer

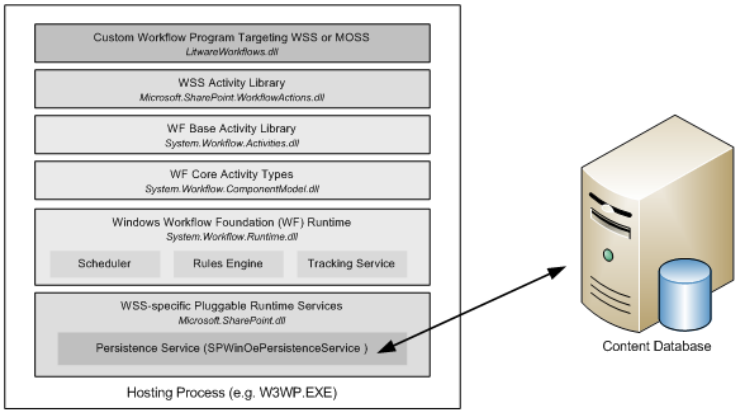


The WF Runtime

```
using System;
using System.Workflow.Runtime;
using System.Workflow.Runtime.Hosting;

namespace HelloWorkflowConsoleApplication {
  class Program {
    static void Main() {
      // start WF runtime
      using(WorkflowRuntime workflowRuntime = new WorkflowRuntime()) {
            AutoResetEvent waitHandle = new AutoResetEvent(false);
            workflowRuntime.WorkflowCompleted +=
               delegate(object sender, WorkflowCompletedEventArgs e)  {
                  waitHandle.Set();
               };
             workflowRuntime.WorkflowTerminated +=
                delegate(object sender, WorkflowTerminatedEventArgs e) {
                   Console.WriteLine(e.Exception.Message);
                   waitHandle.Set();
                };

             // create and start workflow instance
             WorkflowInstance instance = workflowRuntime.CreateWorkflow(
                 typeof(WorkflowConsoleApplication1.Workflow1));
             instance.Start();
             waitHandle.WaitOne();
       }
    }
  }
}
```

WF Runtime Services

- Custom services can be written and plugged in
  - WSS provides its own persistence service



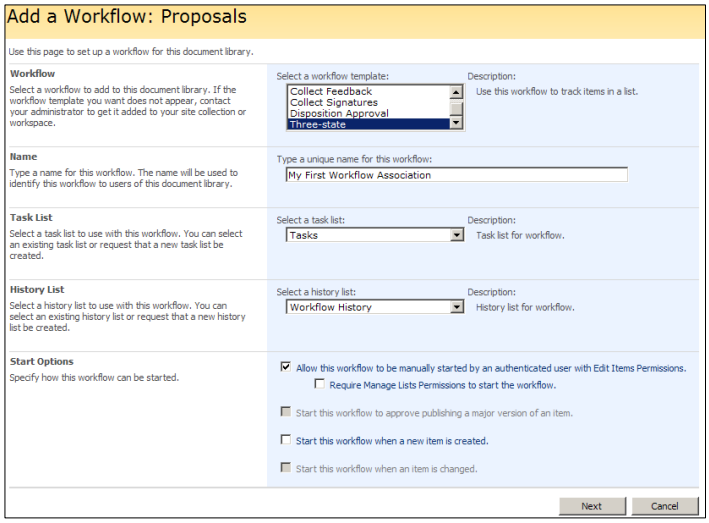
SharePoint Workflow Concepts

- Design goals for WF integration with WSS
  - Use WF to attach logic to items and documents
  - Add a human dimension on top of WF
  - Maintain self-service capabilities common in WSS
  - Create strong developer story for custom WF programs
  - Provide valuable WF programs out-of-box with MOSS
- The human dimension
  - Any SharePoint workflow can assign tasks to users
  - Users can see the status of any workflow instance

SharePoint Workflow Actors

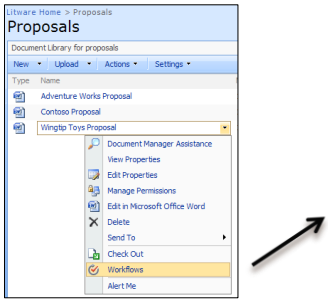
- Workflow Template
  - WF Program and optionally workflow input forms
  - A feature to install it inside WSS farm
- Workflow Association
  - Binding of workflow template to list or content type
  - A named instance containing parameterized data
- Workflow Instance
  - A running instance of a WF program attached to an item

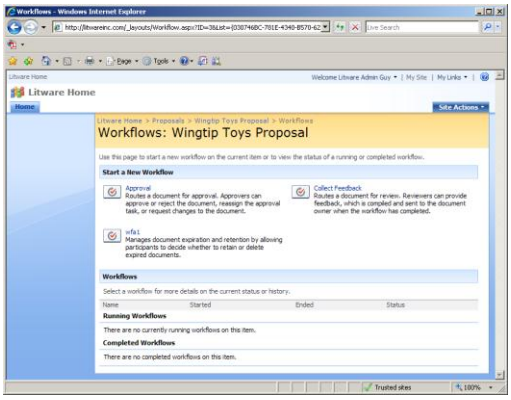
Creating a Workflow Association



Starting a Workflow Instance

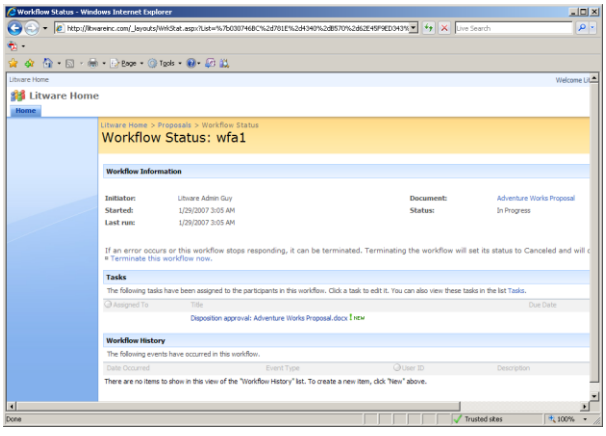
- Users can manually start workflows





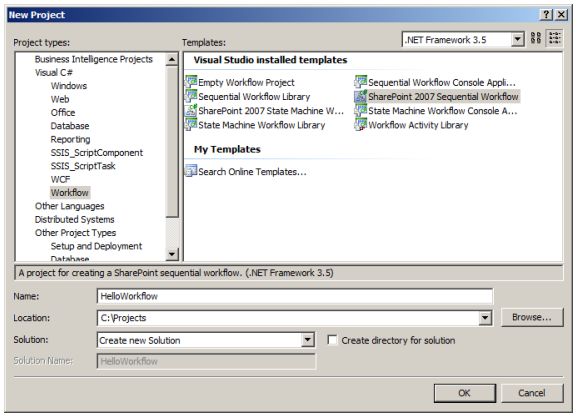
The Workflow Status Page

- Any user can see the status of a workflow instance



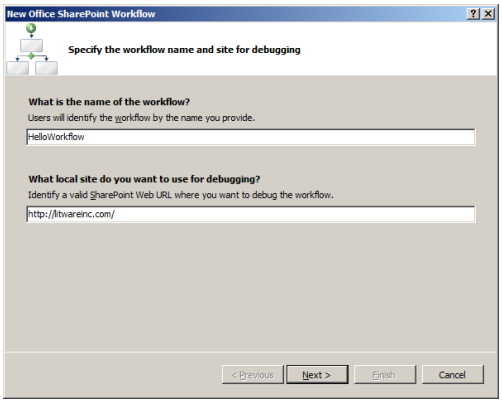
Creating a Workflow Template Project

- Creating a SharePoint Workflow Project in Visual Studio 2008



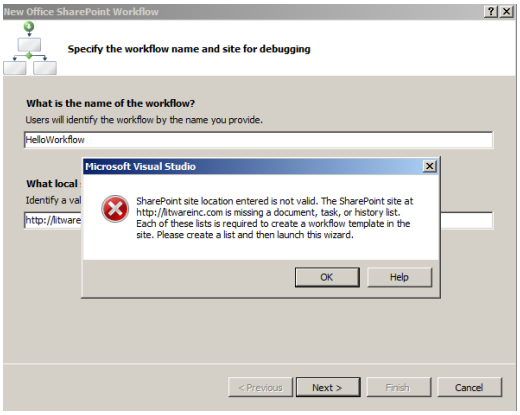
Complete the Wizard

- Step 1 – Specify SharePoint URL
  - Enter the name of the workflow
  - Specify the URL to SharePoint site



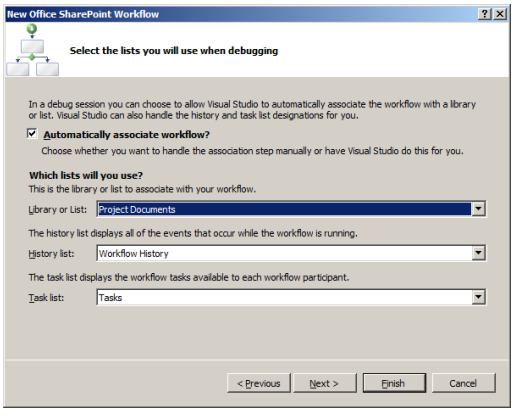
Complete the Wizard

- Following lists need to be available:
  - Document Library
  - Tasks list
  - History list



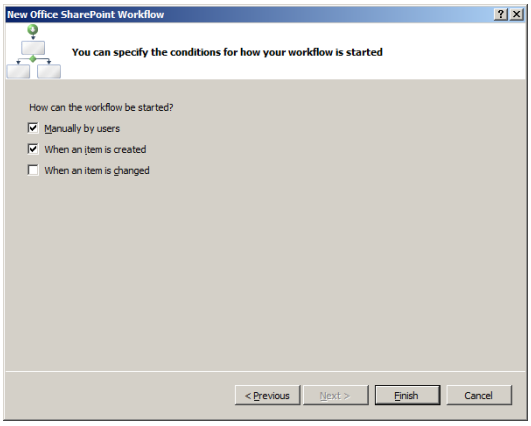
Complete the Wizard

- Step 2 – select the necessary lists
  - List or document library to associate workflow
  - History list
  - Tasks list



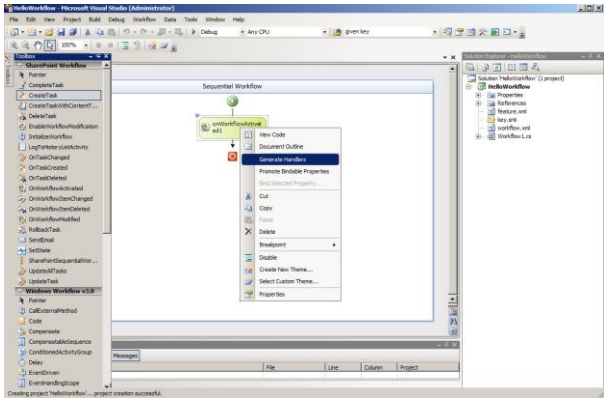
Complete the Wizard

- Step 3 – decision on how to start the workflow



Developing the WF Program

- Getting around inside the Workflow Designer
  - Learn to move between Designer View and Code View
  - Get to Know the Activities in the SharePoint Activity Library



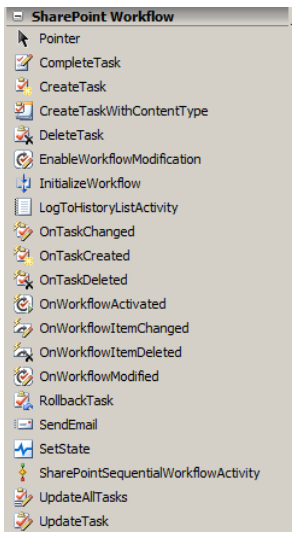
Working in Code View

- Here is what you get as a starting point

```
using Microsoft.SharePoint.Workflow;

namespace HelloWorkflow {

  public partial class Workflow1 : SharePointSequentialWorkflowActivity {

    // code to call wizard-generate code
    public Workflow1() {
      InitializeComponent();
    }

    // default fields added by project template
    public Guid workflowId = default(System.Guid);
    public SPWorkflowActivationProperties workflowProperties =
                          new SPWorkflowActivationProperties();

    // TODO: add fields here

    // TODO: add event handlers here
  }
}
```

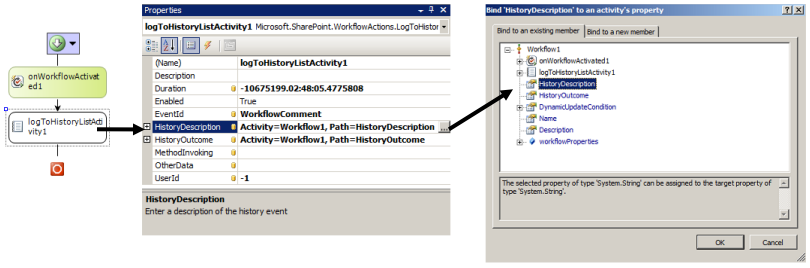
SharePoint Activity Library

- WSS-specific activities used to create SharePoint WF Programs



Data Bound Properties

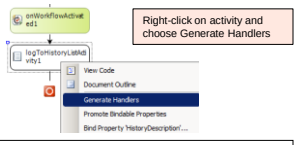
- WF supports data binding of properties
  - Allows for declarative flow of data between activities
  - Used extensively for creating SharePoint WF programs




```
public partial class Workflow1 : SharePointSequentialWorkflowActivity {
  // other members removed for clarity
  public String HistoryDescription;
  public String HistoryOutcome;
}
```

Generating Event Handlers

- Generate event handlers to add code
  - Event handlers can program against WF objects




```
public class Workflow1 : SharePointSequentialWorkflowActivity {

  public SPWorkflowActivationProperties workflowProperties =
                          new SPWorkflowActivationProperties();
  public String HistoryDescription;
  public String HistoryOutcome;

  private void logActivated_MethodInvoking(object sender, EventArgs e) {
    // Generate message using information of current item
    SPListItem item = workflowProperties.Item;
    // determine whether workflow is running on a standard item or a document
    if (item.File == null) {
      HistoryDescription = "Workflow started on item " + item.Title;
    }
    else {
      HistoryDescription = "Workflow started on document " + item.File.Name;
    }
    HistoryOutcome = "Workflow activation complete";
  }
}
```

Workflow Template Deployment

- Workflow Templates are deployed via Features
  - Feature must be scoped to site collection (Scope=Site)
  - Feature may contain multiple workflow templates

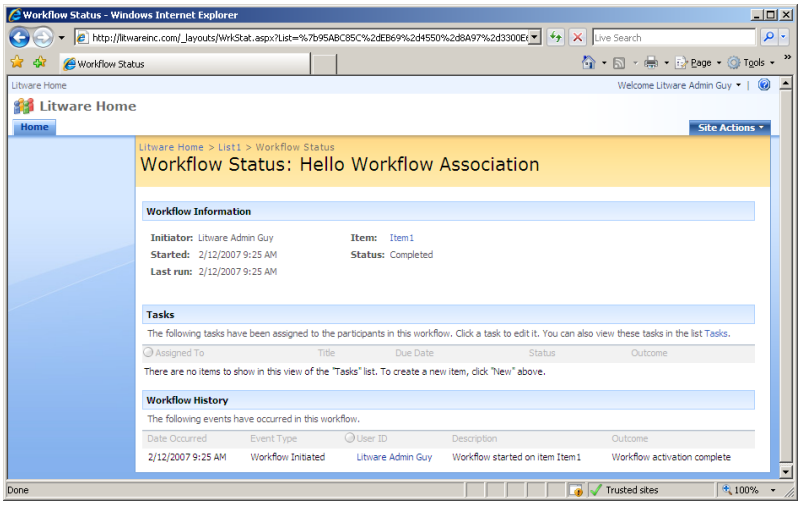
```
<Feature
  Id="0CEED7AE-D327-41ad-BC33-B3F3F8A4DAD2"
  Title="Hello World Workflow Template Feature"
  Description="This feature installs our Hello World Workflow Template"
  Version="12.0.0.0"
  Scope="Site"
  xmlns="http://schemas.microsoft.com/sharepoint/">

  <ElementManifests>
    <ElementManifest Location="workflow.xml" />
  </ElementManifests>

</Feature>
```

Workflow Template Definition

- Workflow Element defines Workflow Template
  - Must point to one specific WF program
  - WF program must be compiled into an assembly DLL
  - Assembly DLL must be installed in GAC

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Workflow
    Id="1EE1C818-DB7A-4a55-B21B-959D413C6A9C"
    Name="Hello World Workflow Template"
    Description="This WF template provides classic Hello World functionality"
    CodeBesideClass="HelloWorkflow.Workflow1"
    CodeBesideAssembly="HelloWorkflow, [four-part assembly name]" >

    <Categories/><!-- no categories needed -->

    <MetaData /><!-- no metadata needed -->

  </Workflow>
</Elements>
```

Testing 'Hello World' Workflow Template

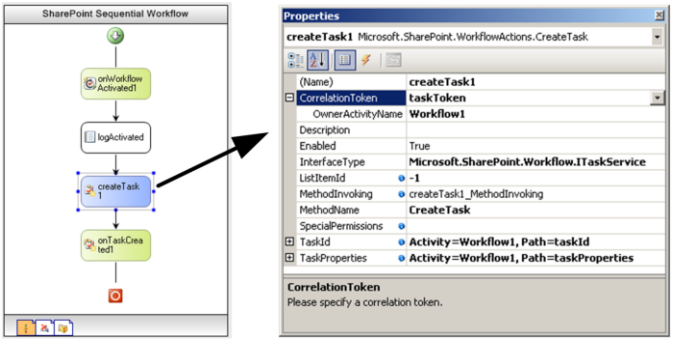


Creating and Waiting on Tasks

- SharePoint Workflows revolve around tasks
  - Represents significant value-add WSS brings to WF
  - Based on standard WSS tasks visible/editable by users
  - Users update tasks through browser or Office programs
  - Your code automatically wakes up and executes
- WSS Tasks are generated with subscriptions
  - WSS encapsulates the listener mechanism
  - WSS registers event handlers behind the scenes
  - You just add event activities and write event handlers

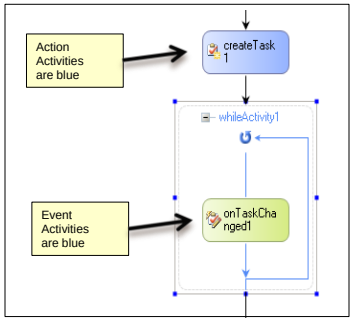
Task GUIDs and Correlation Tokens

- WSS sets up subscriptions for tasks
  - Based on registering event handlers
  - WSS needs way to identify certain task across activities
  - Each tasks is assigned a GUID and a correlation token



Action Activities vs. Event Activities

- Action activities perform work
  - Their event handlers fire before work is done
- Event activities run code in response to an event
  - Their event handlers run after the event has occurred



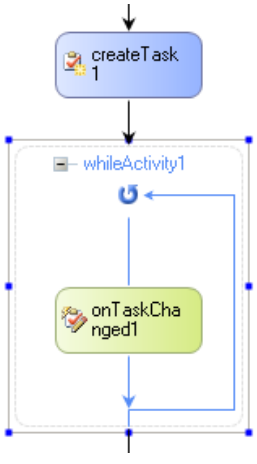
Initializing a New Task

- Add event handler behind CreateTask activity
  - This event handlers fires before task creation
  - Gives you a chance to initialize task properties

```
// these fields are data-bound to properties of task activities
public Guid taskId = default(System.Guid);
public SPWorkflowTaskProperties taskProperties =
                                new SPWorkflowTaskProperties();

private void createTask1_MethodInvoking(object sender, EventArgs e) {
  // generate new GUID used to initialize task correlation token
  taskId = Guid.NewGuid();
  // assign initial properties prior to task creation
  taskProperties.Title = "Task for " + workflowProperties.Item.Title;
  taskProperties.Description = "Please review and approve this item.";
  taskProperties.AssignedTo = @"LITWAREINC\BrianC";
  taskProperties.PercentComplete = 0;
  taskProperties.StartDate = DateTime.Today;
  taskProperties.DueDate = DateTime.Today.AddDays(2);
}
```

Waiting on a Task

- Event activity creates subscription
  - OnTaskChanged puts activity to sleep
  - Event handler fires upon modifcation
- While activity used to control flow
  - While activity loops until task complete



Creating Workflow Forms with ASP.NET

- Workflow input forms can be created in ASP.NET
- Benefits to creating workflow forms with ASP.NET
  - Can run from WSS-only farms
- Drawback to creating forms with ASP.NET
  - More coding involved

ASP.NET Workflow Form Intergation

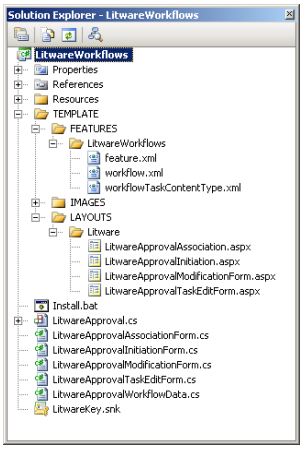
```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Workflow
    Name="LitwareWorkflowsInfoPath"
    Description="Simple workflow with InfoPath forms."
    Id="48500BEB-D1BE-4ec4-8D21-5DEF76BEEDA8"
    CodeBesideClass="LitwareWorkflowsInfoPath.Workflow1"
    CodeBesideAssembly="LitwareWorkflowsInfoPath, [full assembly name]"
    TaskListContentTypeId="0x01080100C9C9515DE4E24001905074F980F93160"
    AssociationUrl="_layouts/Litware/LitwareApprovalAssoc.aspx"
    InstantiationUrl="_layouts/Litware/LitwareApprovalInit.aspx"
    ModificationUrl="_layouts/Litware/LitwareApprovalMod.aspx">

    <MetaData>
      <StatusPageUrl>layouts/WrkStat.aspx</StatusPageUrl>
    </MetaData>

    <Categories/>

  </Workflow>
</Elements>
```

standard MOSS task content type

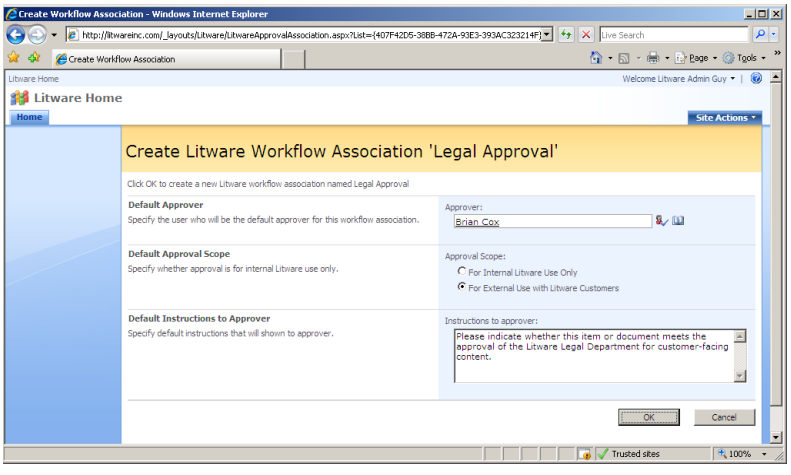
custom application pages

Integrating Workflow Input Forms

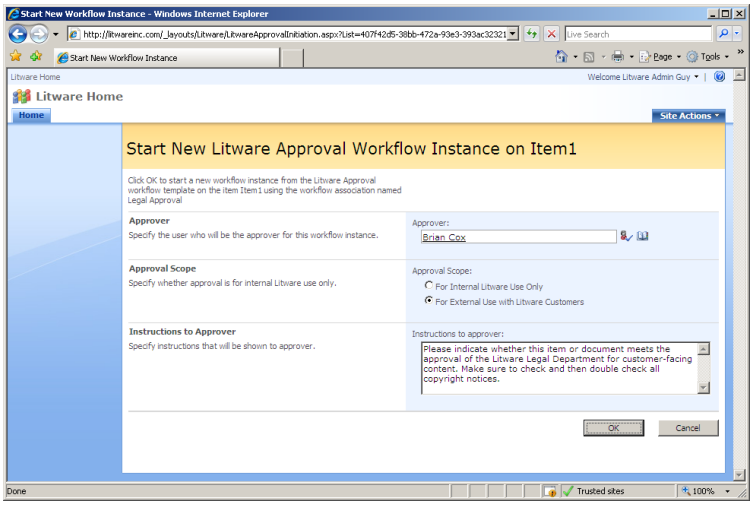
- Workflow Input Form Types
  - Association Form
  - Initiation Forms
  - Modification Forms
  - Task Edit Form
- Sample Project
  - LitwareWorkflows



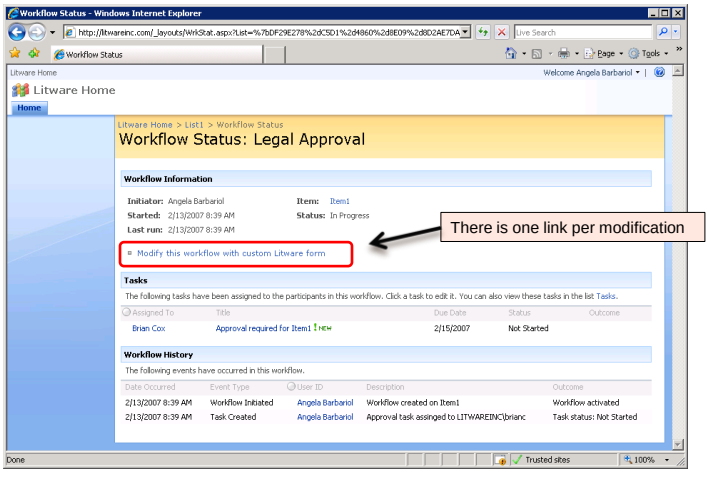
The Association Form



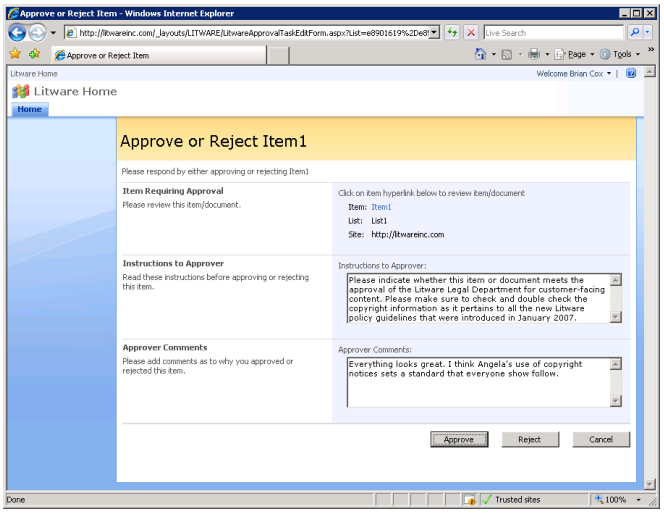
The Initiation Form



Invoking the Modification Form



The Task Edit Form



Creating Workflow Forms with InfoPath

- Workflow input forms can be created in InfoPath
- Benefits to creating workflow forms with InfoPath
  - Significantly better forms designer experience
  - Significantly less coding
  - Forms can be opened directly with Office client apps
- Drawback to creating forms with InfoPath
  - Workflow template will only run in MOSS farms
  - Workflow template will not run in WSS-only farms

InfoPath Workflow Form Intergation

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Workflow
    Name="LitwareWorkflowsInfoPath"
    Description="Simple workflow with InfoPath forms."
    Id="48500BEB-D1BE-4ec4-8D21-5DEF76BEEDA8"
    CodeBesideClass="LitwareWorkflowsInfoPath.Workflow1"
    CodeBesideAssembly="LitwareWorkflowsInfoPath, [full assembly name]"
    TaskListContentTypeId="0x01080100C9C9515DE4E24001905074F980F93160"
    AssociationUrl="_layouts/CstWrkflIP.aspx"
    InstantiationUrl="_layouts/IniWrkflIP.aspx"
    ModificationUrl="_layouts/ModWrkflIP.aspx">

    <MetaData>
      <Instantiation_FormURN>[INFOPATH FORM ID]</Instantiation_FormURN>
      <Association_FormURN>[INFOPATH FORM ID]</Association_FormURN>
      <Task0_FormURN>[INFOPATH FORM ID]</Task0_FormURN>
    </MetaData>

    <Categories/>

  </Workflow>
</Elements>
```

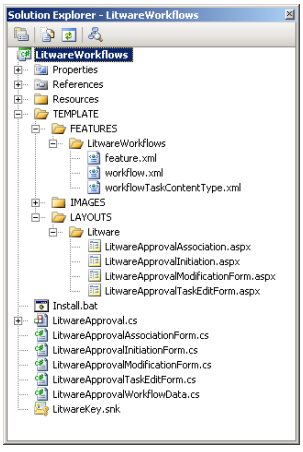
standard MOSS task content type

standard MOSS application pages

urn:schemas-microsoft-com:office:infopath:ReviewInitiationForm2:-myXSD-2005-11-22T23-49-53

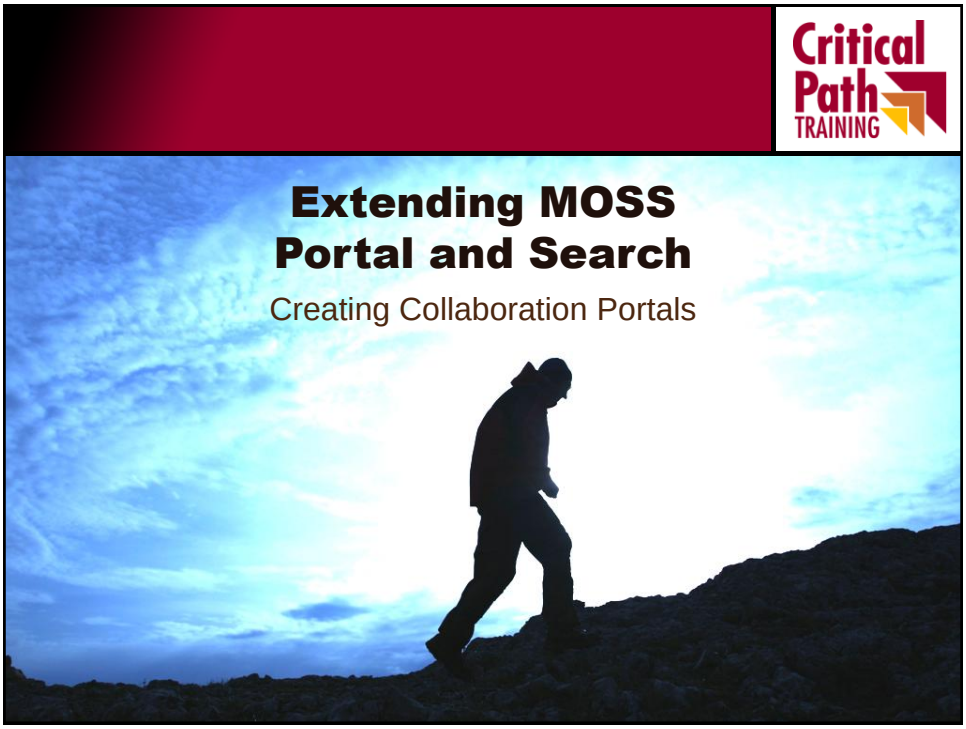
Integrating Workflow Input Forms

- Workflow Input Form Types
  - Association Form
  - Initiation Forms
  - Modification Forms
  - Task Edit Form
- Sample Project
  - LitwareWorkflows



Summary

- Windows Workflow Foundation (WF) Primer
- Creating WF programs in Visual Studio
- Creating workflow templates for WSS
- Workflow associations and workflow instances
- Creating and waiting on WSS tasks
- Integrating workflow input forms



Agenda

- Collaboration Portals
- Shared Service Provider (SSP) Architecture
- Creating corporate portal sites
- User Profiles
- Audience Targeting
- MySites
- Extending Search

SharePoint Portal Server 2003 (SPS)

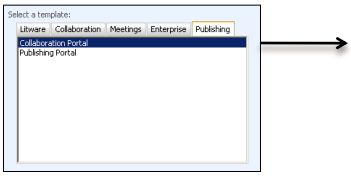
- SharePoint Portal Server 2003 Features
  - Areas and listings
  - User profiles and audience targeting
  - Search
  - My Sites
  - Shared Services
  - Single sign on

Problems with SPS 2003

- Problems with SharePoint Portal Server 2003
  - A strange semi-undocumented layer on top of WSS V2
  - Limited to one SPS portal site per IIS Web application
  - Portal user interface hard to customize and extend
  - Shared Services architecture hard to deploy and administrate
  - Limited and clunky integration with Content Management Server

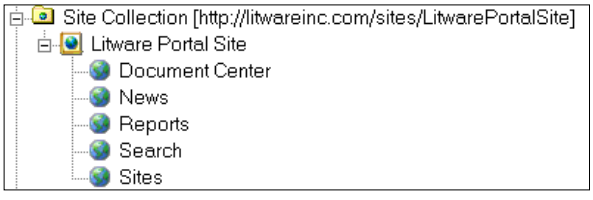
Collaboration Portal Site Template

- Used to create SPS-like Portal sites
  - Supplies same basic features as SPS
  - Except that Areas and Listings have been eliminated



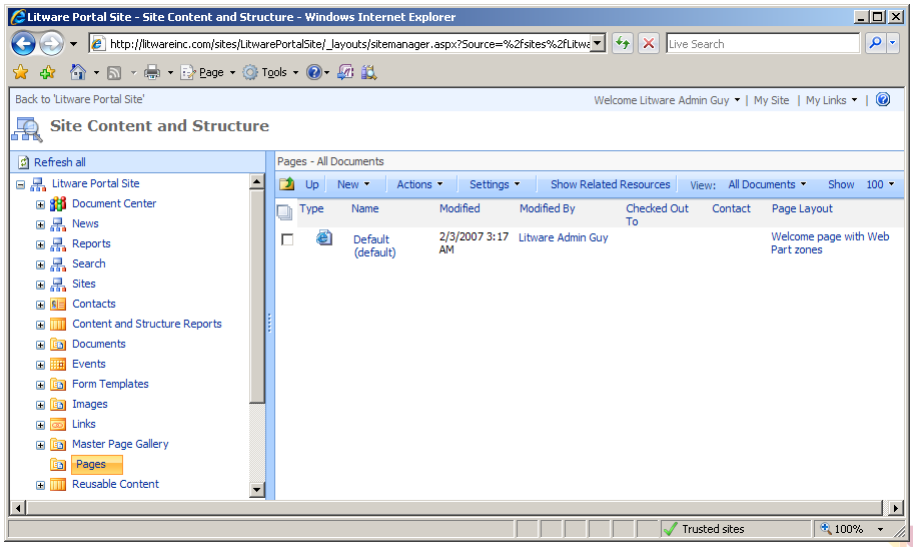
Collaboration Portal Architecture

- A collaboration portal is a hierarchy of WSS sites
  - Created as a WSS site collection
  - Created using standard Site Definitions and Features



- Architectural improvements over SPS portals
  - Not limited to one portal per IIS Web site
  - Portals extended using standard WSS V3 development

Site Content and Structure

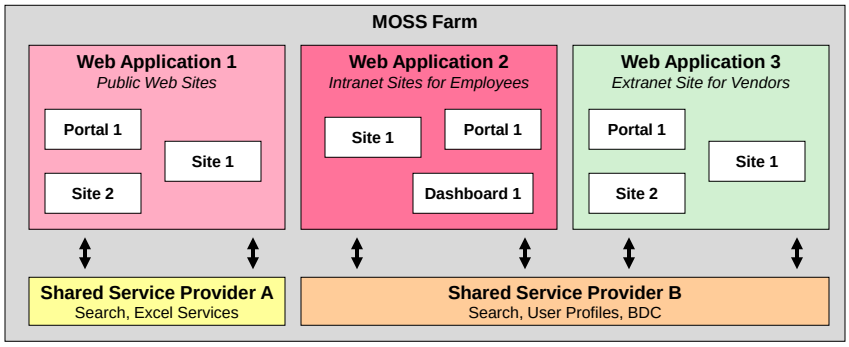


Sharing Resources and Services

- Need to share resources/services across sites
  - Content crawling and index creation
  - Querying for search results
  - Allocating and managing storage for My Sites
- SPS used 1st generation shared services model
  - One portal site is selected as "Master" portal
  - Other portals use shared services of master portal
- MOSS takes a different and better approach
  - Enter the Shared Service Provider (SSP)

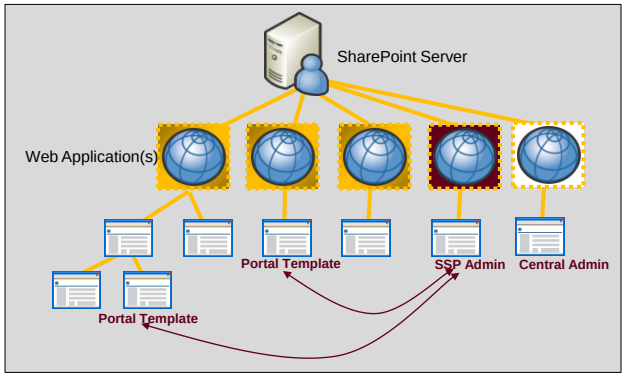
SSP Architecture

- Shared Service Provider (SSP)
  - Configured independently of any portal or site
  - SSP resources and services are accessible to all types of sites
  - Each Web Application maps to exactly one SSP



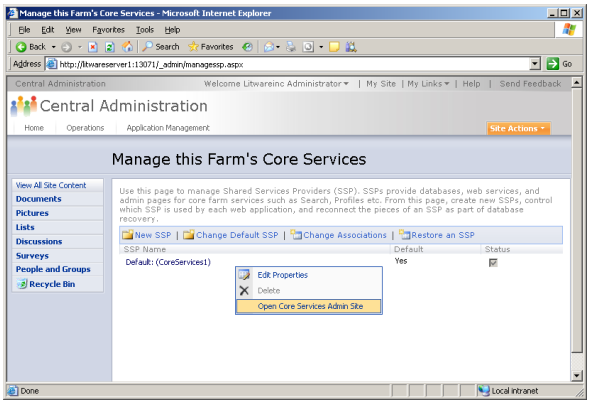
So what is a Portal, Really?

- A Site Collection that…
  - has the correct features activated
  - is consuming services from an underlying SSP
  - might or might not have been created from a portal site template



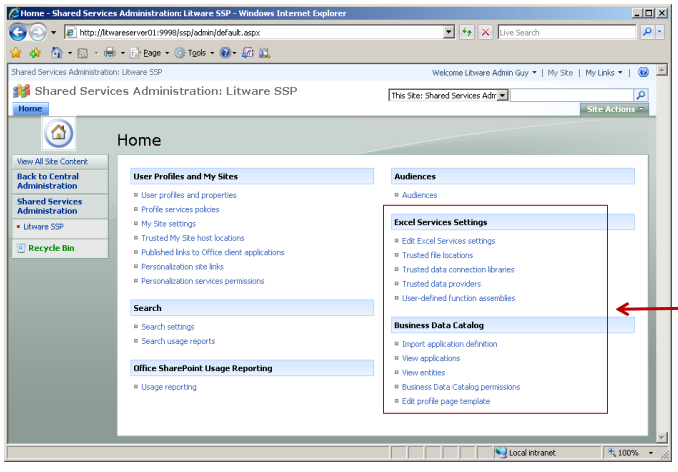
Creating An SSP

- Central Administration ->
  Application Management ->
  Create or Configure this Farm's Core Services



SSP Administration

- Functionality of SSP is Broken out into Sections



User Profiles

- User profiles track user information for social networking
  - Information used to describe users to each other
  - Information used to target information to interested parties
- MOSS maintains user profiles in SQL Server
  - Profile data can be imported from Active Directory or LDAP
  - Profile data can be extended with custom properties
  - Profiles can be extended with a BDC data source
- What MOSS features rely on user profiles?
  - My Site infrastructure
  - Audience targeting
  - People Search

People and Personalization

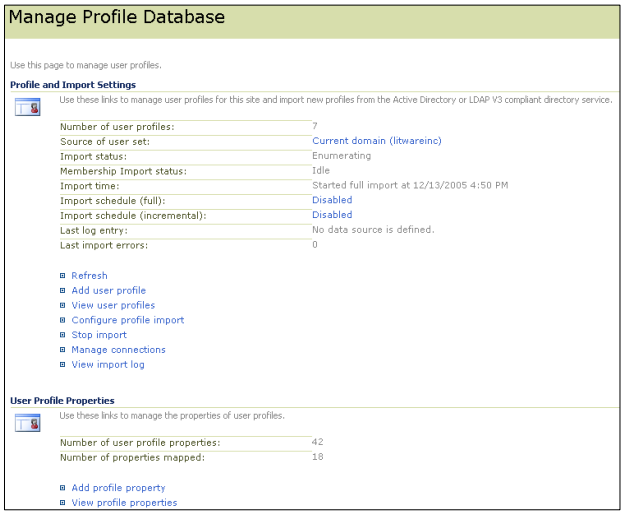
- Custom components can access…
  - Privacy policies
  - Audience targeting
- User profile store: reading/writing:
  - Users
  - Profiles
  - Memberships
  - Colleagues
  - Audiences

Configuring User Profiles

Administrative control for adding, viewing, managing and importing profiles

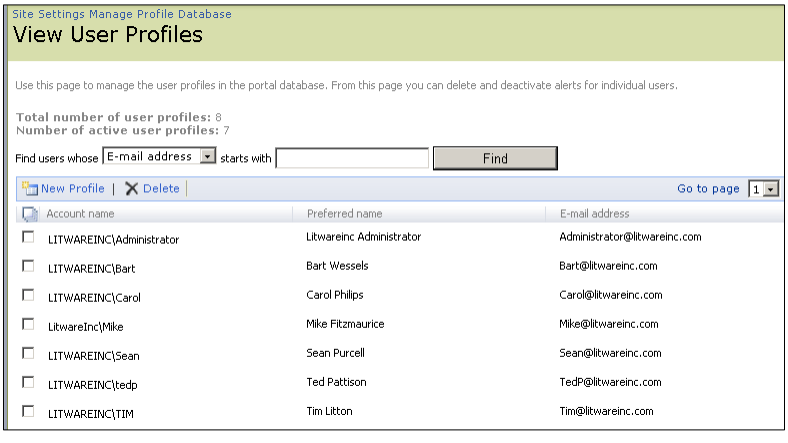
You can see and modify user profile properties

You can add application-specific properties



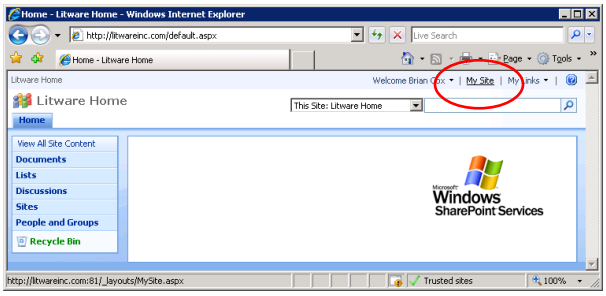
Viewing User Profiles

- User profiles can be added and deleted



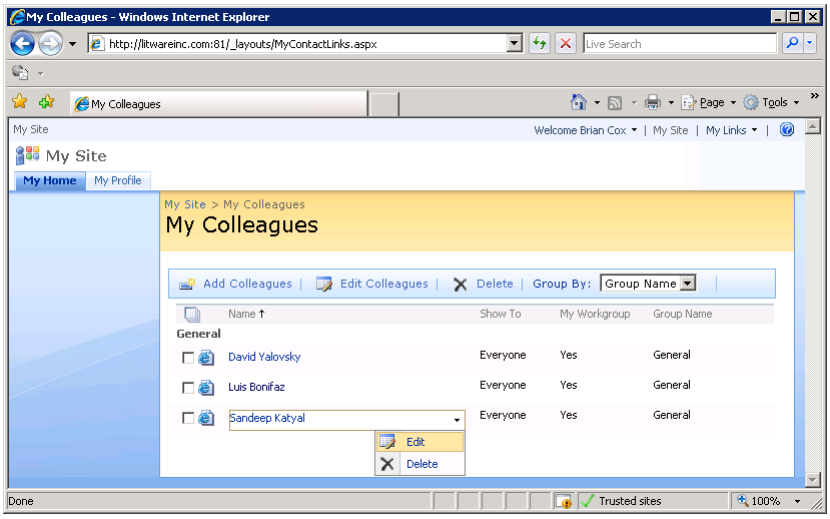
Creating a new My Site

- Each My Sites is a site collection that…
  - is provided on demand upon first access
  - maps to a specific user profile
  - enables users to edit some aspects of their profile
  - Provides a private aspect and a publicly-facing aspect



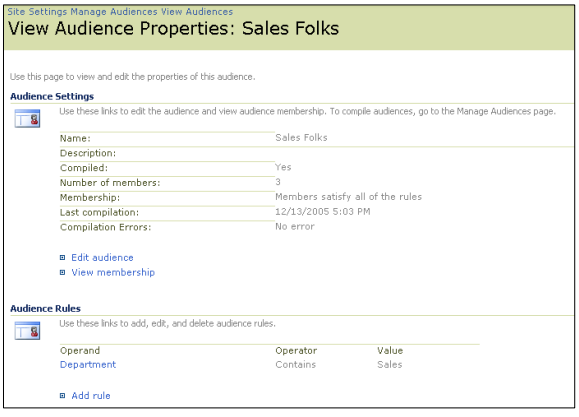
Updating A User Profile

Colleagues



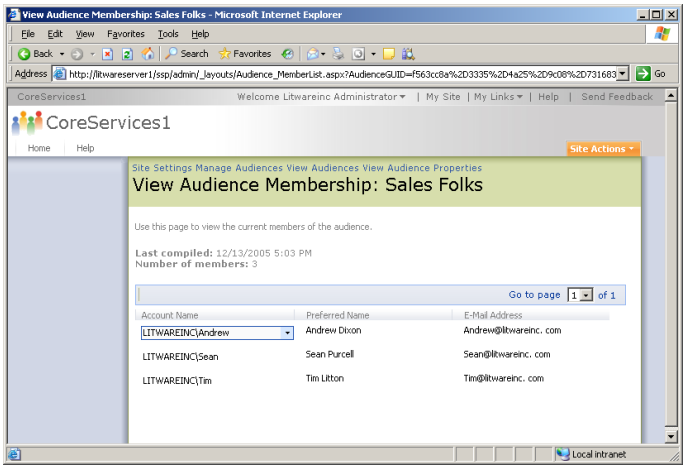
Audiences

- "Audience" involves creating rules and then compiling
  - Rules define what user accounts should be included or excluded
  - The Compilation process adds users to an audience table in SQL Server



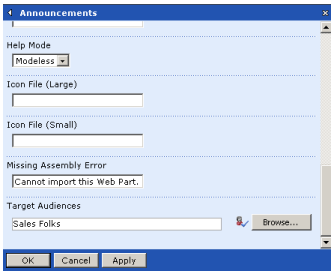
Compiled Audience

- Compiled Audience defines a list of users



Audience Targeting

- Web Part output can be targeted to an audience
  - Web Part content is shown to members of that audience
  - Web Part content is not shown to users not in that audience
  - Great way to get content to interested parties while filtering noise from those that are not interested

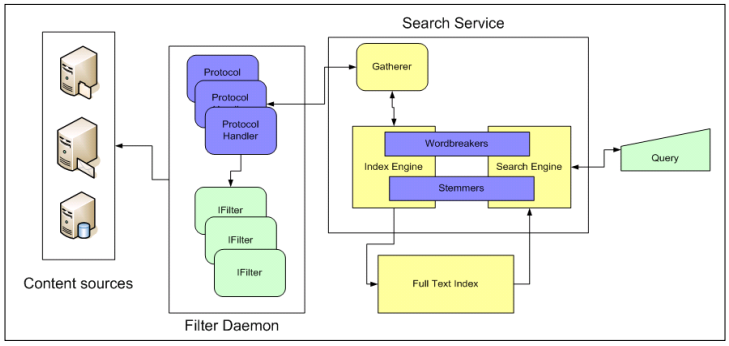


WSS Versus Office Server Search

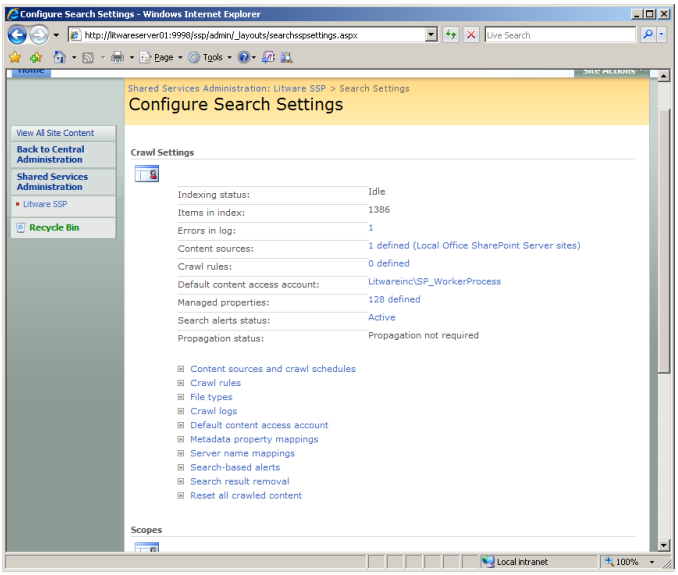
- Windows SharePoint Services (WSS 3.0)
  - WSS search is a subset of the MOSS search feature
  - Eases transition from WSS to MOSS
  - Indexing and query are always on the same machine
  - Search over site content only
- Microsoft Office SharePoint Server (MOSS)
  - Adds new search functionality over base WSS search
  - Indexing and query can be separated on different machines
  - Multiple catalogs supported per indexer and query servers
  - Aggregated content - local + external

Search Architecture and Terminology

- Key pieces to search infrastructure
  - The Gatherer
  - Content sources, protocol handlers and IFilters
  - Index Files

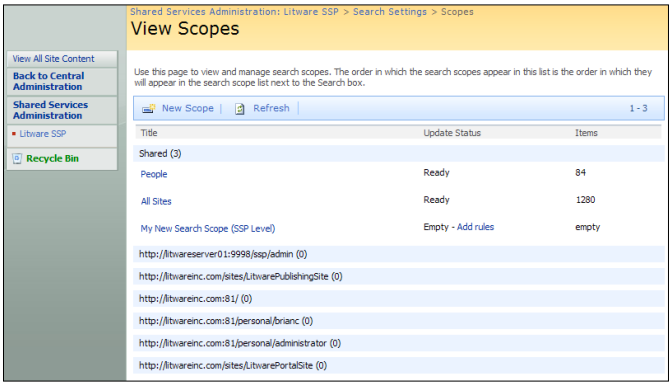


Configuring Search



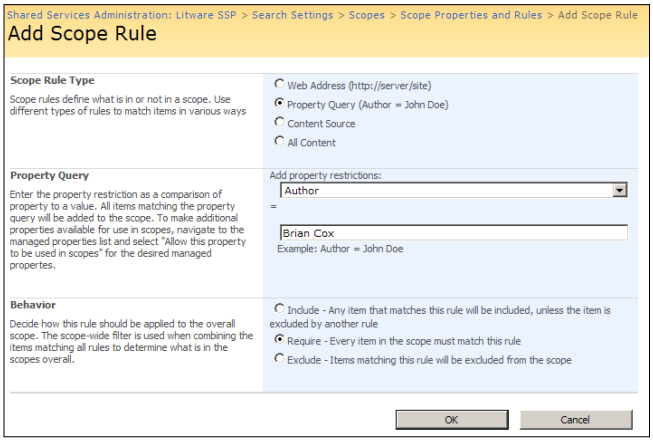
Search Scopes

- Created at one of two different levels
  - Can be created within context of an SSP
  - Can be created within context of a site collection



Adding Rules to a Search Scope

- Each search scope has one or more rules
  - Rules define criteria to include/exclude content



Search Center

- Customize and extend Search Center
  - Modify query parameters
  - Add tabs
  - Modify XSLT, CSS on results
  - Custom search Web Parts

Search Web Parts

- 9 OOB web parts including
  - Search Box
  - Core Results
  - High Confidence
  - Statistics
  - Pagination
  - Action Links
  - Matching Keywords and Best Bets
  - Search Summary (Did you mean?)
  - Advanced Search
  - Share data through hidden object
- Web Part properties such as
  - Formatting
  - From turning stemming on/off to the # of results returned
  - XSL

Customize UI With XSLT

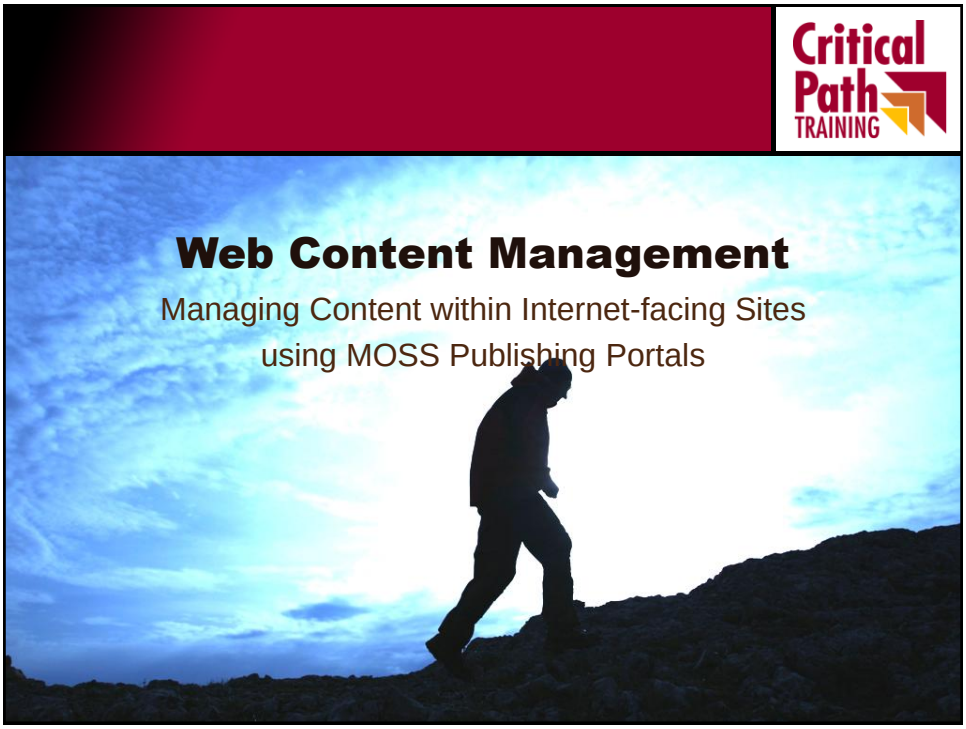
- Appropriate for scenarios requiring:
  - Change results layout, look and feel
  - Pivoting using the keyword syntax
- Web part property
- Power of XSLT
  - Formatting
  - Logic
  - Math

Extending Search

- Customizing the Query and Results
  - Query Object Model
- Customizing the Index
  - Index custom data - Protocol Handlers, IFilters, BDC
  - Custom Query Time Security Trimmer

Summary

- Collaboration Portals
- Shared Service Provider (SSP) Architecture
- Creating corporate portal sites
- User Profiles
- Audience Targeting
- MySites
- Extending Search



Agenda

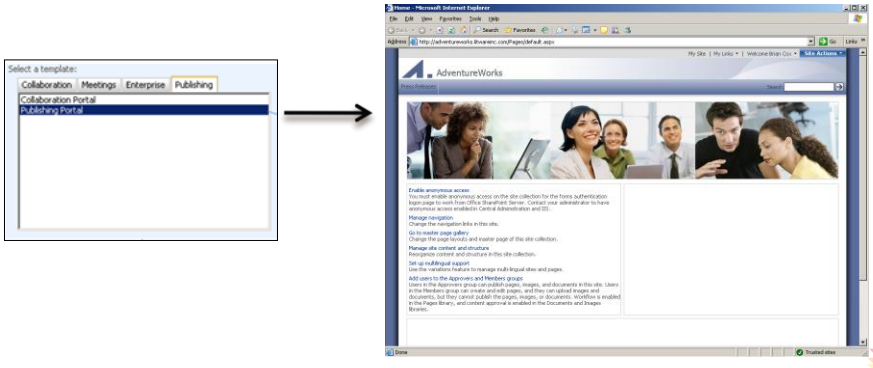
- The Publishing Site template
- The MOSS Approval Process
- Creating custom page layouts
- Converting Office documents
- Content Translation using Variations
- Optimization through Caching Profiles

MOSS WCM Features

- Branding
  - Define the look, feel, and navigation of the site
- Decentralized Authoring
  - Allow users to easily create and contribute content
- Workflow/Scheduling
  - Supervisors approve content before it is posted.
- Data Integrity
  - Enforce validation of content structure for publishing
  - Ensure content published/removed in timely manner

Creating A Publishing Portal

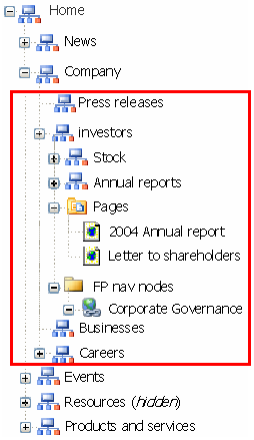
- Creating with WSS Central Administration
  - Create a site collection based on Publishing Portal

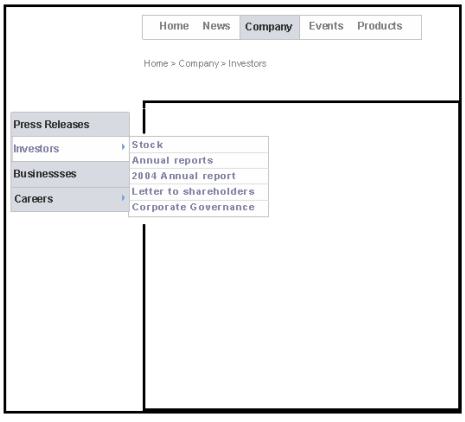


Site Hierarchy

- In the past a lot of confusion
  - Windows SharePoint Services 2003 → sites
  - SharePoint Portal Server 2003 → areas
  - Content Management Server 2002 → channels
- In SharePoint 2007 everything is a site

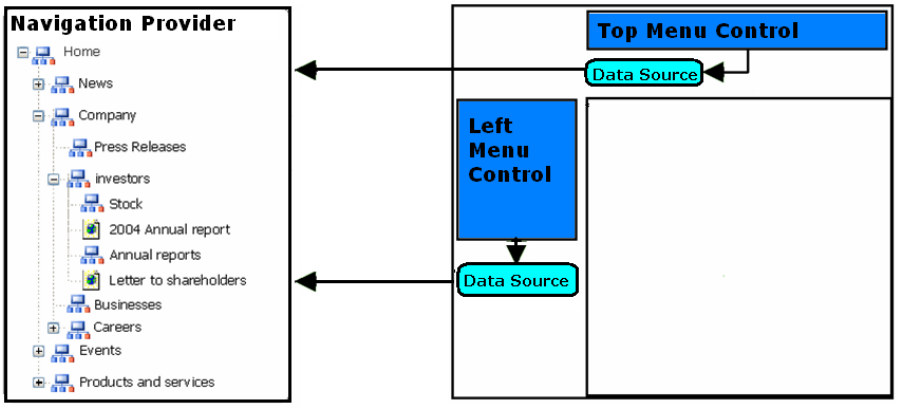
Navigation





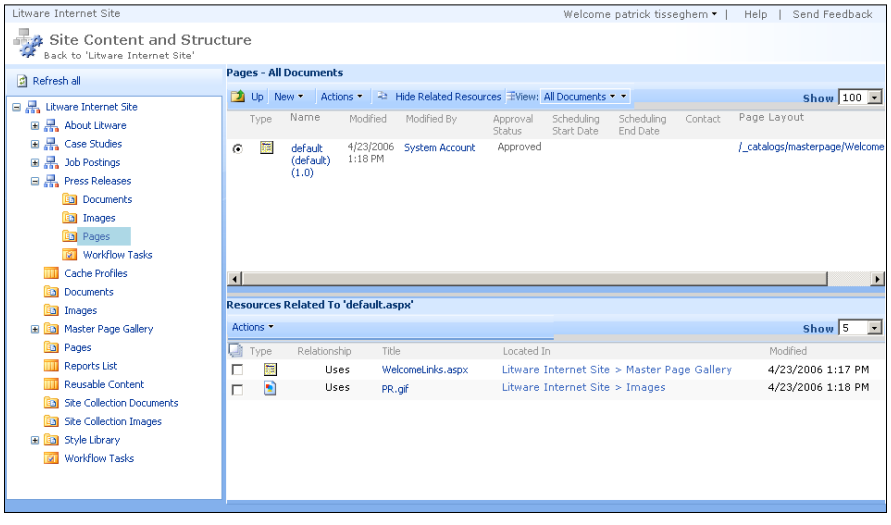
- Dynamic navigation based on site hierarchy
- Includes webs, pages and authored links
- Navigation links trimmed based on security, workflow state and publishing schedule

Navigation and ASP.NET



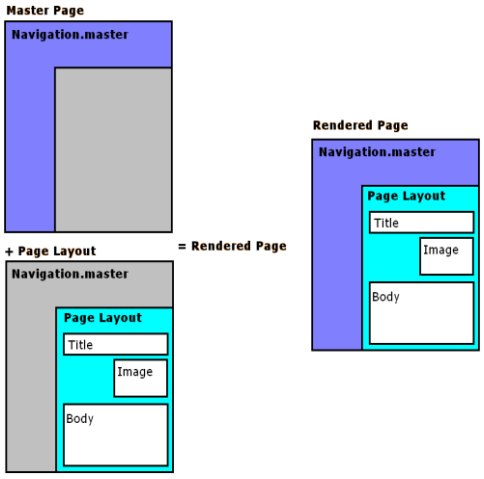
- Based on the ASP.NET 2.0 navigation model
- Works with standard ASP.NET 2.0 navigation controls
- Out-of-the-box Hierarchy navigation provider
- High performance: provider support runtime object caching

Site Content and Structure



Page = Master Page + Page Layout

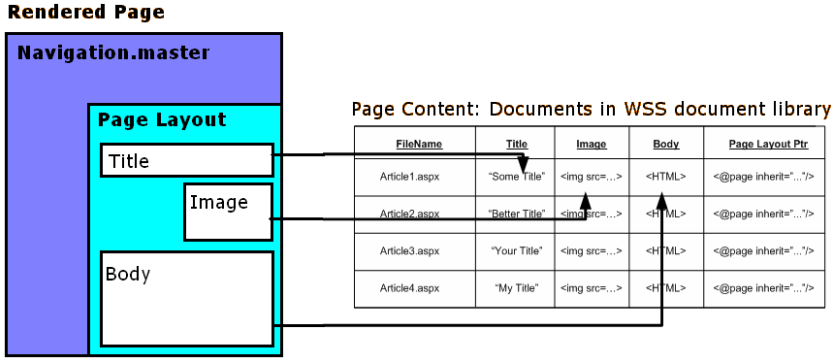
- Master page defines banner and navigation
- Page layout ASPX defines how page content is rendered
- Possible scenario
  - 1-3 Master pages
  - 10-25 Page Layouts
  - 10s of 1000s of Content Pages



Page Layouts

- Page execution:
- Page URL requested
- Page layout executed in content of page
- Content server controls bind to page fields
- Rendered page returned

Inherited from WSS:
- Versioning,
- Check-in/Check-out
- Content types
- Access control
- Workflow



Steps to Create a New Page Layout

- Create shared columns
- Create content type
- Add created site columns to content type
- In the Master Page Gallery
  - Create new Page Layout file
  - Check-out file and edit in SharePoint Designer
  - Populate the file with content fields
  - Check-in and approve
- Use the new page layout file

Steps to Create a New Page Layout

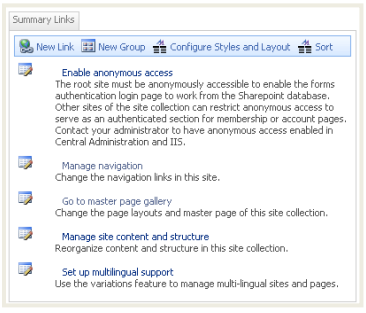
- Create shared columns
- Create content type
- Add created site columns to content type
- In the Master Page Gallery
  - Create new Page Layout file
  - Check-out file and edit in SharePoint Designer
  - Populate the file with content fields
  - Check-in and approve
- Use the new page layout file

Publishing Cycle

- Workflow based on Windows Workflow Foundation
- Light-weight approval workflow is active OOB
  - Based on approval
  - Minor versions need to be approved to become major versions
  - Visitors only see the major (published) versions
- Workflow can be replaced by custom workflow
  - OOB delivered with MOSS 2007
  - Designed using SharePoint Designer 2007
  - Created using Visual Studio.NET 2005

WCM Web Parts

- Summary Links Web Part
  - Custom annotated, stylized links



- Table of Contents Web Part
  - Displays navigation information of your site

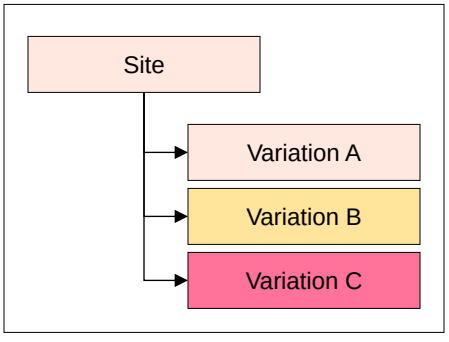
- Content Query Web Part
  - Displays a dynamic view of the content in your site

Multilingual sites

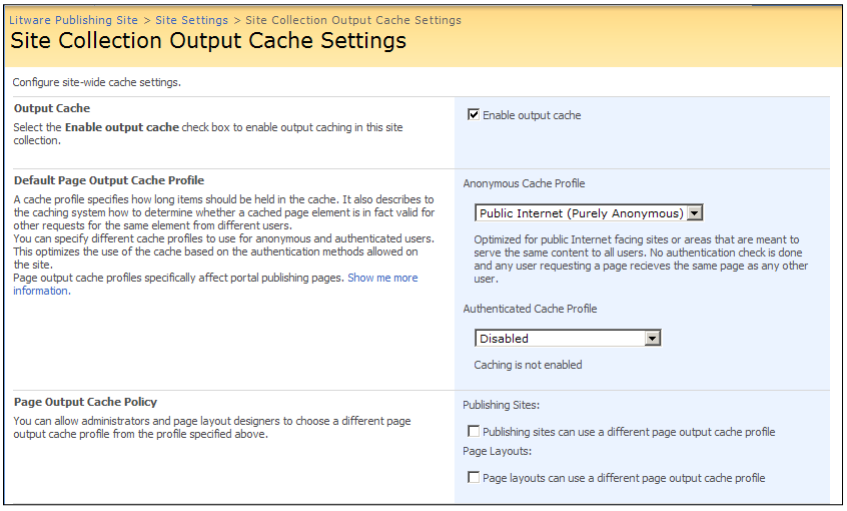
- Common pattern
  - Parallel sites in multiple languages
  - In concept, they are localized mirrors
  - In reality, there are exceptions and customizations for different regions
- Modeled as variations
  - Admin creates multiple labels
  - System creates and maintains parallel versions of containers and items
  - Exceptions are allowed
- Not just for language translations
  - Multilingual sites, multi-device sites, and multi-branded sites

Site Variations

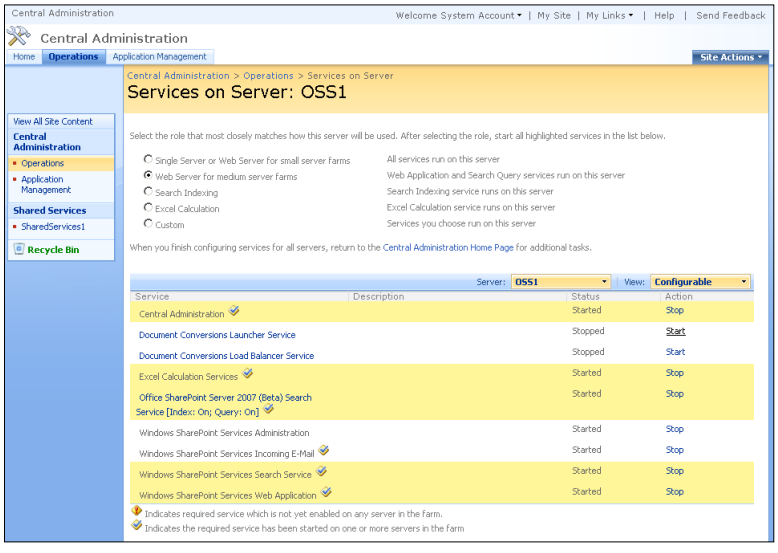
- Allow for publishing of related sites and pages
  - Multilingual scenario
  - Device targetting



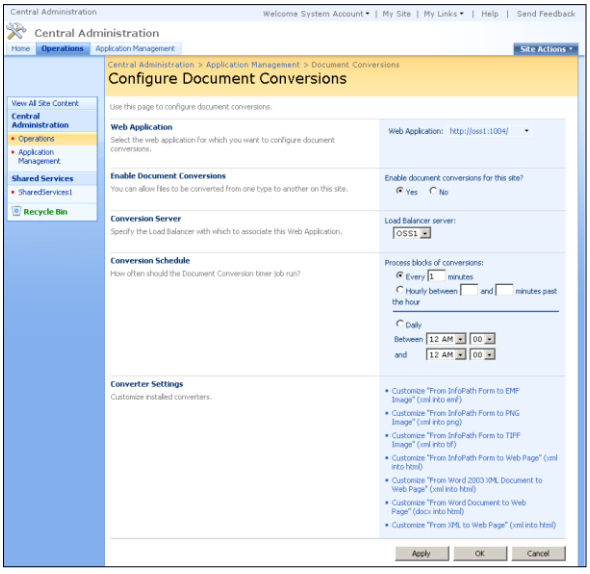
Profile Caches



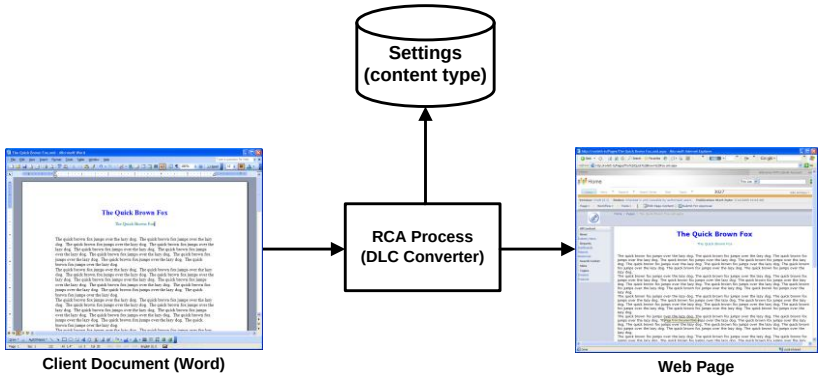
Configuring Document Conversion



Configuring Document Conversion



Smart Client Authoring

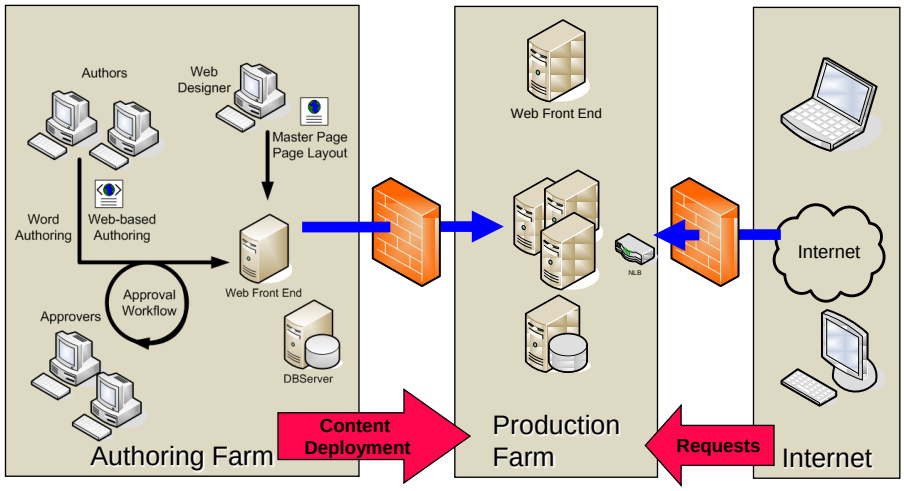


- Convert documents of different formats into web pages (server side)
- Linkage between document and page preserved
- Document Content Type stores info about how and where to create page
- Pluggable converter model
  - Builds on document services conversion infrastructure
  - Out-of-the-box converter for Word documents

Content Deployment

- Transfers content from one site collection to another
  - Paths define the relationship between source and destination
  - Jobs define the content to deploy and a schedule

Multi-Farm Topology



Summary

- The Publishing Site template
- The MOSS Approval Process
- Creating custom page layouts
- Converting Office documents
- Content Translation using Variations
- Optimization through Caching Profiles

Leveraging the Business Data Catalog

Surfacing backend data in SharePoint Server sites

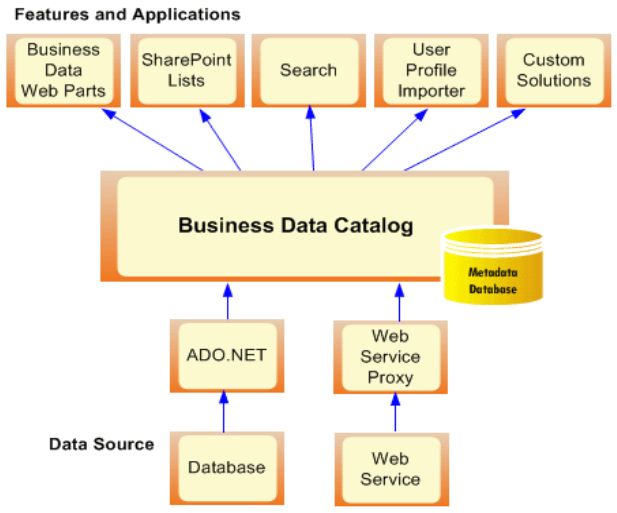
Agenda

- Motivation for the BDC
- Application Definition Files
- Application, Entities, Methods and Associations
- Using the built-in BDC Web Parts
- BDC integration with MOSS search
- Creating custom BDC Web Parts

Design Motivations

- Reduce the need for custom front-end code
  - Reuse SOA investments already in place
- Create manageable method for reusing data
  - Centralized deployment
  - Centralized data security
  - Low latency
- Designed for portal & collaboration scenarios
  - Data query, indexing, personalization
- The BDC is not about: transactions, workflow, data transformation, adapters, or precomposition

BDC Architecture



Application Definition Files

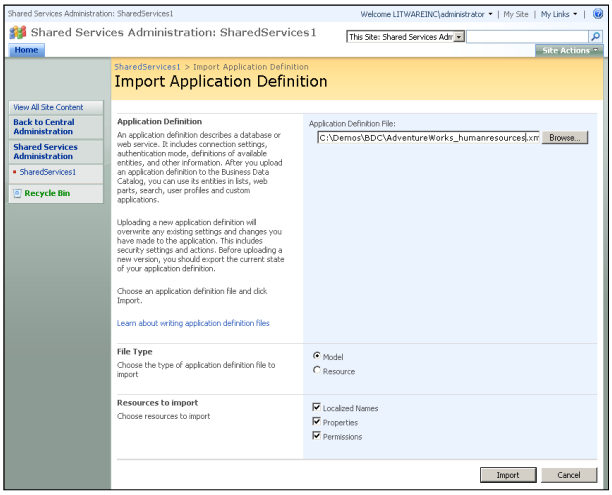
```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<LobSystem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.microsoft.com/office/sps/2005/bdcMetadata BDCMetadata.xsd"
 Type="Database" Version="1.0.0.0" Name="AdventureWorksSample"
 xmlns="http://schemas.microsoft.com/office/sps/2005/bdcMetadata">

 <LobSystemInstances>
    <LobSystemInstance Name="AdventureWorksSample">
      <Properties>
        <Property Name="AuthenticationMode" Type="System.String">PassThrough</Property>
        <Property Name="DatabaseAccessProvider" Type="System.String">SqlServer</Property>
      </Properties>
    </LobSystemInstance>
  </LobSystemInstances>

 <Entities>
    <Entity EstimatedInstanceCount="10000" Name="Product"/>
    <Entity EstimatedInstanceCount="10000" Name="SalesOrder"/>
    <Entity EstimatedInstanceCount="10000" Name="Customer"/>
  </Entities>

  <Associations>
    <Association AssociationMethodEntityName="Customer"
                 AssociationMethodName="GetSalesOrdersForCustomer"
                 AssociationMethodReturnParameterName="SalesOrders"
                  Name="CustomerToSalesOrder" IsCached="true">
      <SourceEntity Name="Customer" />
      <DestinationEntity Name="SalesOrder" />
    </Association>
  </Associations>
</LobSystem>
```

Tools for Building Definitions

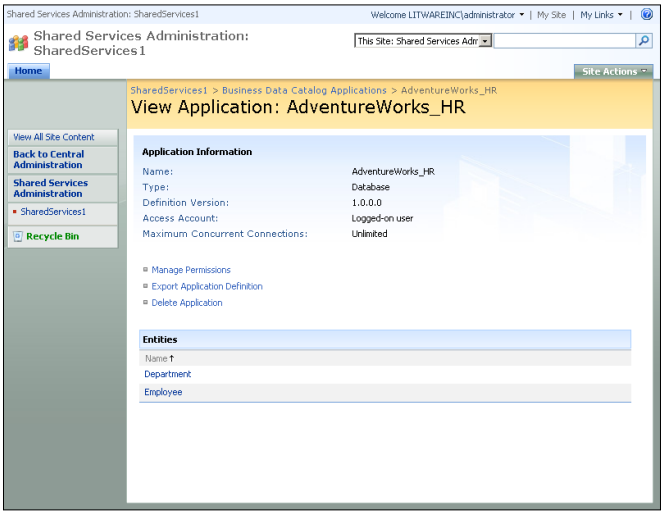
- BDC Meta Man
  - http://www.bdcmetaman.com/
  - Free version for SQL Server data sources
  - Licensed version adds Web service, Oracle, ODBC

    `Action builder`
    `Visual designer`
    `Multiple entity support`
    `Configuration of LOBSystems, LOBSystemInstances, Entities, Methods, etc.`
- Simego MOSS BDC Design Studio
  - http://www.simego.com/#MOSS
- Various efforts on CodePlex, GotDotNet
- Some SDK-delivered functionality is in the works

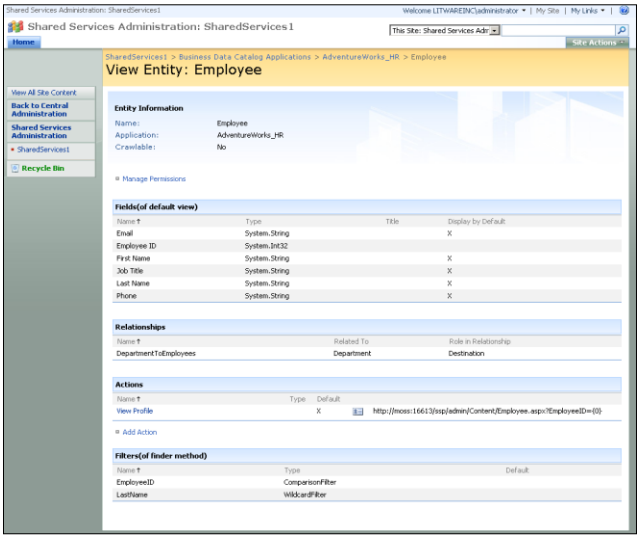
Importing Application Definition Files



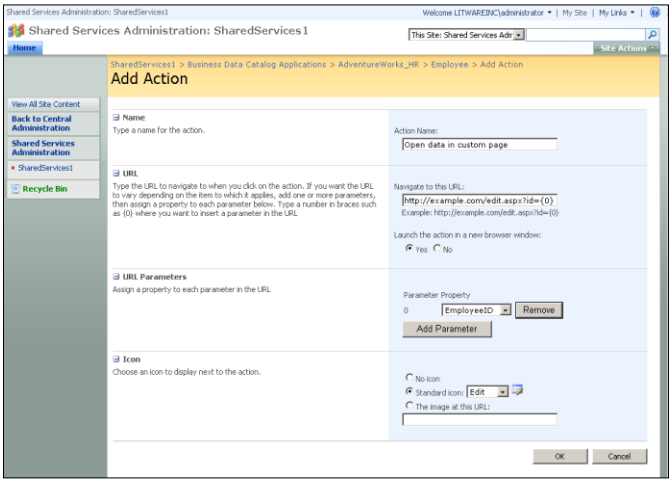
Administration of BDC Applications



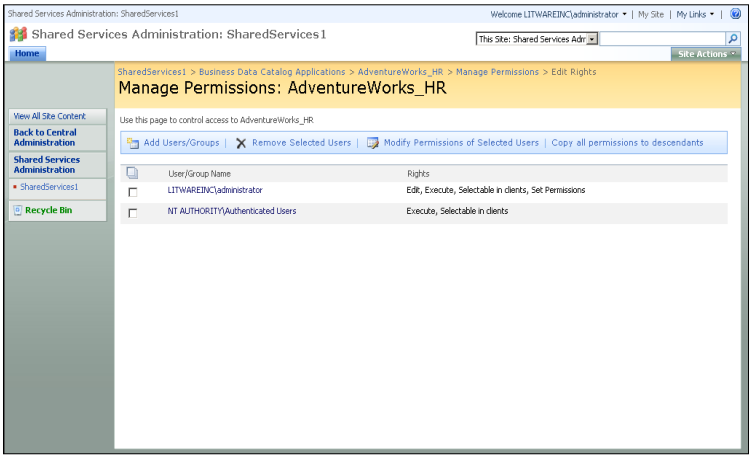
Examining BDC Application Entities



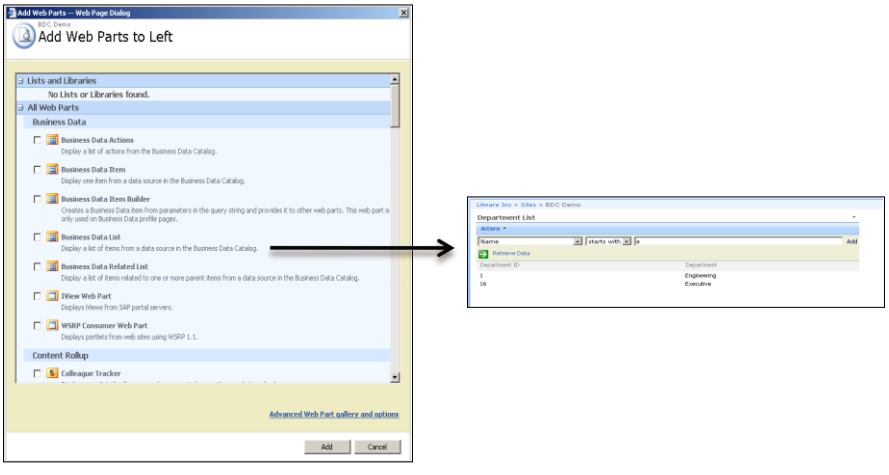
Adding Actions to an Entity



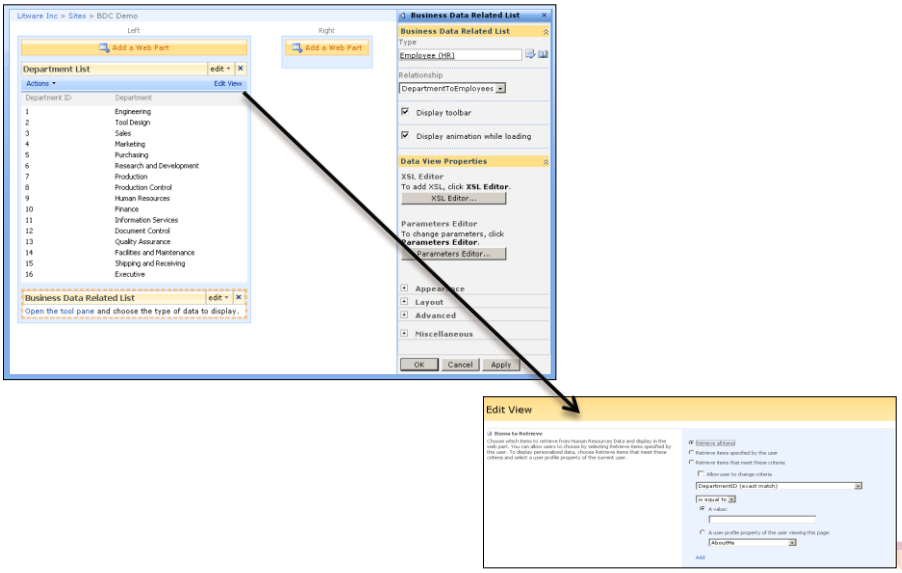
Administrating Security



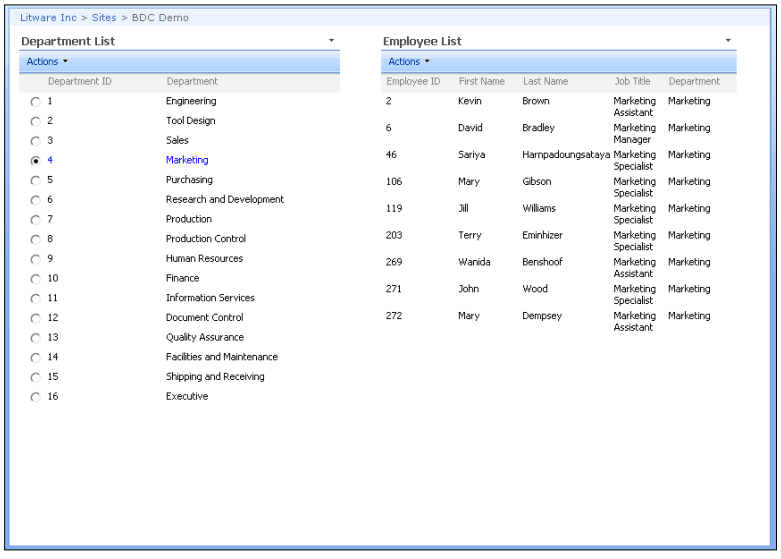
Using BDC Web Parts



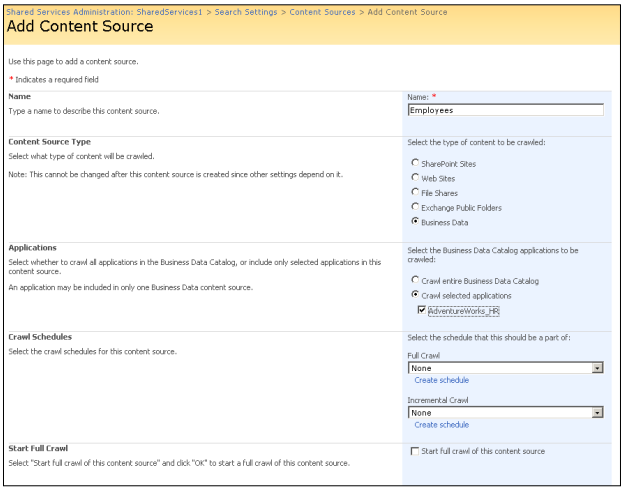
Editing BDC Web Parts



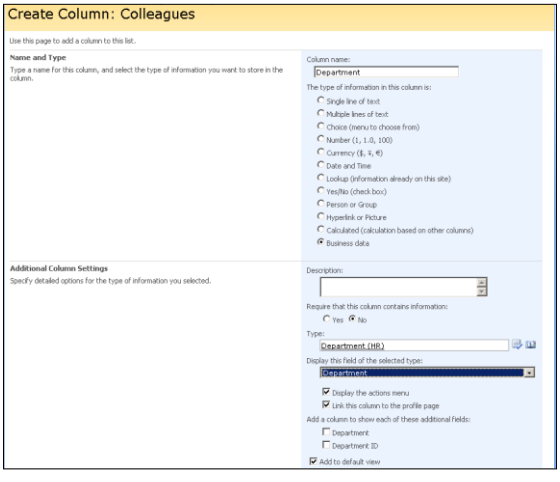
Connecting Web Parts with Associations



Searching through BDC Applications

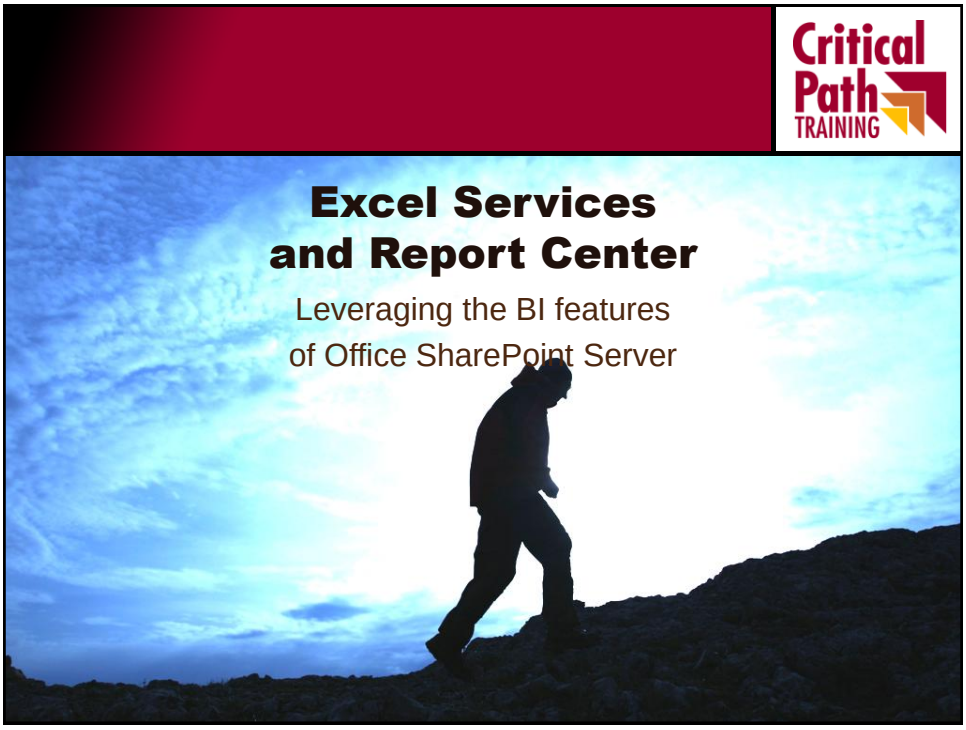


Adding BDC Columns to WSS Lists



Summary

- Motivation for the BDC
- Application Definition Files
- Application, Entities, Methods and Associations
- Using the built-in BDC Web Parts
- BDC integration with MOSS search
- Creating custom BDC Web Parts



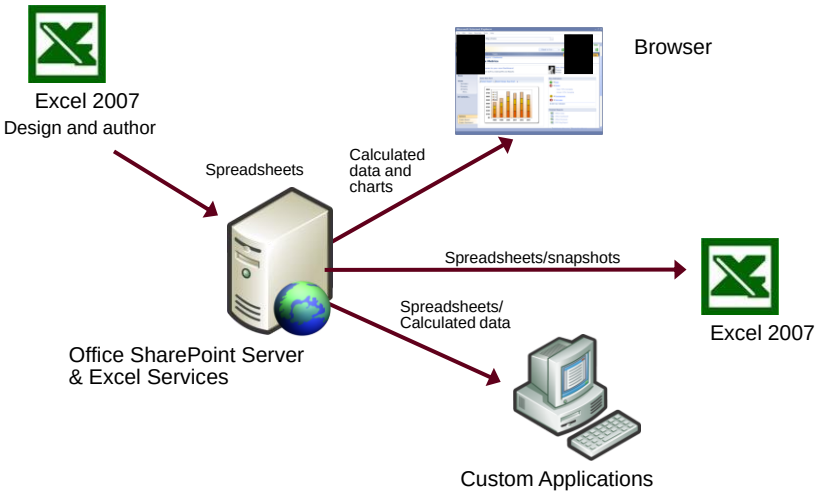
Agenda

- Excel Services
  - Publishing spreadsheets that render in the browser
  - Configuring Trusted Locations
  - Connections
  - Using user-defined functions (UDFs)
- Report Center
  - Creating Dashboards
  - Key Performance Indicators (KPIs)
  - Filters

Why Do We Need Excel Services?

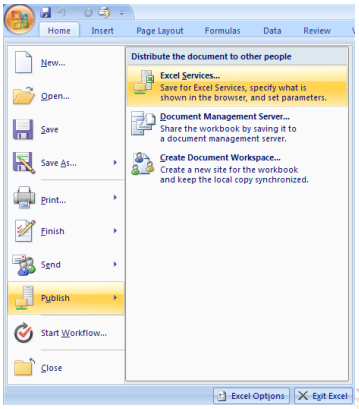
- Common customer requirements/complains
  - Distributing spreadsheets to users creates many copies
  - Excel doesn't play well in the BI dashboard and reporting world
  - It's difficult to protect proprietary information in spreadsheets
  - Incorporating Excel logic into applications is hard
  - Excel was designed as a desktop application
    (read: Excel really stinks as a server-side application)

Excel Services



Walk Through of using Excel Services

1. Add the URL of the document library as trusted location
   1. SharePoint Central Administration
   2. In Application Management, configure the Farm's core services
   3. In Excel Services Management, add the URL of the doc lib as a trusted file location
2. In Excel 2007, publish to Office Server
   1. Decide what worksheets to publish
   2. Named ranges can be dynamically populated with values in the browser
   3. Give the URL of a document library
3. Users now have Web access to the spreadsheet

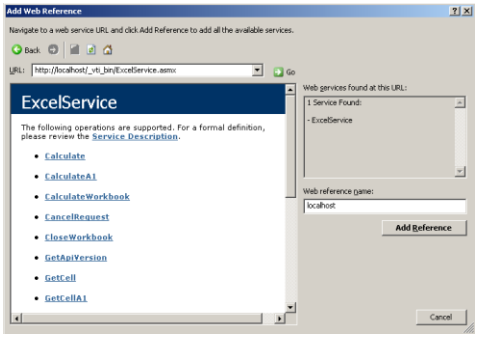


Excel 2007 Web Services

- Using server-side Excel logic in applications
  - Author part of the business logic in Excel
  - Protect and maintain proprietary information
- Automating spreadsheet updates on servers
  - Refresh external data and parameterize
  - Process generated spreadsheets
  - Create, store and deliver snapshots
- Custom UI to server-side Excel calculation

Excel 2007 Web Services

- Follow the previous steps to publish a spreadsheet to Office Server
  - Add Web Reference to your .NET application

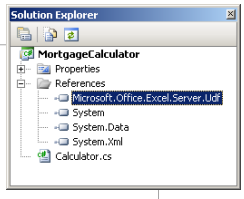


Excel 2007 Web Services

```
//-- create instance of proxy and take care of authentication
ExcelService ws = new ExcelService();
ws.Credentials = System.Net.CredentialCache.DefaultCredentials;

//-- open the Excel workbook
Status[] status = null;
string sessionID = null;
sessionID = ws.OpenWorkbook(textBoxURL.Text, "en-US", "en-US", out status);

//-- set the cell values
status = ws.SetCellA1(sessionID, "Mortgage Calculator", "CustomerName", textBoxCustomerName.Text);
status = ws.SetCellA1(sessionID, "Mortgage Calculator", "MortgageAmount", textBoxAmount.Text);
status = ws.SetCellA1(sessionID, "Mortgage Calculator", "InterestRate", textBoxInterestRate.Text);
status = ws.SetCellA1(sessionID, "Mortgage Calculator", "MortgageLength", textBoxLength.Text);

//-- calculate the workbook and get result
status = ws.CalculateWorkbook(sessionID, CalculateType.Recalculate);
object result = null;
result = ws.GetCellA1(sessionID, "Mortgage Calculator", "Payment", true, out status);

//-- display result and close workbook
if (result != null)
    MessageBox.Show("You pay " + result.ToString());
status = ws.CloseWorkbook(sessionID);
```

Using UDFs with Excel 2007 Services

- UDF definition
  - Methods in .NET classes, callable from Excel formulas




```
using System;
using Microsoft.Office.Excel.Server.Udf;

namespace MortgageCalculator {

  [UdfClass]
  public class Calculator {

    [UdfMethod]
    public double CalculateMortgage(int salesPrice, int mortgageLength,
                                    double downPaymentPercentage,
                                    double annualInterestPercentage) {
      // implementation
      double financed = (1 - downPaymentPercentage) * salesPrice;
      int nrOfMonths = mortgageLength * 12;
      double monthlyInterestRate = annualInterestPercentage / 12;
      return financed *
             (monthlyInterestRate /
             (1 - Math.Pow((1 + monthlyInterestRate), nrOfMonths * -1)));
    }
  }
}
```

Report Center

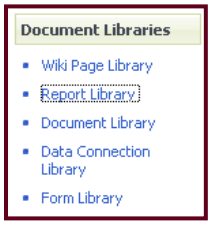
- With Office 2007, SharePoint Server become the hub for BI on the server
  - Excel Services and Reporting Services in the portal
  - Out-of-the-box BI portal experience
  - Dashboards, KPIs, and Report Libraries
  - Integrated with Portal, Collaboration, Enterprise Content Management and Workflow functionality

SQL Server Reporting Services

- Integrated with WSS to enable publishing, viewing, and management of reports
- Microsoft Office SharePoint Server light up
  - Report library integration
  - Dashboards and filter Web Parts
- Integration is interesting because…
  - New capabilities for Reporting Services users
  - Great example of deep integration for ISVs

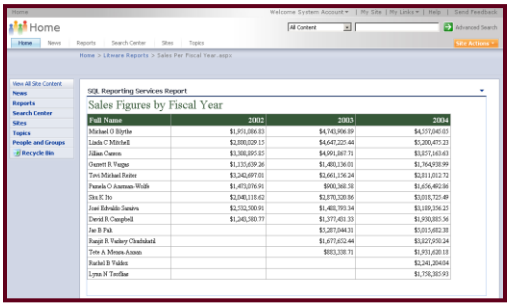
Storing Reports

- Reports have more specific needs than documents
  - History is very important
  - Many instances of the same report can exist
- Therefore, Office Server adds a Report Library template
  - Displays current spreadsheet / report by default
  - History available via search and list views
  - Custom profile page
  - Can be part of Report Center
  - A list template that can be used anywhere



Viewing Reports

- Works in both non-integrated mode (via IFRAME) as well as integrated mode
- Remote render in response though callbacks via the click handler
  - .RDL files are registered in SharePoint Services



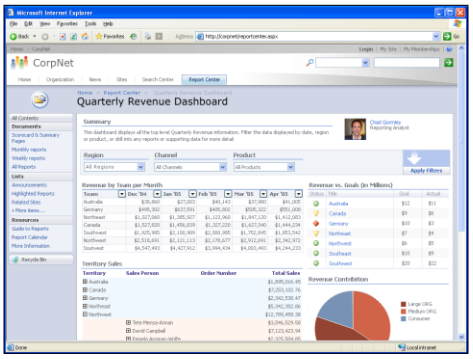
- On rendering, callback to SharePoint Services object model to synchronize contents
- Consumes filter part to part connection for specifying report parameter values

Key Performance Indicators

- Goals
  - KPIs can be an important instrument in the organization, so let's make it easier to create them
  - And let's use the environment we already know – the portal and team sites
  - Support KPI types from simple to enterprise class
- Types
  - Manually entered
  - SharePoint list
  - Excel workbook
  - SQL Server Analysis Services
- Technologies
  - KPI web part & list
  - Customizable KPI profile page
  - Dashboard template focused on KPIs

Dashboards

- Dashboards are SharePoint pages
- Dashboard pages are in same document library as spreadsheets and reports
- Types
  - Generic dashboard
  - KPI focused



Filtering

- Filtering is the natural next step after building a dashboard
- Filter for eastern region, last quarter
- Automatically show just *your* customers when you load page
- Accept values from query string
  - http://server/dashboard.aspx?Product=452

Filtering Web Parts

Display options

- Type in value
- Pick from list
- Tree view
- Hidden

Extensibility

- Custom providers & consumers
- Standard interfaces that ship in WSS

Filter value sources

- User entered value
- Manual list
- SharePoint list
- Analysis Services
- Bus. Data Catalog
- SharePoint profile
- Query string

Summary

- Excel Services
  - Publishing spreadsheets that render in the browser
  - Configuring Trusted Locations
  - Connections
  - Using user-defined functions (UDFs)
- Report Center
  - Creating Dashboards
  - Key Performance Indicators (KPIs)
  - Filters

Agenda

- User Authentication
  - Windows Authentication
  - Forms Authentication
- WSS Identities and Security Contexts
  - Application Pool Identity
  - The SHAREPOINT\System account
  - Escalation of Privileges
  - Delegating User Credentials
- The MOSS Single Sign-On Service (SSO)

Security Concepts 101

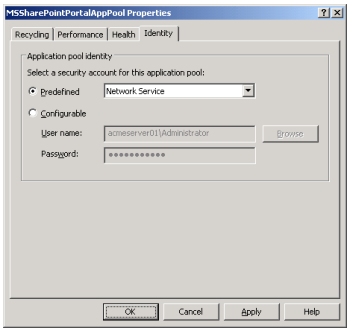
- Authentication
  - Verifying the identity of whom someone claims to be
  - Based on using credentials to verify identity
- Authorization
  - Process of determining what an identity has access to
  - Takes place after authentication

WSS Identities

- WSS Application Pool Identity
  - Configured with IIS or WSS admin tools
  - Authenticated when worker process is launched
- WSS System Identity
  - Used by WSS to hide application pool identity
- User Identity
  - Authenticated by Windows or another authentication provider
  - Used for authorization and auditing

WSS Worker Process Identity

- WSS runtime is hosted by IIS Application Pools
  - Each WSS Web Application runs in an IIS Web site
  - Each IIS Web site runs with in a specific IIS application pool
  - Application pool identity configured with local or domain account
  - Domain account recommended in farms of two or more servers



WSS Authentication with SQL Server

- WSS system code must access SQL Server
  - WSS must create and access the configuration database
  - WSS must create and access content databases
- SQL Server must authenticate WSS identity
  - Can be configured with Integrated Windows Authentication (best)
  - Can be configured with standard SQL authentication
- WSS accesses SQL Server using App Pool Identity
  - When using Integrated Window Security, SQL Server authenticates WSS using the Application Pool Identity
  - WSS Identity must be in SQL Server roles of Database Creators and Security Administrators

The WSS System Account

- Some WSS operations require more permissions than the user has
  - WSS will sometimes run code under IIS app pool identity
  - IIS app pool identity has full control of the WSS content database
  - Exposing IIS app pool identity poses security risks
  - WSS "V3" introduces a new WSS System Account
- The System account has the login name: SHAREPOINT\system
  - SID as S-1-0-0 (Null SID)
  - User id as 1073741823 (0x3FFFFFFF)
  - When a list item is created by the application pool identity, it will show as “created by SHAREPOINT\system

WSS User Authentication Options

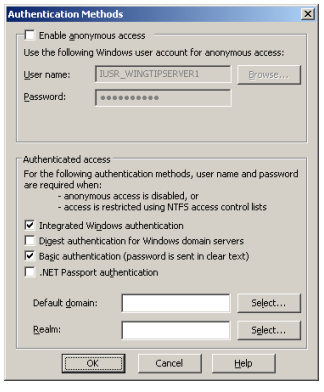
- Windows Authentication
  - IIS performs authentication with client
  - Users authenticated to Windows account (AD or local)
  - Only type supported in WSS V2 and SPS 2003
- ASP.NET Forms Authentication
  - Based on ASP.NET 2.0 authentication provider FX
  - IIS configured for anonymous access

Authentication And WSS Zones

- WSS partitions user authentication into zones
  - 1 Zone = 1 WSS-extended Web Application
  - Each zone is based on its own IIS Web site
  - Each zone has its own web.config file
- Zones can have only one authentication provider
  - Choose between (1) Windows, (2) FormsAuth or (3) WebSSO
- Can WSS sites support multiple authentication types?
  - Yes, two WSS Web Applications can point to the same content
  - More on this later in this lecture

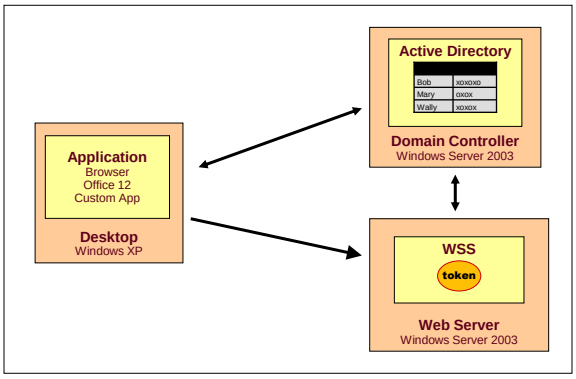
Configuring Windows Authentication

- Authentication performed against Windows accounts
  - Local Accounts can be used in single-server configurations
  - Active Directory accounts are usually a much better choice
- Popular Authentication types
  - Windows Integrated Authentication
  - Basic Authentication



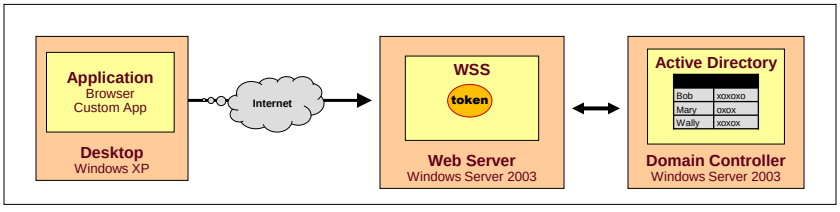
Integrated Windows Authentication

- Authentication using Windows protocols
  - Enhancements to WSS V3 enable Kerberos protocol
  - WSS V3 still uses NTLM protocol when necessary
  - Authentication results in creation of Windows security token



Basic Authentication

- Commonly used in Internet scenarios
  - Industry-standard, firewall-friendly protocol for authentication



  - No need for client to access Windows domain controller
  - Authenticates to Windows account and creates security tokenUser name and password passed in clear text
  - Must use HTTPS for any reliable level of security

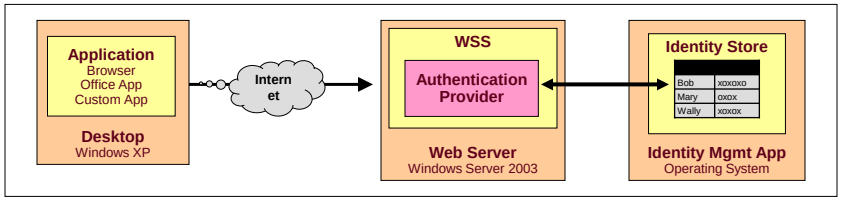
Windows Authentication Zones

- The web.config file configures basic ASP.NET settings
  - Authentication specifies resolving to Windows accounts
  - Impersonation is set to true
- WSS is the entity that adds authorization
  - The following markup shows ASP.NET configured to allow all users access to everything

```
<!-- selected snippets from web.config for integrated Windows auth -->
<configuration>
  <system.web>
    <!-- use Intergated Windows Authentication -->
    <authentication mode="Windows" />
    <!-- impersonate Windows user -->
    <identity impersonate="true" />
    <!-- configure ASP.NET to grant all access to resources -->
    <authorization>
      <allow users="*" />
    </authorization>
  </system.web>
</configuration>
```

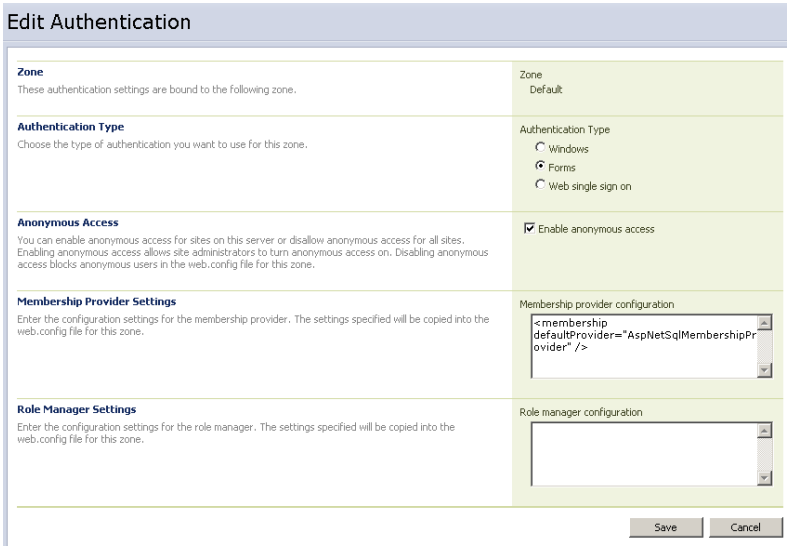
ASP.NET Forms Authentication

- WSS V3 supports ASP.NET forms authentication
  - Allows you to authenticate without requiring Active Directory
  - Based on ASP.NET 2.0 pluggable authentication providers
  - ASP.NET 2.0 role providers can optionally be used as well



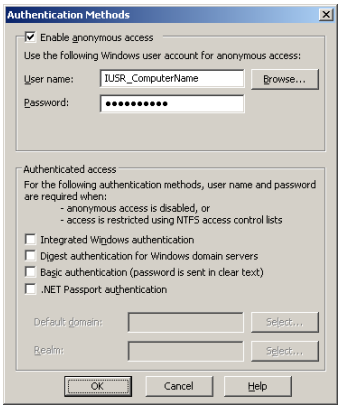
- Out-of-the-box Authentication providers
  - ASP.NET 2.0 SQL Server authentication provider
  - LDAP Authentication provider

WSS-supplied Configuration UI



Configuring Forms Authentication

- Forms Authentication requires anonymous access
  - All users run under the identity of a single Windows account
  - Identity is configurable within IIS administration tools



Forms Authentication Zones

- The web.config file configures basic ASP.NET settings
  - Authentication configured for Forms
  - Membership provider is configured
  - Impersonation is set to true (e.g. impersonated IUSR_XXX)

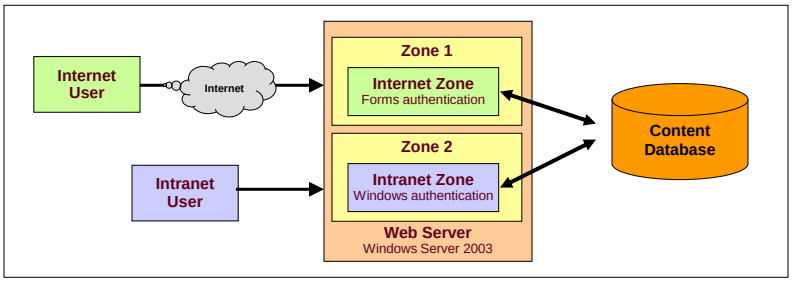
```
<!-- selected snippets from web.config for Forms auth -->
<configuration>
  <system.web>
    <!-- use Forms Authentication -->
    <authentication mode="Forms">
      <forms loginUrl="/_layouts/login.aspx" />
    </authentication>
    <!-- configure membership provider -->
    <membership defaultProvider="AspNetSqlMembershipProvider" />
    <!-- impersonate anonymous access user -->
    <identity impersonate="true" />
    <!-- configure ASP.NET to grant all access to resources -->
    <authorization>
      <allow users="*" />
    </authorization>
  </system.web>
</configuration>
```

Forms Authentication By Smart Clients

- A truly clever smart client could
  - Recognize a HTTP redirect request
  - Open a browser window for the login form
  - Submit the form content back to the server
- But
  - It can't confidently know that the redirect request is for a logon form URL
- Therefore
  - Users must first visit the site with a browser
  - Server must issue a (short-lived) persistent cookie
- This affects all Office System client applications

Multiple Authentication Types

- Two WSS Web Applications can access the same sites
  - Configure Web Applications to point to the same content database
  - Makes it possible to support multiple authentication providers



- An important issue with access control configuration
  - ACLs will mix up users from different authentication systems
  - Authorization for users using 2 zones must be configured twice

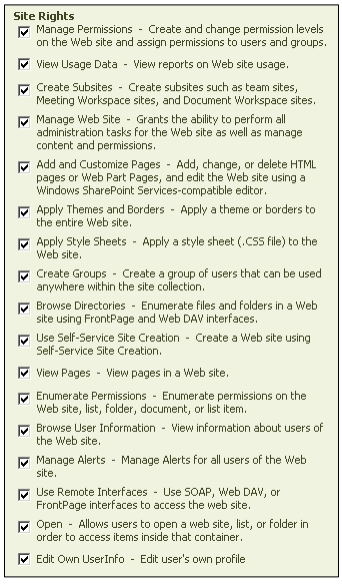
Authorization and Access Control

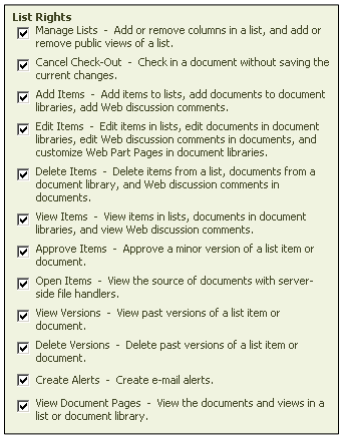
- Authorization involves granting rights to identities
  - Rights are what users can and cannot do in a site
- Abstractions make configuring authorization easier
  - Users and Groups from authentication provider
  - WSS Groups
  - WSS Permission Levels (a.k.a. roles)

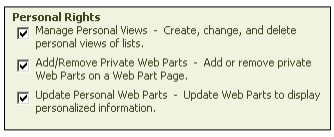
Sites and Site Collections

- A Site Collection is an autonomous unit of security configuration
  - Security configuration is not shared across site collections
- With a site collection…
  - Site collection owners have full administrative rights
  - Site collection administrators have full administrative rights
  - Rights for other users are configured on a per-site basis

Permissions Managed Using Rights

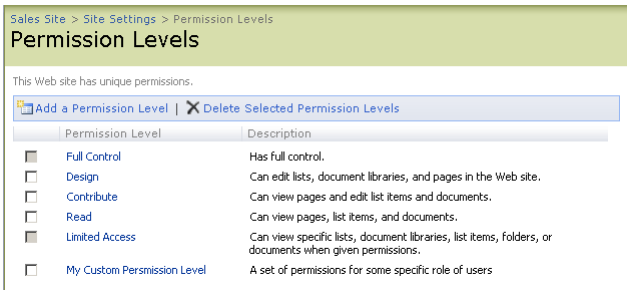






Permission Levels

- WSS rights are managed through permission levels
  - Each permission level consists of a set of rights
  - Permission levels define rights required by business roles
  - They are defined on a per-site basis
  - Permissions are assigned to people and groups



People

- WSS V3 introduces people to visual identities associated with the current site

Groups

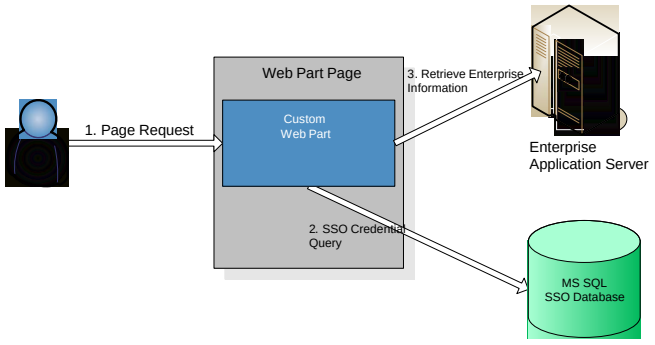
- WSS introduces groups to visually aggregate groups from many places
  - WSS-created groups
  - Active Directory groups
  - Local groups
  - Groups from a custom role provider

Levels of Rights Assignment

- Access control to content is configured at many levels
  - Site Collection
  - Site
  - List or Document Library
  - Item or Document
  - Child Site

Single Sign-On

- Storage and Mapping of Credentials
  - Used for Accessing Enterprise Applications
  - Serves to reduce the Number of Credential Prompts
  - Leveraged through custom Web Parts and BDC



Summary

- User Authentication
  - Windows Authentication
  - Forms Authentication
- WSS Identities and Security Contexts
  - Application Pool Identity
  - The SHAREPOINT\System account
  - Escalation of Privileges
  - Delegating User Credentials
- The MOSS Single Sign-On Service (SSO)