WCM401 Slide Manual

Developing Publishing Sites with SharePoint Server 2007 Web Content Management

Schedule of Lectures

1) Windows SharePoint Services 3.0 Primer
2) MOSS 2007 & Web Content Management Overview
3) Authentication & Authorization
4) Creating Master Pages & Configuring Navigation
5) Creating Custom Page Layouts
6) Extending the Out-Of-The-Box Authoring Experience
7) Custom Field Types & Controls
8) Leveraging Publishing & Custom Web Parts
9) Performance Tuning Publishing Sites
10) Understanding Workflow Foundation & Creating Custom Workflows
11) Content Deployment
12) Implementing Multilingual Sites Using Variations

Windows SharePoint Services 3.0 Development Primer

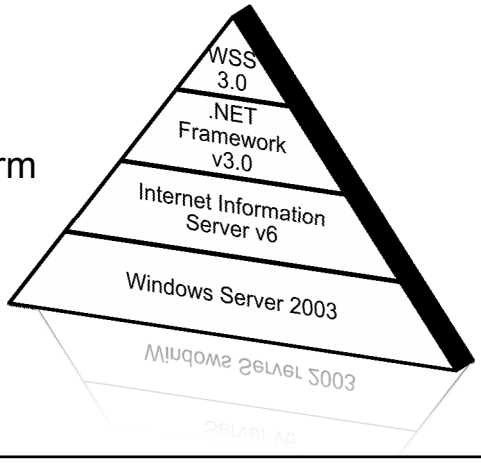
Refresher & Object Model Primer of WSS 3.0

Agenda

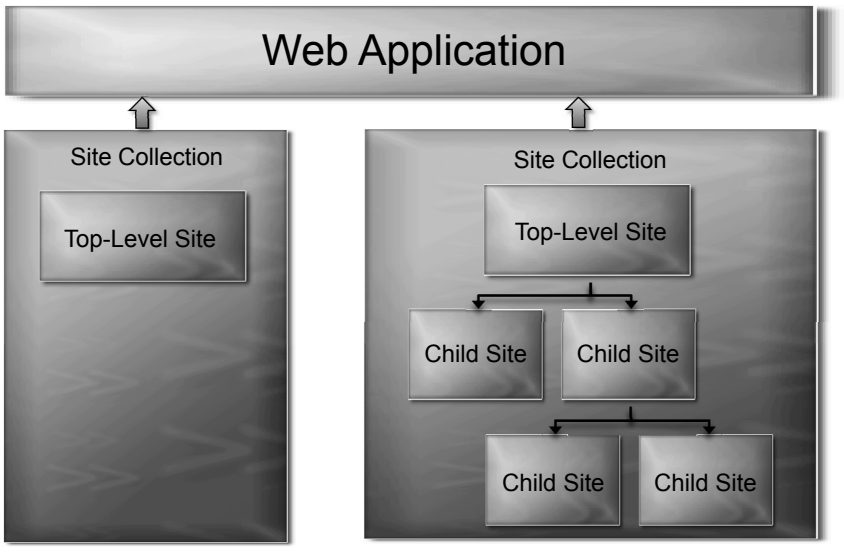
- Introduction to Windows SharePoint Services 3.0
- Overview of WSS architecture
- Overview of WSS collaboration capabilities
- SharePoint customization vs. development
  - Customized / unghosted vs. uncustomized / ghosted
  - Templates vs. instances
- Overview of SharePoint Features
- Overview of WSS solution packages
- Working with the `Microsoft.SharePoint` Namespace

Introduction to WSS 3.0

- Windows SharePoint Services 3.0
  - Site provisioning engine
  - Core collaboration services
  - Application development platform
  - Licensing: Part of Windows Server 2003



SharePoint Logical Architecture



SharePoint Logical Arch. (Part 2)

- Administrator creates Web application & provisions it with WSS 3.0
  - Used as the HTTP entry point
- Web applications contain site collections
- Site collections have exactly 1 top-level site
- Site collections can contain a hierarchy of sites
- Site collection content is stored in a content DB

SharePoint Physical Architecture

- SharePoint content resides in SQL Server DB's
- One central database contains farm configuration data
- Different sized farms:
  - Small: WSS & SQL installed on same server
  - Medium: WSS & SQL installed on separate servers
  - Large: WSS installed on multiple application servers & Web Front End (WFE) servers for load balancing & all connect to a central SQL server

Implications Of .NET 3.0 As The Foundation

- Unlike the previous version, WSS 3.0 is built on top of the .NET Framework 3.0
- All capabilities provided by ASP.NET 2.0 bleed through to WSS v3
  - Web Part framework
  - Navigation provider model
  - Pluggable authentication via authentication providers
  - Master page – content page infrastructure
  - Can create custom HTTP handlers & modules
  - Caching
  - No ISAPI filter!!!
- Windows Workflow Foundation

Creating a New WSS Site Collection

- Site collections can be created via:
  - Central Administration Web site
  - STSADM.EXE
  - SharePoint API

```
C:\>stsadm -o CreateSite -url http://litwareinc.com/sites/WCM401 -owneremail adm
inistrator@litwareinc.com -sitetemplate STS#0 -title "Sample Blank Site" -ownerl
ogin litwareinc\administrator
```


```
SPSite site= new SPSite("http://foo.com");
SPWeb web =
  site.AllWebs.Add("http://foo.com/sites/bar");
web.Title = "A new site";
web.Update();
```

WSS 3.0 Sites

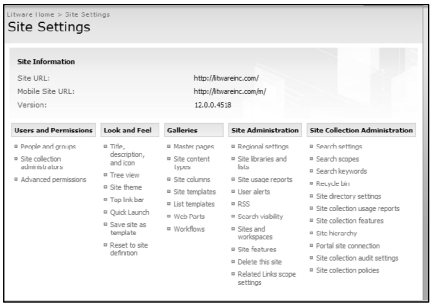
- Site owners manage content & security
  - Can delegate creation / management of content & security permissions to other users
- Top-level sites can contain child sites
  - Child sites can also have subsites, and so on

SharePoint Administration

- Central Administration Web Site
  - Special WSS v3 Web site available to farm admins
  - Can do most admin functions, but not all
  - MOSS adds additional links for additional admin functionality
  - Extensible (just another SharePoint site)…
  - Can not be personalized
- STSADM.EXE command line utility
  - Can do everything Central Administration Web site can do, and more
  - Commands can be automated / scripted via batch files
  - Extensible… developers can add custom commands

Site Administration – Site Settings

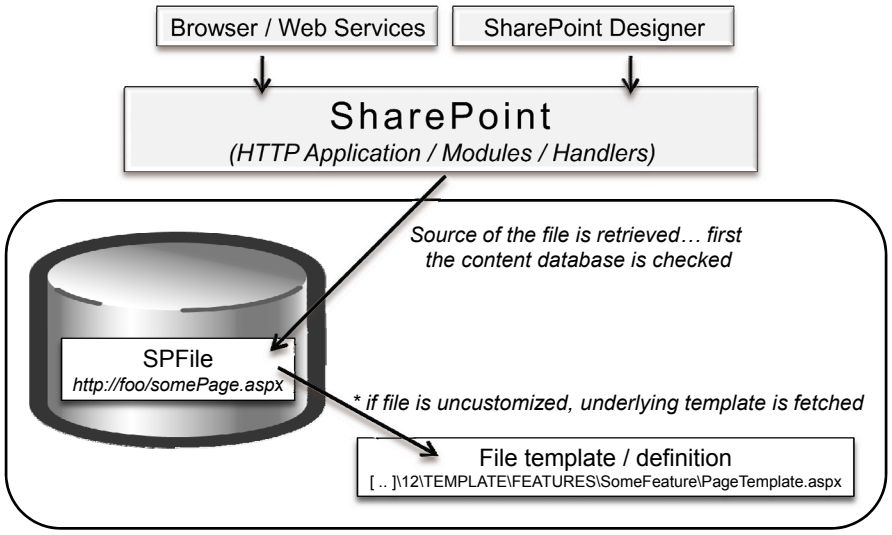
- All WSS sites have a site settings page, accessible via the Site Actions menu
- Contains links to admin pages (security trimmed)
- Top-level site in a site collection has extra site collection administration links



Concept of Templates vs. Instances

- Pages in a site exist in the site object model as `SPFiles`
- All pages live within folders (`SPFolder`)
- Concept of customized vs. uncustomized files
  - Uncustomized Page – Appears in site's hierarchy, but content of page resides on the file system
  - Customized Page – Appears in site's hierarchy, but content of page resides in the site's content DB

Uncustomized vs. Customized Files



Typical Site Development

- Infrastructure assets
  - Create site column and content types via browser
- Layout assets
  - Create master pages and page layouts with SharePoint Designer 2007
  - Modify look and feel, adding CSS and images using SharePoint Designer 2007

"Big Picture" Challenges

- How do you make this development process repeatable? How to automate it?
- What about source control?
  - SharePoint has version control on lists and libraries
  - No version control for infrastructure assets
  - No version control for site pages and files
- What about a true software development lifecycle process?
- What about change management process?

SharePoint Customization

- Creating and editing **instances** of assets that live exclusively in the site's content DB
- Even if originally based off a template, source still lives in content DB
- Tools used in SharePoint customization
  - SharePoint API
  - SharePoint browser interface
  - SharePoint Designer 2007

SharePoint Customization

- Advantages
  - Plenty of resources documenting the process
  - WYSIWYG development with SharePoint Designer 2007
  - Easily make changes in multiple environments
- Disadvantages
  - Site columns and content types are not easily moved between environments
  - Challenging to package and deploy

```
New files and changes to existing files
Multiple environments
```

Challenges In Creating Publishing Sites

- Publishing sites exacerbate the challenges
- All projects require sharing assets
  - Between developers on project team
  - Between environments (Dev / QA / UAT / Prod)
- How to move asset change to other environments?
  - Manually move infrastructure and layout assets
  - Backup / restore *(*** bad bad bad option! ***)*
  - Content deployment
- Does not promote code reuse!

SharePoint Development

- Goals
  - Developers work at a lower level (file system)
  - Keep infrastructure / layout assets out of DB
- SharePoint Feature schema allows creation of all kinds of assets
- Upon activation, Feature creates **uncustomized instances** in the site
- SharePoint solutions enable packaging of Features in one portable file

SP Development: Disadvantages

- SharePoint Features are tedious to build
  - No visual designer
  - Lots of CAML (XML) markup
  - Minimal debugging support
- Provisioning files requires double development
- Feature deactivation leaves artifacts
- Current tools do not promote this approach

SP Development: Advantages

- Developers stay in Visual Studio
- Keeps layout files uncustomized on server
- Easy to package changes in order to
  - Share with other members on project dev team
  - Deploy to other environments
- Fully leverage existing SCM solutions
- Easier to adhere to SDL and change control

Tips And Tricks To SP Dev

- Add CAML IntelliSense to Visual Studio
  - via Visual Studio XML Schema Cache
- Using existing tools as IDE, then “Featurize”
  - Use browser for site column and content type creation
  - Use SharePoint Designer 2007 to create master pages and page layouts faster
  - Utilize SharePoint API to extract assets

    `STSADM.EXE Custom Commands`
- Automate process of building WSP’s

DEMO: Page Templates vs. Instances

- Demo: Create a page using Office SharePoint Designer 2007
- Demo: **WssCreatePages**
  - Create a page & add to site via Feature
  - Important Takeaways:
    `Customization vs. development`
    `Concept of page template vs. page instance`
- Demo: **WssCreatePageObjectModel**
  - Create a customized page through the object model

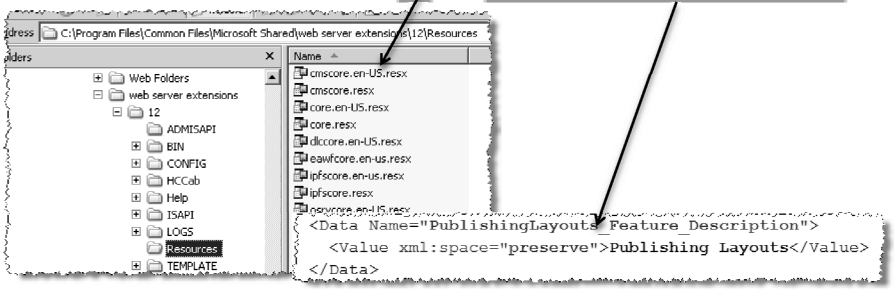
Two Types of Pages

- Content pages:
  - Web Part Pages
  - List pages (display / new / edit forms)
  - Can be personalized
  - Can use customized master pages
- Application pages:
  - Shared with all sites on a WFE server
  - Live within `[..]\12\TEMPLATE\LAYOUTS`
  - Can not be personalized
  - All share a single master page (`application.master`)

WSS Resource Files

- Provide an easy way for localizing a solution
- Special tokens used in XML files
- Ability to target a specific token file

```
<Feature  Id="D3F51BE2-38A8-4e44-BA84-940D35BE1566"
          Title="$Resources:cmscore,PublishingLayouts_Feature_Title;"
          Description="$Resources:cmscore,PublishingLayouts_Feature_Description;"
```



SharePoint Features

- Features are the building blocks of WSS 3.0 sites
- Low-level, reusable vehicle to define and deploy elements within a site
- Two primary uses:
  - Define and create site elements
  - Deployment and implementation of new functionality
- Can be activated on existing sites!
- Added flexibility that site definitions can't provide

What Can Features Do?

- Deploy site columns & content types
- Define & deploy list templates
- Create list instances from templates
- Define & deploy page templates
- Create page instances from templates
- Deploy Web Parts
- Deploy custom workflow templates
- Deploy event receivers to lists templates
- Create new menu items in the various menus within SharePoint sites

Advantages of Features

- Promote functionality & code reuse across sites
  - No longer have to copy the same list template definition / instances across multiple site definitions
- Features can be applied to any site within a SharePoint farm
- Features can be added / removed to sites via activation / deactivation
- Features can activate other dependent Features

Advantages of Features (Part 2)

- Used to deploy custom code solutions
  - Event receivers
  - Create custom permission levels
  - Workflows
- Features can be “stapled” to site definitions, thereby adding functionality to new sites
- Features are extensible!
  - Developers can write custom code to handle Feature events

Feature Installation & Scope

- All Features live in one place:
  - `[..]\12\TEMPLATE\FEATURES`
- Must be installed to make SharePoint aware
  - Installation only possible via STSADM.exe or the API
- Once installed, available for activation at it’s defined scope
- Feature scope options:
  - Farm
  - Web Application
  - Site Collection
  - Site

Feature Activation / Deactivation

- Activation options
  - Via pages linked from each site's "Site Settings" page
- Features can be hidden
  - Requiring farm administrators for activation
- Activation dependent upon the scope
- Feature activation dependencies
- Feature stapling – attaching to site definitions
- Feature deactivation
  - Removes functionality in Feature
  - Data created by Feature activation is not removed upon deactivation

WSS Solution Packages

- Deployment vehicle for custom code & files
- WSS solution packages are CAB files with a WSP extension
- Replaces WSS 2.0's Web Part packages
- Possible contents:
  - Site definitions
  - Feature definitions
  - Images & script files for site definitions, Web Parts, etc.
  - Assemblies (deployed to a site's \bin or the GAC)
  - Custom code access security policies
  - InfoPath forms for Forms Services
- `manifest.xml` file used to tell SharePoint where files go

Creating WSS Solution Packages

- Create solutions using MakeCab.exe (included in the Microsoft Cabinet SDK – Q310618)
- `MakeCab.exe` requires a Diamond Directive File (`*.ddf`) as an input parameter
  - Defines what files to include in the package, where they are on the source system, and where to put them in the target package (including subfolders)

Working with Solutions

- Add solution to farm's Solution Store:

```
stsadm.exe –o addsolution –filename c:\foo.wsp
```

- Deploy solution:

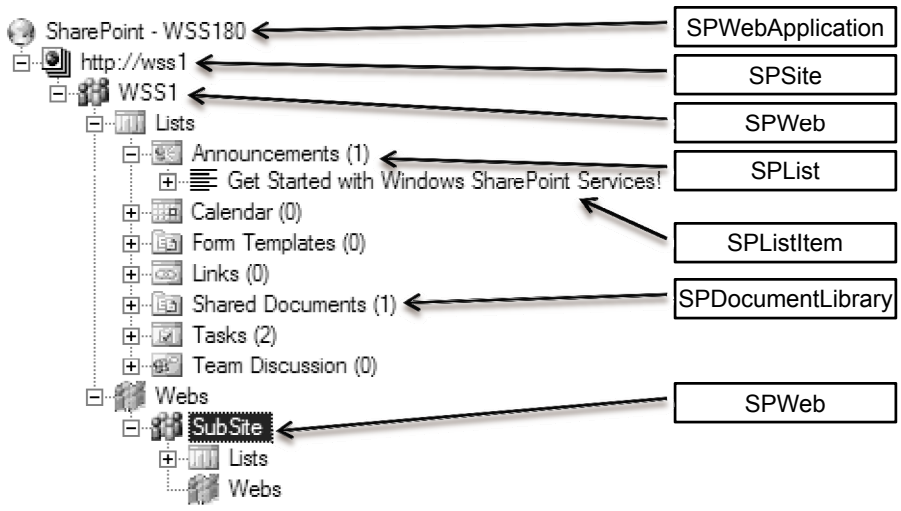
```
stsadm.exe –o deploysolution –name foo.wsp
–url http://bar -local
```

- Upgrading solutions:
  - Re-adds solution to farm's solution store and if previously deployed, redeploys

```
stsadm.exe –o upgradesolution –name foo.wsp
-filename c:\foo.wsp –url http://bar -local
```

- Retracting solutions:
  - "Undeploys" a deployed solution

Introducing Microsoft.SharePoint.dll



DEMO: Utilizing Microsoft.SharePoint

- Demo: **WssObjectModel**
  - Displays information about a specified WSS 3.0 site
  - Displays list of all lists within the specified WSS 3.0 site

Additional Commonly Used Objects

- Event receivers
  - Most contain pre & post events
  - Pre events are synchronous and cancellable
  - Post events are asynchronous and not cancellable
  - Examples:

    `SPItemEventReceiver`
    `SPListEventReceiver`
    `SPWebEventReceiver`
    `SPFeatureReceiver`
- `SPContext` (like ASP.NET 2.0's `HttpContext`)

```
SPSite siteCollection = SPContext.Current.Site;
SPWeb site = SPContext.Current.Web;
```

Development Environment Tips

- Out-of-the-box SharePoint sites ship with ASP.NET 2.0 custom errors turned on
- Sometimes the SharePoint error message doesn't provide enough information

Development Environment Tips (Part 2)

- Configure SharePoint to show the ASP.NET 2.0 "YSOD" (yellow screen of death)

- Configure SharePoint to show the call stack

- Where to do this?
  - `web.config` within the Web application's Web root
  - `web.config` within the `_layouts` virtual directory

Debugging SharePoint Projects

- No "F5" debugging
- Need to manually attach to the Application Pool hosting the SharePoint site (`w3wp.exe` process)
  - Visual Studio: Debug -> Attach to Process…
- Assemblies deployed to the GAC:
  - Need to deploy assembly symbol files to the GAC
  - Start -> RUN: `%systemroot%\Assembly\GAC`
  - Add *.PDB files to:
    `[AssemblyFile] \ [AssemblyVersion]__[AssemblyPublicKeyToken]`
  - Now manually attach to `w3wp.exe` process
- Use `%windir%\system32\iisapp.vbs` to get Application Pool process ID

Summary

- Introduction to Windows SharePoint Services 3.0
- Overview of WSS architecture
- Overview of WSS collaboration capabilities
- SharePoint customization vs. development
  - Customized / unghosted vs. uncustomized / ghosted
  - Templates vs. instances
- Overview of SharePoint Features
- Overview of WSS solution packages
- Working with the `Microsoft.SharePoint` Namespace

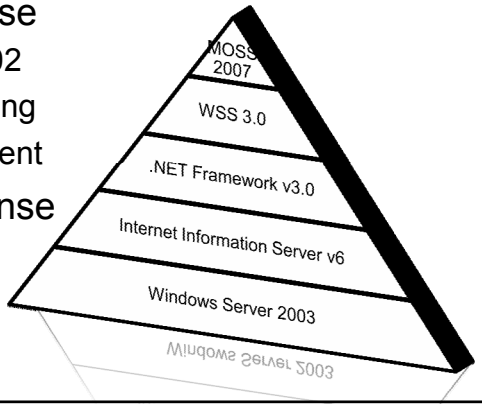
Office SharePoint Server 2007 & Web Content Management Overview

Agenda

- Overview of Office SharePoint Server 2007 components
  - Shared Service Providers
  - Forms Services
  - Enterprise Content Management
- Overview of Web Content Management
- Web Content Management terminology
- Working with the `Microsoft.SharePoint.Publishing` namespace

Overview of MOSS 2007 Components

- Office SharePoint Server 2007
  - Built on top of WSS 3.0
  - Separate license from WSS 3.0
- MOSS Standard License
  - SPS 2003 & MCMS 2002
  - Search, Social Networking
  - Web Content Management
- MOSS Enterprise License
  - Business Data Catalog
  - Excel Services
  - Forms Services



MOSS Social Networking & Search

- Similar search capabilities as SharePoint Portal Server 2003
  - My Sites
  - Profiles
  - Audience targeting
- Added customization & configuration capabilities
  - Search Web Parts
  - Indexing of Business Data Catalog applications

MOSS Business Data Catalog

- Introduced in MOSS 2007
- Adds capability to make MOSS aware of LOB systems
  - Like a connection point to LOB DB's or Web services
- Content is not imported into MOSS
- BDC application connections can be…
  - Indexed by MOSS search
  - Displayed using provided Web Parts
  - Used as lookup data within SharePoint lists
  - Consumed by custom Web Parts

MOSS Excel Services

- Adds Business intelligence capabilities for MOSS
- Tight integration with SQL Server 2005's and Excel 2007's business intelligence capabilities
  - Create KPI's from SQL Server Analysis Services cubes
  - Design complex spreadsheets with Excel 2007 & upload for greater reach via the browser
- Business intelligence features
  - Excel Services
  - Report Center
  - Dashboards

MOSS Forms Services

- Biggest complaint of InfoPath 2003 – client install
- Introducing Forms Services
  - Capability to host forms designed with InfoPath 2007 in the browser (cross browser: IE, Firefox, Safari, Opera)
- Smart rendering
  - If the client has InfoPath 2007 installed, form loaded in the client
  - Otherwise, form rendered in the browser (can be forced)
- Forms can be hosted in Windows forms and traditional ASP.NET 2.0 applications

Enterprise Content Management

- Supporting the full lifecycle of content

Integrated solution to manage the complete content lifecycle

- Records Management
  - Records Repository
  - Hold actions
- Document Management
  - Policies
  - Check in / check out
  - Workflow
- Web Content Management

MOSS Web Content Management

- MOSS WCM features bring capability to author, host, and maintain content-centric sites in MOSS
- Addresses challenges with content-centric sites
  - Maintain complex interactions B/T content authors
  - Enforcement of business rules
  - Promotes content reuse (not duplication)
  - Enforcing a common branding & user experience
  - Simplify content discovery
- Terminology:
  - WCM = publishing technologies & capabilities
  - Publishing sites = SharePoint site that leverages WCM

Site Structure

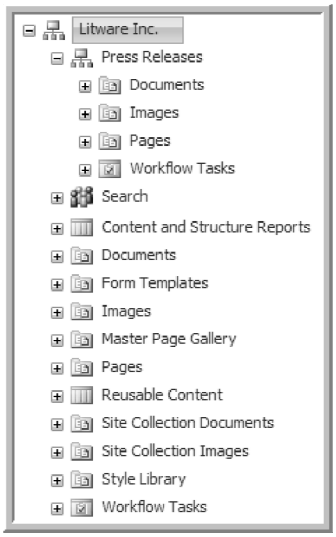
- Publishing sites are a hierarchy of Windows SharePoint Server (WSS) “Sites”
  - Hierarchy defines URL & navigation structure
- Each site has a library for Pages
  - Pages are rendered by Page Layouts
  - Pages can be branded using Master Pages

Architecture of a Publishing Site

- Site Collection – complete Publishing site
  - http://wcm.litwareinc.com
- Sites – sections within a site
  - http://wcm.litwareinc.com/Divisions
- Each site contains a Pages library, where all pages reside
  - http://wcm.litwareinc.com/Divisions/Pages/default.aspx

Architecture of a Publishing Site

- Each site can have unique permissions
- Each site has content galleries
  - Documents, Images & Pages
- Root site has special galleries
  - Master Page Gallery
  - Reports
  - Reusable Content
  - Site Collection Documents & Images



ABC's of Web Content Management

- Authoring – empowering content owners
  - Web-based authoring experience
  - Word / InfoPath authoring experience
- Branding – enforcing consistent user experience
  - Master pages & page layouts
- Controlled Publishing – enforcing rules & policies
  - Controlling who can author content & where
  - Controlling who can approve & publishing

Authoring Content

- Various methods to enter / edit content
  - Web-based authoring experience
  - Word / InfoPath authoring experience via Document Converters (extensible!!!)
- Content types & page layouts
- Content reports
- Reusable content

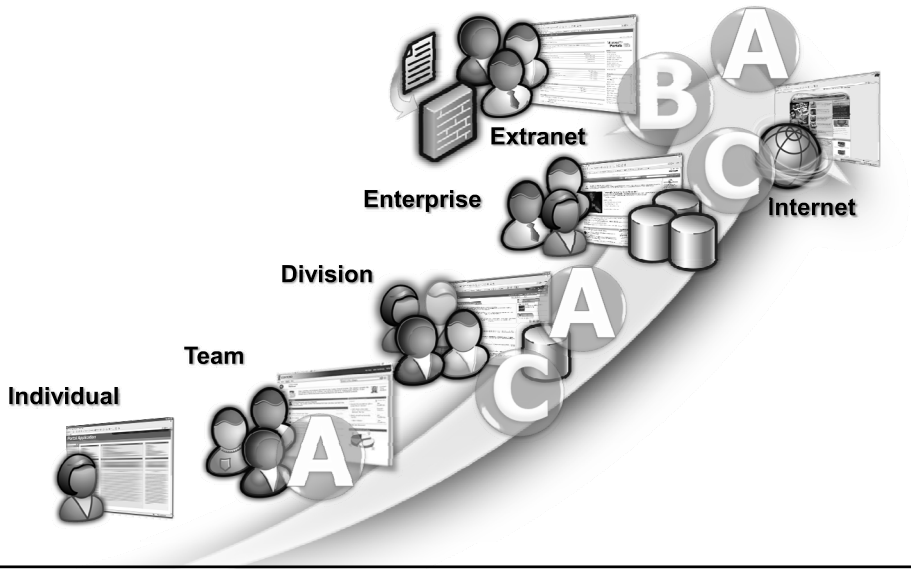
Branding: Implementing Consistent UX

- Common look & feel
  - Master pages, images & CSS
- Create content page templates
  - Site columns & content types
  - Page layouts
- Editable content regions
  - Field controls & Web Parts

Controlled Publishing

- Granular permissions
- History & versioning
- Page scheduling
- Out-of-the-box workflow
- Custom workflow
- Content deployment

Single Infrastructure for Intranet, Internet, and Extranet Portals



Page Editing Toolbar

- Contains most common page authoring tasks
  - Creating / editing pages
  - Checking in / out pages
  - Starting & interacting with workflows

- Three main sections:
  - Page Status Bar
  - Page Editing Menu
  - Quick Access Buttons

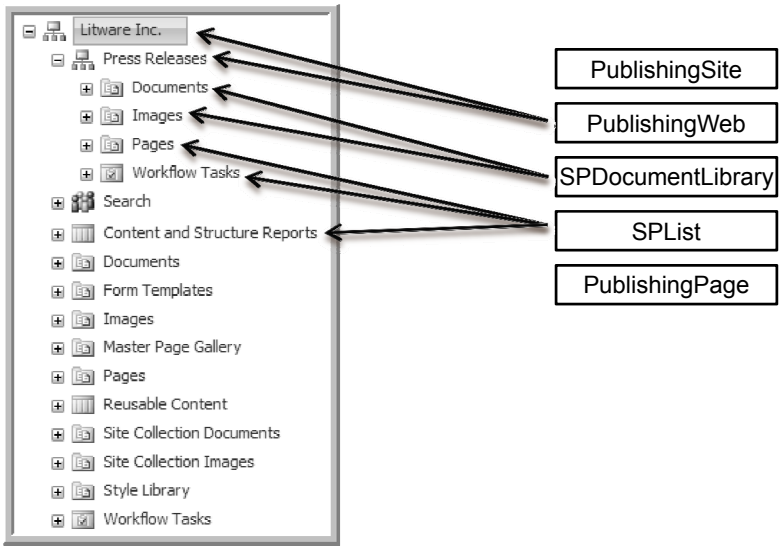
Content & Site Owner Tools

- Manage Content & Structure Page
  - Site & content owner accessible only
  - Overhead view of entire site topology
- Reports
  - Saved CAML queries
- Reusable Content
  - Content fragments for inclusion in pages
  - Choice of inserting a copy or an auto-updating reference

DEMO: Creating A Publishing Site

- Demo: Creating a Publishing Site
  - Creating standard Publishing site and taking a look around

Introducing Microsoft.SharePoint.Publishing.dll



DEMO: MSFT.SharePoint.Publishing.dll

- Demo: **PublishingObjectModel**
  - Displays information about a specified Publishing site
  - Displays all Page Layouts within a specified Publishing site
  - Displays all pages within a specified Publishing site

Summary

- Overview of Office SharePoint Server 2007 components
  - Shared Service Providers
  - Forms Services
  - Enterprise Content Management
- Overview of Web Content Management
- Web Content Management terminology
- Working with the `Microsoft.SharePoint.Publishing` namespace

Authentication & Authorization

SharePoint Security Fundamentals

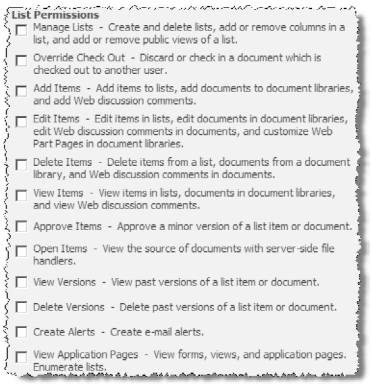
Agenda

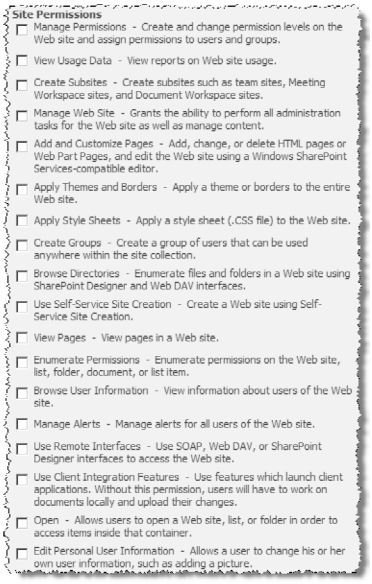
- Windows SharePoint Services security groups
- Web Content Management security groups
- Permission management & inheritance
- Creating custom permission levels
- WSS security architecture
  - Pluggable authentication
  - Alternate access mappings

Overview of Security & Permissions

- Securable objects can be assigned permissions
  - Site collections
  - Sites
  - Lists & libraries
  - List items & documents
- Use permission levels to group permission types
- Assign permission levels to SharePoint groups
- Use SharePoint groups to grant people rights
- Assign users and groups to SharePoint groups

WSS 3.0 Permissions





WSS 3.0 Permission Levels

- Limited Access:
  - Can view specific lists & libraries, list items, and folders when given permissions
  - Can't be removed
- Read:
  - Read only access… no edit rights
- Contribute:
  - Can view, add, update & delete items in lists & libraries
- Design:
  - Contribute rights + approve & customize
- Full Control:
  - Unrestricted access

WSS 3.0 SharePoint Groups

- Visitors
  - Granted permission level Read
- Members
  - Granted permission level Contribute
- Owners
  - Granted permission level Full Control

WCM Permission Levels

- *Same permission levels as WSS sites plus the following…*
- Approve:
  - Can edit & approve pages, list items & documents submitted for approval
- Manage Hierarchy:
  - Can create & manage sites, pages, list items and documents
- Restricted Read:
  - Can read pages & documents, but not previous versions or rights information

WCM SharePoint Groups

- *Same SharePoint groups as WSS sites plus the following…*
- Approvers
  - Granted Approve & Limited Access permission levels
- Designers
  - Granted Design & Limited Access permission levels
- Hierarchy Managers
  - Granted Limited Access & Manage Hierarchy permission levels
- Quick Deploy Users
  - Granted Limited Access permission level
- Restricted Readers
  - Granted Limited Access & Restricted Read permission levels
- Style Resource Readers
  - Granted Read permission to Master Page Gallery & Restricted Read to Style Library
  - All authenticated users are members of this group

Permission Mgmt & Inheritance

- By default, all SharePoint securable objects inherit permissions of their parent object
- Administrators can:
  - Break inheritance & assign unique permissions to any securable object
  - Re-establish permission inheritances

Customizing Groups & Permission Levels

- SharePoint Groups & Permission Levels can be created & customized via the browser UX or API
- Important object model classes:
  - `SPUser` = SharePoint user
  - `SPGroup` = SharePoint group
  - `SPRoleDefinition` = Permission level
  - `SPRoleAssignment` = used to associate a user / group with a permission level (in conjunction with the following class…)
  - `SPRoleDefinitionBindingCollection` = similar to a join table… used to associate users / groups with permission levels

Adding Permission Levels To Groups

```
using (SPSite siteCollection = new SPSite("http://wcm.litwareinc.com")) {
  using (SPWeb site = siteCollection.RootWeb) {
    // get reference to the site gorup
    SPGroup group = site.Groups["Litware Inc. Visitors"];

    // create new assignment for the group
    SPRoleAssignment roleAssignment = new SPRoleAssignment(group);

    // get reference to Designers permission level
    SPRoleDefinition roleDefinition = site.RoleDefinitions["Designers"];

    // add Designers permission level to site group
    roleAssignment.RoleDefinitionBindings.Add(roleDefinition);

    // add the role assignment to the site assignments collection
    site.RoleAssignments.Add(roleAssignment);

    // update the site
    site.Update();

    // visitors now granted Designer permission level
  }
}
```

DEMO: Working with Permissions via API

- Demo: **CustomPermissionLevelFeature**
  - Create a custom permission level with a Feature
  - Custom permission levels can be used as application roles

The "Lockdown Feature"

- By default, all users in a SharePoint site can access the SharePoint-y pages
  - http://wcm.litwareinc.com/pages/forms/allitems.aspx
- Not desirable in Publishing sites
- Controlled by the "View Application Pages" right granted by "Limited Access"
  - `SPBasePermissions.ViewFormPages`
- Feature ViewFormPagesLockdown removes this right from Limited Access
  - Automatically activated by Publishing Portal template

Policy for Web Applications

- Allows farm administrators to specify which users have global rights to all sites in site collections within a Web application
- Options:
  - Full Control
  - Full Read
  - Deny Read
  - Deny Write
- Overrules any permissions set at a lower level
- Example use: company facing litigation can deny an employee write access to a site collection

Pluggable Authentication

- ASP.NET 2.0's membership provider model available to all SharePoint sites!
- Not restricted to Active Directory
- Forms Based Authentication (FBA)
  - Including custom authentication providers
- Anonymous Access

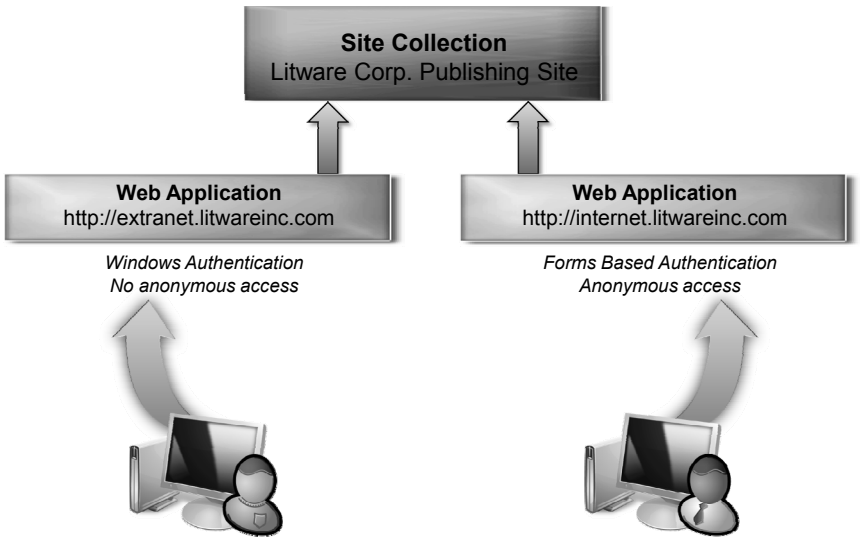
Alternate Access Mappings

- Create multiple paths into a site collection
  - Up to 5 unique paths (one for each zone)
- Possible uses:
  - Limit external access
  - Leverage existing internal authentication
- Example:
  - http://internet.litwareinc.com

    `Anonymous Access + FBA against custom user store`
  - http://extranet.litwareinc.com

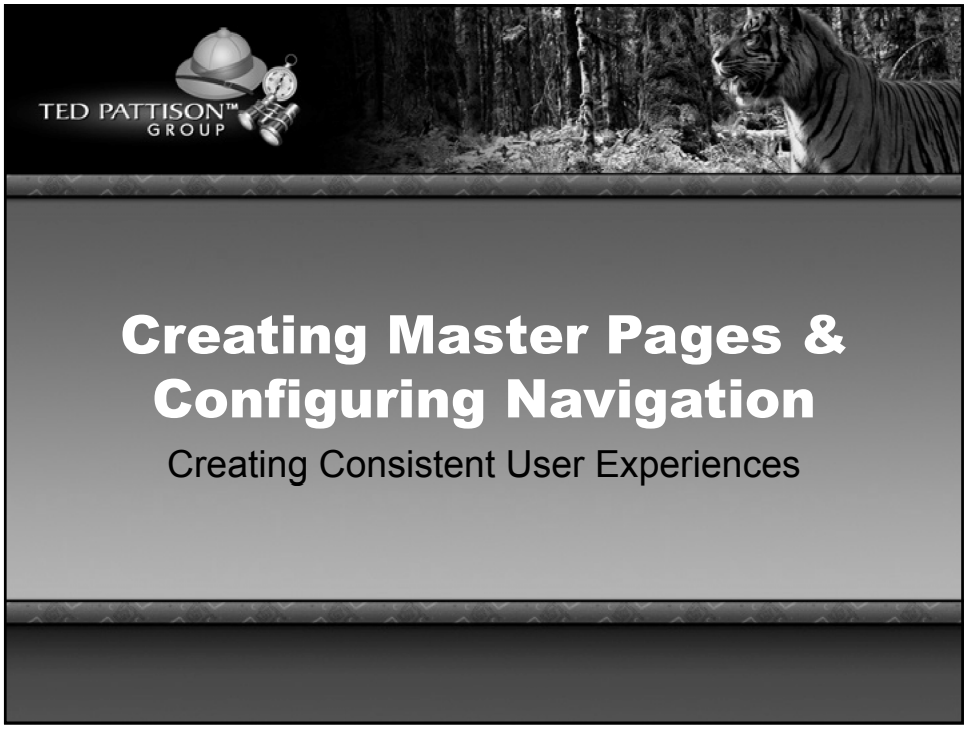
    `Non-anonymous access + AD against corp AD`

Alternate Access Mappings (Part 2)



Summary

- Windows SharePoint Services security groups
- Web Content Management security groups
- Permission management & inheritance
- Creating custom permission levels
- WSS security architecture
  - Pluggable authentication
  - Alternate access mappings



Creating Master Pages & Configuring Navigation

Creating Consistent User Experiences

Agenda

- Master pages in Windows SharePoint Services 3.0
  - Page templates vs. page instances
- Creating, editing & managing master pages
  - Office SharePoint Designer 2007
  - Visual Studio (Features & solutions)
- WSS 3.0 & MOSS 2007 navigation
- Managing & customizing navigation in Publishing sites

Master Pages in WSS 3.0

- Recall two previous concepts:
  - Page templates vs. instances
  - Content pages vs. application pages
- Master pages live within the Master Page Gallery
  - http://www.litwareinc.com/_catalogs/masterpage
  - Exists *only* in the top-level site within a site collection
- Upon site provisioning, `default.master` added to Master Page Gallery as `GhostableInLibrary`
  - While living in the Master Page Gallery as uncustomized, the `default.master` on the file system is used

Master Pages in Publishing Sites

- Publishing Portal site template provisions 8 additional master pages, specific to publishing sites
- Provisioned using the `PublishingLayouts` Feature

BlueBand.master	BlueGlassBand.master
BlackSingleLevel.master	BlueTabs.master
BlackVertical.master	BlueVertical.master
BlackBand.master	OrangeSingleLevel.master

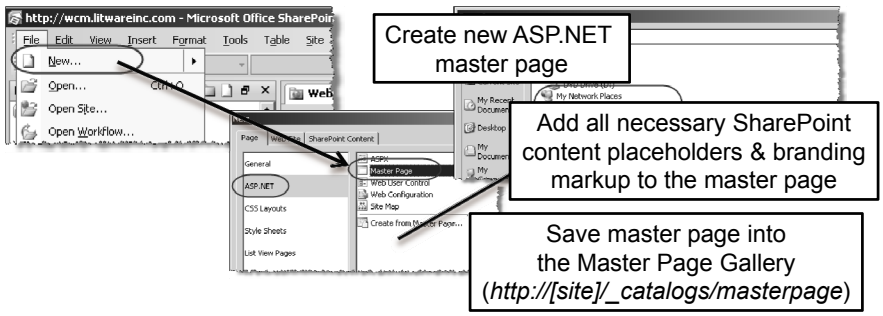
Creating & Editing Master Pages

- Create & edit using:
  - Office SharePoint Designer 2007
  - Any text editor
- All master pages in SharePoint must contain a minimum number of content placeholders
  - Can hide unused ones with `<asp:panel visible="false" runat="server">`
- Add master pages to Master Page Gallery using:
  - Office SharePoint Designer 2007

    `Initially added as customized`
  - Features using `<module>` elements

    `Initially added as uncustomized instances based off file templates`

Creating Master Pages with SPD



- In order for others to see and use it:
  - Check in
  - Publish a major version
  - Approve Publishing request

Creating Master Pages with Features

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- Add the master page to the Master Page Gallery -->
  <Module Url="_catalogs/masterpage"
          Path="MasterPages"
          RootWebOnly="TRUE">
    <!-- provision the minimal master page into the Master Page Gallery -->
    <File Url="Minimal.master"
          Name="MinimalMasterFeature.master"
          Type="GhostableInLibrary">
      <Property Name="ContentType"
                Value="$Resources:cmscore,contenttype_masterpage_name;" />
      <Property Name="PublishingPreviewImage"
                Value="~SiteCollection/_catalogs/masterpage/Preview Images/Litware/MinimalPreviewFeature.gif,
~SiteCollection/_catalogs/masterpage/Preview Images/Litware/MinimalPreviewFeature.gif" />
      <Property Name="Title"
                Value="MinimalMasterFeature.master" />
      <Property Name="Description"
                Value="Minimal master page provisioned into the Master Page Gallery via the
MinimalMasterFeature." />
    </File>
  </Module>

  <!-- Add the preview image to the Master Page Gallery -->
  <Module Url="_catalogs/masterpage/Preview Images/Litware"
          Path="MasterPages"
          RootWebOnly="TRUE">...
</Elements>
```

DEMO: Minimal Master Pages

- Demo: **Minimal.master**
  - Creating a “minimal master”, or a master page with no branding using Office SharePoint Designer 2007
- Demo: **MinimalMasterFeature**
  - Create & provision a minimal master page template & using a Feature
  - Create a Feature that adds the minimal master to all newly created sites via “Feature stapling”

Two Types of Master Pages

- `MasterUrl`
  - In Publishing sites, used in all "SharePoint-y pages"
  - List pages: new / edit / display
  - On the Site Settings / Master Page configuration page, this is the System Master Page (the 2nd one)
- `CustomMasterUrl`
  - In Publishing sites, used by all Page Layouts
  - On the Site Settings / Master Page configuration page, this is the Site Master Page (the 1st one)

Special Master Page Tokens

- SharePoint uses tokens with the `MasterUrl` & `CustomMasterUrl` attribute in a page declaration to dynamically inject the appropriate master page URL
- Dynamic tokens (whole string is the token):
  - **~masterurl/default.master** – replaced at runtime with the value in `SPWeb.MasterUrl` property
  - **~masterurl/custom.master** – replaced at runtime with the value in `SPWeb.CustomMasterUrl` property
- Static tokens:
  - **~sitecollection/[x].master** & **~site/[x].master** – replaced at runtime with the site-relative or site collection-relative master page

SharePoint Delegate Controls

- Areas carved out on master pages or site pages, designated for dynamic content injection
- Content registered for a specific delegate control done using Features
- Content can be any type of control (including server control)
- Sequences specified at registration time dictate priority over other registered controls
- Developers can add delegate controls… no special registration necessary

SharePoint Navigation Basics

- SharePoint leverages ASP.NET 2.0's navigation provider model
  - Navigation data source
  - Navigation [rendering] control
- ASP.NET 2.0 navigation provider model separates navigation data from rendering
- Enables easy modification of rendering
- SharePoint includes modified ASP.NET 2.0 menu navigation control: `<sharepoint:aspmenu>`
- Like permissions, can optionally inherit from parent

Publishing Site Navigation

- Simple navigation modifications via Site Settings
  - Change sorting (manual & automatic)
  - Add headings & pages
  - Include / exclude subsites & pages
- Additional customization via properties on core navigation components
- Three core components:
  - Navigation control: `<SharePoint:AspMenu>`
  - Data source properties: `PortalSiteMapDataSource`
  - Site map provider: `PortalSiteMapProvider`

DEMO: Examining Navigation Components

- Demo: Touring the primary navigation components within a Publishing site

SharePoint:AspMenu

- `DataSourceID`
  - ID of the data source providing nav hierarchy
- `Orientation`
  - Horizontal | Vertical
- `StaticDisplayLevels`
  - Number of hierarchy navigation levels to display
- `MaximumDynamicDisplayLevels`
  - Maximum number of hierarchy navigation levels to display
  - DHTML flyouts

PortalSiteMapDataSource

- `ShowStartingNode`
  - Include starting node in navigation data or not
- `StartFromCurrentNode`
  - When true, data source uses own logic to determine where to start (leave set to true)
- `TreatStartingNodeAsCurrent`
  - Which node is treated as the current node

PortalSiteMapDataSource (Part 2)

- `TrimNonCurrentTypes`
  - Remove nodes directly in current node
- `TrimNonAncestorTypes`
  - Remove nodes directly above or below current node
- `TrimNonAncestorDescentent`
  - Remove nodes above or below current node
- *Possible Types:*
  - *Heading*
  - *Page*
  - *AuthoredLink*

PortalSiteMapProvider

- `NavigationType`
  - **Global**: includes links from TopNavigationBar collection
  - **Current**: includes links from QuickLaunch collection
  - **Combined**: acts like Global when site not set to inherit; acts like Current when site set to inherit
- `EncodeOutput`
  - HTML encodes the navigation node title
- `DynamicChildLimit`
  - Number of navigation nodes to include for flyouts

PortalSiteMapProvider (Part 2)

- `RequireUniqueKeysForNodes`
- `IncludeSubSites` & `IncludePages`
  - **Always**: ignore `SPWeb`'s settings, always include these types of navigation items
  - **PerWeb**: respects each `SPWeb`'s settings
  - **Never**: ignore `SPWeb`'s settings, never include these types of navigation items
- `IncludeHeadings` & `IncludeAuthoredLinks`
  - Boolean flags to include / not include types of links

DEMO: Customizing the OOTB Nav.

- Demo: Customizing the out-of-the-box navigation components
  - Navigation control (`<SharePoint:AspMenu>`)
  - Site map data source

Customizing Navigation via API

- What if `PortalSiteMapProvider` is not suitable?
  - Do not recreate a brand new site map provider
  - Subclass SharePoint `PortalSiteMapProvider`

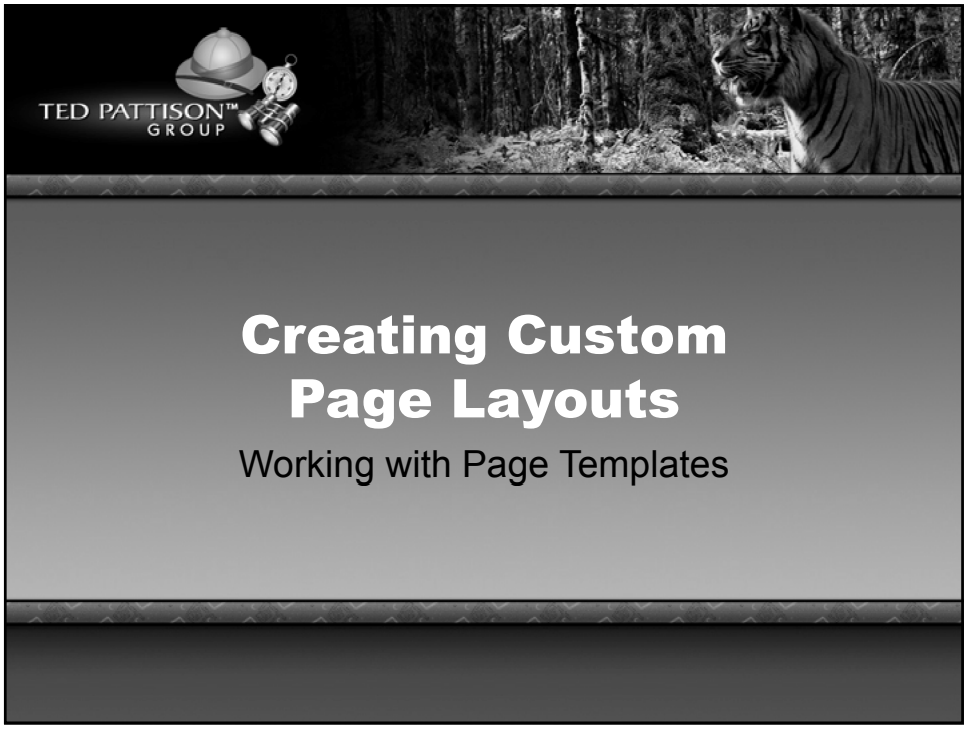
```
using System;
using System.Web;
using Microsoft.SharePoint.Publishing;
using Microsoft.SharePoint.Publishing.Navigation;

namespace WCM401 {
  /// <summary>
  /// Reguardless of settings, never includes contents of a list in nav.
  /// </summary>
  public class WcmPortalSiteMapProvider : PortalSiteMapProvider {
    public override SiteMapNodeCollection GetChildNodes (SiteMapNode node) {
      PortalSiteMapNode smNode = node as PortalSiteMapNode;

      // if nothing defined or if the node is a list, return nothing
      if (smNode == null || smNode.Type == NodeTypes.List)
        return new SiteMapNodeCollection();
      else // else, act normally
        base.GetChildNodes(smNode);
    }
  }
}
```

Customizing Navigation via API (Part 2)

- Manipulate navigation via SharePoint API
  - `SPWeb.Navigation.QuickLaunchBar` – provides a reference to the site's Quick Launch navigation
  - `SPWeb.Navigation.TopNavigationNode` – provides a reference to the site's Global Navigation (tabbed, horizontal navigation)
  - `SPNavigationNode` & `SPNavigationNodeCollection` – used to create navigation items (menu items)

Summary

- Master pages in Windows SharePoint Services 3.0
  - Page templates vs. page instances
- Creating, editing & managing master pages
  - Office SharePoint Designer 2007
  - Visual Studio (Features & solutions)
- WSS 3.0 & MOSS 2007 navigation
- Managing & customizing navigation in Publishing sites



Agenda

- Role of site columns & content types in Publishing sites
- Creating, editing & managing page layouts
  - Office SharePoint Designer 2007
  - Visual Studio (Features & solutions)
- Field controls & Web Parts
- Customizing Rich HTML Editor control

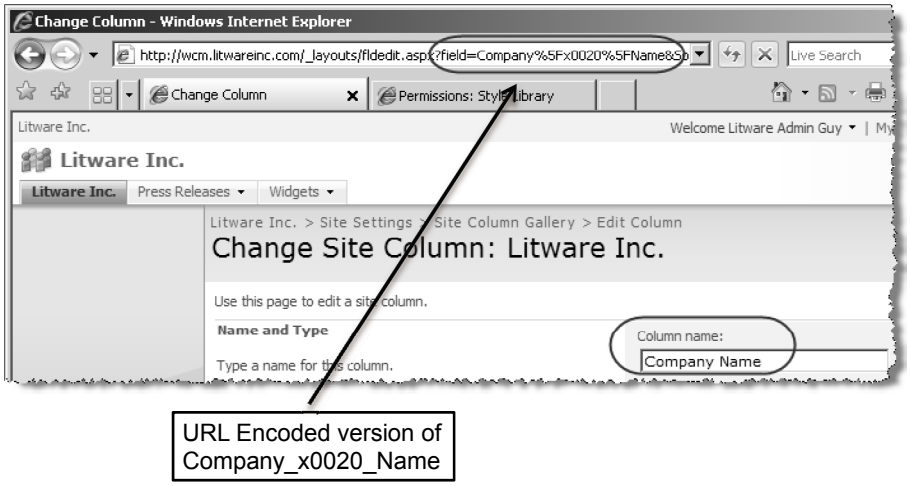
Site Columns

- Reusable column template for use within lists and content types
- Specify:
  - Name & description
  - Data type
  - Site column group
- Scoped at the site level
  - Available to child sites
- Adding to a list / content type creates a copy
- To manage, must have Web designer rights

Creating & Managing Site Columns

- Create / manage many ways:
  - Browser-based user interface
  - SharePoint object model
  - Feature XML `<Field>`
- Features & object model provide most control
  - Easy to version in virtually all source control systems
  - Highest level of reuse
- Avoiding dreaded **company_x0020_name** in CAML

The Feared & Dreaded _x0020_ !!!



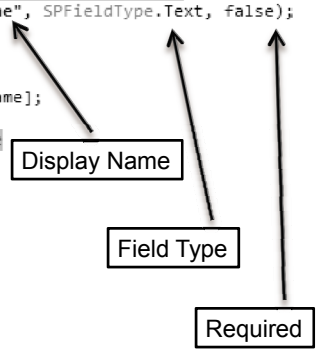
Creating Site Columns with Features

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- new site column based off the "single line of text" field type -->
  <Field SourceID="http://schemas.microsoft.com/sharepoint/v3"
         ID="{0D499375-D734-43DB-A2EE-343B490B9CB0}"
         Name="PRByLine"
         DisplayName="Press Release ByLine"
         Group="WCM401"
         Type="Text"
         Required="FALSE"
         Sealed="FALSE"
         Hidden="FALSE" />
</Elements>
```

Creating Site Columns via API

- Need a workaround to avoid the _x0020_ issue
  - Create field using the internal name & immediately update display name

```
using (SPSite siteCollection = new SPSite("http://wcm.litwareinc.com")) {
  using (SPWeb site = siteCollection.RootWeb) {
    // create field but use the desired internal name as the display name
    string prByLineFieldName = site.Fields.Add("PRByLine", SPFieldType.Text, false);
    site.Update();

    // get reference to the field just created
    SPField prByLineField = site.Fields[prByLineFieldName];

    // now set the display name to something desireable
    prByLineField.Title = "Press Release ByLine";

    // set other properties
    prByLineField.Group = "WCM401";
    prByLineField.Sealed = false;
    prByLineField.Hidden = false;
    site.Update();
  }
}
```



Content Types

- Used to define a type of data
- Enable storage of different types of content in same list or library
- Specify:
  - Name & description
  - Site columns, workflows, event receivers, policies, etc.
  - Content type group
- Scoped at the site level
  - Available to child sites
- Adding to a list creates a copy
- To manage, must have Web designer rights

Creating & Managing Content Types

- Create / manage many ways:
  - Browser-based user interface
  - SharePoint object model
  - Feature XML `<ContentType>`
- Features & object model provide most control
  - Easy to version in virtually all systems
  - Highest level of reuse

Creating Content Types with Features

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ContentType ID="[content type id]"
               Name="Press Release"
               Description="Litware Inc Press Release"
               Group="WCM401">
    <FieldRefs>
      <FieldRef ID="{0D499375-D734-43DB-A2EE-343B490B9CB0}" Name="PRByLine" />
      <FieldRef ID="{0C50E0BF-A35C-4A38-B784-AFA60BBADC00}" Name="PRBody" />
    </FieldRefs>
    <DocumentTemplate TargetName="/_layouts/CreatePage.aspx" />
  </ContentType>
</Elements>
```

Creating Content Types via API

```
using (SPSite siteCollection = new SPSite("http://wcm.litwareinc.com")) {
  using (SPWeb site = siteCollection.RootWeb) {
    // assume "Page" is the base content type
    SPContentType baseContentType = site.AvailableContentTypes["Page"];

    // create the content type and set a few properties
    SPContentType pressReleaseContentType =
      new SPContentType(baseContentType, site.ContentTypes, "Press Release");
    pressReleaseContentType.Description = "Litware Inc Press Release";
    pressReleaseContentType.Group = "WCM401";
    pressReleaseContentType.DocumentTemplate = "/_layouts/CreatePage.aspx";

    // add fields to the content type
    SPField prByLinefield = site.AvailableFields["Press Release ByLine"];
    pressReleaseContentType.FieldLinks.Add(new SPFieldLink(prByLinefield));
    SPField pBodyField = site.AvailableFields["Press Release Body"];
    pressReleaseContentType.FieldLinks.Add(new SPFieldLink(pBodyField));

    // add content type to the site and update both
    site.ContentTypes.Add(pressReleaseContentType);
    pressReleaseContentType.Update();
    site.Update();
  }
}
```

Site Columns & Content Types In Publishing Sites

- Site columns & content types play a key role in Publishing sites
- Both are used to define schema for content pages
- Content types must inherit from the **Page** content type (which inherits from **System Page**)
  - [..]\FEATURES\PublishingResources\PublishingContentTypes.xml
  - 0x010100C568DB52D9D0A14D9B2FDCC96666E9F2007948130EC3DB064584E219954237AF39
- Page layouts must be associated with content types

Overview of Page Layouts

- Combined with master page to define the rendering of a content page
  - Obtains master page from `SPWeb.CustomMasterUrl`
- Hosts field controls and Web Part zones
- Complies to the Page Layout content type
- Has an associated content type
  - Dictates which content type the page layout is used for rendering
- Can bind multiple page layouts to content type
- Associated with exactly one content type

Page Layouts in Publishing Sites

- Recall previous concept:
  - Page templates vs. instances
- Page layouts live within the Master Page Gallery
  - http://www.Litwareinc.com/_catalogs/masterpage
  - Exists *only* in the top-level site within a site collection
- Upon site provisioning, page layout added to Master Page Gallery as `GhostableInLibrary`
  - While living in the Master Page Gallery as uncustomized, the page layout on the file system is used

Components of Page Layouts

- Field Controls (*Module 8*)
  - Field controls are bound fields within pages list item
  - Used to statically place content on a page
  - Content owners can NOT move or modify field controls
  - Developers can create custom field controls
- Web Parts (*Module 7*)
  - Designers & developers place Web Part zones on page layouts
  - Allow content owners more control over content
- Edit Mode Panel (*Module 6*)
  - Used to show / hide renderings in display / edit mode

Field Controls & Web Parts Compared

	Field Controls	**Web Parts**
Content Storage	In a field (column) in the page's underlying `SPListItem`	Within the Web Part data of the page
Personalization	No	Yes
Versioning	Versioning tied to page with complete history	Versioning tied to page *without* history
Who has ultimate control?	Page designer / developer	Page designer (in placement of Web Part zones); content owner in managing of zone's contents (add/edit Web Parts within Web Part zone)
When to use?	Specific types of content must appear in specified places of a page; structured formatting / branding	Structure of content on page (in part of a page) isn't important; let content owners have full control

Customizing Rich HTML Editor Control

- Rich HTML editor control used by default with fields of type Publishing HTML
- Developers can specify if the following are / aren't allowed within the field:

HTML Tables	Formatting (bold, italic, etc.)
Hyperlinks	Access to raw markup
Images	Which CSS Classes Are Available
Font Family or Sizes	Reusable Content
Font Colors	

- Specify using attributes in markup (or with SharePoint Designer's tool window)
- Extensible – add custom buttons to floating toolbar
  - *Module 6*

Creating Page Layouts with SPD 2007

- In order for others to see and use it:
  - Check in
  - Publish a major version
  - Approve publishing request

Creating Page Layouts with Features

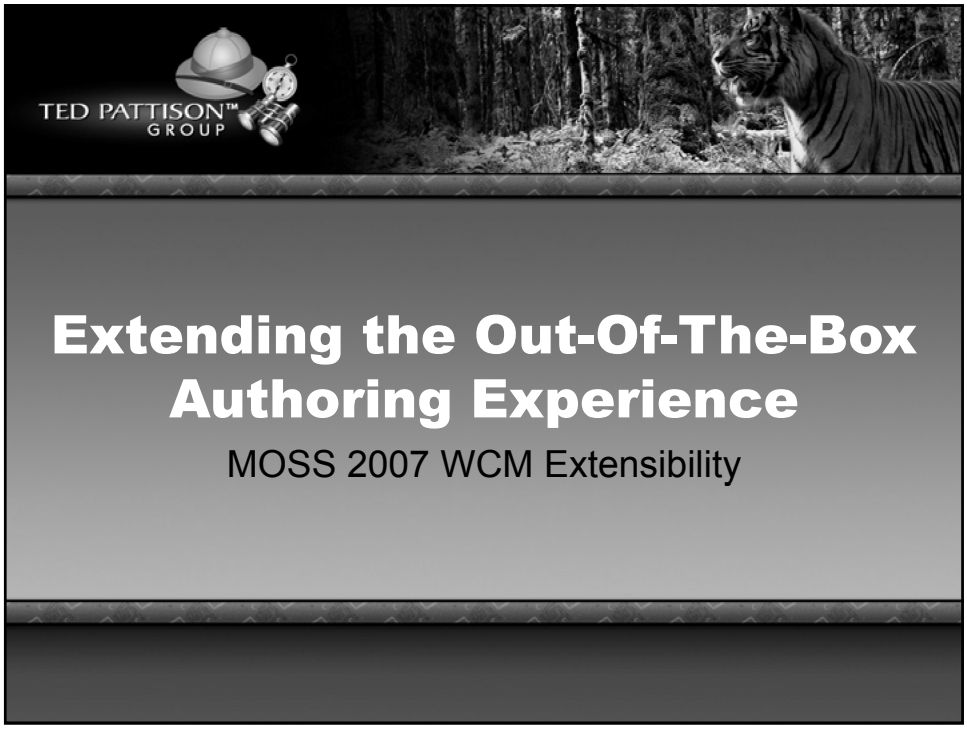
```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Url="_catalogs/masterpage"
          Path="PageLayouts"
          RootWebOnly="TRUE">
    <File Url="CustomPageLayout.aspx"
          Type="GhostableInLibrary">
      <Property Name="PublishingAssociatedContentType"
                Value=";#Press Release;#
0x010100C568DB52D9D0A14D9B2FDCC96666E9F2007948130EC3DB064584E219954237AF3900D626686F6C2C4C97AAA4BB1D8F2EDA5C;#
" />
      <Property Name="PublishingPreviewImage"
                Value="~SiteCollection/_catalogs/masterpage/Preview Images/Litware/CustomPageLayoutPreview.gif
, ~SiteCollection/_catalogs/masterpage/Preview Images/Litware/CustomPageLayoutPreview.gif" />
      <Property Name="ContentType"
                Value="$Resources:cmscore,contenttype_pagelayout_name;" />
      <Property Name="Title"
                Value="Litware Press Release" />
    </File>
  </Module>

  <!-- Add the preview image to the Master Page Gallery -->
  <Module Url="_catalogs/masterpage/Preview Images/Litware"
          Path="PageLayouts"
          RootWebOnly="TRUE">...
</Elements>
```

DEMO: Creating Page Layouts

- Demo: Creating content types & page layouts with the browser & SharePoint Designer interface
  - Create a new content type
  - Create a new page layout using SharePoint Designer
  - Add field controls, Web Part zones & Web Parts
  - Create new content page based on new page layout
- Demo: **PageLayoutFeature**
  - Create new site columns, content type and page layout using a Feature instead of SharePoint Designer
- Demo:
  - Modify which page layouts can be used within a subsite

Summary

- Role of site columns & content types in Publishing sites
- Creating, editing & managing page layouts
  - Office SharePoint Designer 2007
  - Visual Studio (Features & solutions)
- Field controls & Web Parts
- Customizing Rich HTML Editor control



Extending the Out-Of-The-Box Authoring Experience

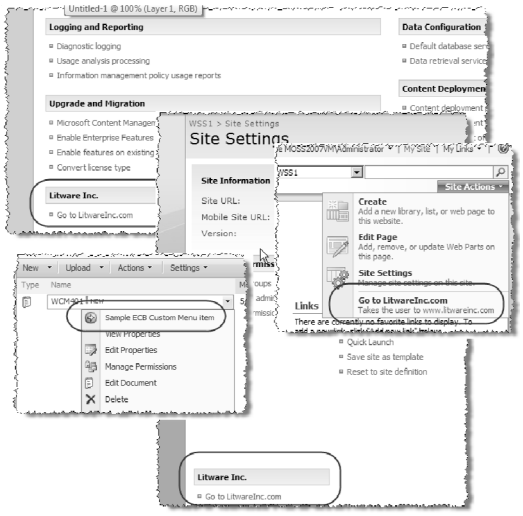
MOSS 2007 WCM Extensibility

Agenda

- Custom actions (augmenting SharePoint menus)
- Extensibility opportunities with Publishing sites
- Leveraging the Edit Mode Panel
- Extending the Rich HTML Editor control
- Implementing Telerik's RadEditor Lite
- Extending the Page Editing Toolbar
- Implementing an offline authoring experience
  - Document converters

Augmenting SharePoint Menus

- SharePoint menus are easily managed using custom actions in Features
- `<CustomActionsGroup>`
- `<HideCustomAction>`
- `<CustomAction>`



Custom Actions

- Areas affected by custom actions:
  - Central Administration
  - Site Settings
  - Site Actions
  - Edit Control Block (ECB)
  - List menus (New | Upload | Actions)
  - List form toolbars (New | Edit | View)
- Custom Actions Can:
  - Point to a URL, class or ASCX file
  - Be associated with specific list types
  - Security trimmed with rights assignments

Overview of Extensibility Options

- Edit Mode Panel
  - Useful to show certain elements in page layout files in different presentation modes (display | edit)
- Rich HTML Editor field control
  - Add new buttons to the authoring control
- Page Editing Toolbar
  - Can add / remove menu items from the Page Editing menu or Quick Access buttons area
- Offline authoring capabilities
  - Author pages using Word, InfoPath or another tool
  - Upload to document library & pages generated automatically

Leveraging the Edit Mode Panel

- Edit Mode Panel:
  - Control that contains other controls
  - Renders contents in diff. states (display | edit) via `PageDisplayMode`
- Possible uses:
  - Article rollup image / keywords / metadata for the page

Rich HTML Editor Field Control

- OOTB provides built-in buttons to add & manage tables, links, formatting, images, etc.

- Buttons can be enabled / disabled via markup
- Custom buttons can be added
- Custom styles can be specified, forcing content owners to use approved formatting (no free-form bold / italics / underline / font)

DEMO: Customizing the HTML Editor

- DEMO: Customizing the Rich HTML Editor field control
  - Disabling certain buttons
  - Designating specific CSS classes for use
  - Adding custom buttons to the field control

Telerik's RadEditor Lite

- OOTB MOSS Rich HTML editor not cross browser capable (targeted for Internet Explorer)
- Telerik's RadEditor Lite
  - *Note: this is not the commercial full featured, RadEditor*
  - Fully cross browser
  - Adds multiple undo levels
- Customers with a MOSS license get a RadEditor Lite for MOSS license
- Get it from: http://www.telerik.com/sharepoint

DEMO: Implementing Telerik RadEditor

- DEMO: Implementing the Telerik RadEditor
  - Installation of the free Telerik RadEditor control
  - Incorporating RadEditor as the HTML editor within page layouts

Extending the Page Editing Toolbar

- Capability to add / remove items from the Page Editing Menu and / or Quick Access Buttons

- How to customize:
  - OOTB Editing Menu driven from XML files in `..\12\TEMPLATE\LAYOUTS\EditingMenu`
  - Problem is these are global to all sites on the server
  - For site-by-site customizations, edit similar XML files in the site's Master Page Gallery \ Editing Menu using SharePoint Designer

Customizing the Page Editing Menu

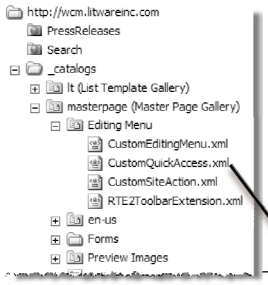
- Deploy assembly containing class that inherits:
  - `Microsoft.SharePoint.Publishing .WebControls.EditingMenuActions .ConsoleAction`
- Add a reference to assembly containing new command
  - Similar to a ASPX / ASCX `Register` directive
- Manipulate new / existing nodes within the `<structure>` section

Customizing the Page Editing Menu

Customizing Quick Access Buttons

- Deploy assembly containing class that inherits:
  - `Microsoft.SharePoint.Publishing.WebControls.EditingMenuActions.ConsoleAction`
- Add a reference to assembly containing new command
  - Similar to a ASPX / ASCX `Register` directive
- Manipulate new / existing nodes within the `<structure>` section

Customizing the Quick Access Buttons




```
<Console>
  <references>
    <reference TagPrefix="wcm401"
               assembly="[[ 4-part assembly name ]]"
               namespace="PublishingPageDetailViewer" />
  </references>
  <structure>
    <ConsoleNode Sequence="1"
                 ConfigMenu="Add"
                 HideStates="PageHasCustomizableZonesFalse|PageHasFieldControlsFalse"
                 Action="wcm401:PublishingPageDetailMenuItem"
                 ID="wcm401PubPageDetailQuickAccess" />
  </structure>
</Console>
```

DEMO: Page Editing Toolbar

- DEMO: **PublishingPageDetailViewer**
  - Adding a new menu action to the Page Editing menu
  - Adding a new button to the Quick Access Buttons

Offline Authoring Capabilities

- Content authors can author pages using Word / InfoPath
  - Once the document has been created, upload to a document library
  - A registered document library picks up document & runs it through a document converter
  - Document converter parses document, creating a content page in the configured /Pages library
  - SharePoint creates an association between the document and content page for future updates
- Document Converters included OOTB:
  - Word, InfoPath or XML

Note: Microsoft calls this "smart client authoring"

Setting up Document Converters

- Configure two services on the SharePoint server:
  - Document Converter Launcher Service
  - Document Converter Load Balancer Service
- Service setup & configuration:
  - Central Admin -> Operations -> Services on server
- Configure document conversions on a per Web application
  - Central Admin -> App. Management

Note: Document converters can not run on Windows Active Directory Domain Controllers

Creating & Deploying Document Converters

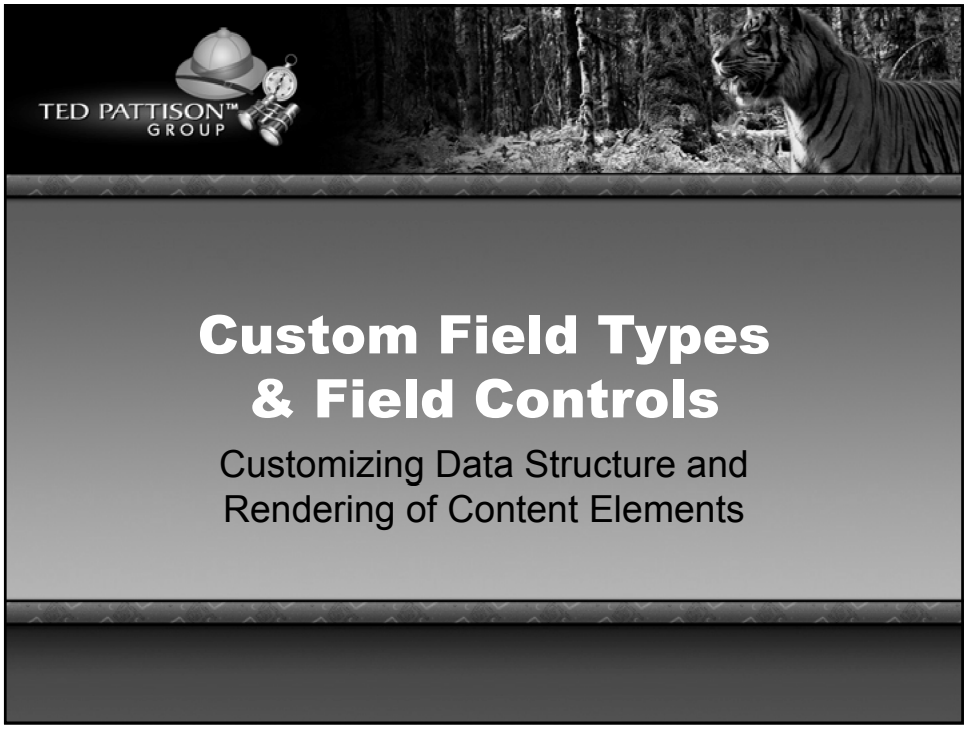
- Create a console application that does the conversion work
  - Must accept following input parameters:

```
-in inFile
-out outFile.html
[-config configuration.xml]
[-log logfile.xml]
```

- Register custom converter with SharePoint via a Feature
  - Use the `<DocumentConverter>` node in a Feature elements manifest
  - Must be scoped at the Web application level
- Optionally create a custom ASPX for converter-specific settings

Summary

- Extensibility opportunities with Publishing sites
- Custom actions (augmenting SharePoint menus)
- Leveraging the Edit Mode Panel
- Extending the Rich HTML Editor control
- Implementing Telerik's RadEditor Lite
- Extending the Page Editing Toolbar
- Implementing an offline authoring experience
  - Document converters



Agenda

- Understanding the relationship between field types and field controls
  - All those moving parts!!!
- Custom field types
  - Creating & deploying custom field types
- Custom field controls
  - Creating & deploying custom field controls
- Incorporating custom field controls into page layouts

Starting With An Analogy…

- "What's old is new again…"
- Think of custom field types in the same context as SQL Server 2005
- Sometimes the provided field types don't meet business needs

SharePoint	SQL Server
OOTB Field Types	OOTB Data Types
Custom Field Type	Custom CLR Data Type
List Template / Content Type Schema	Table Schema (DDL / TSQL)
List Instance	Table Instance

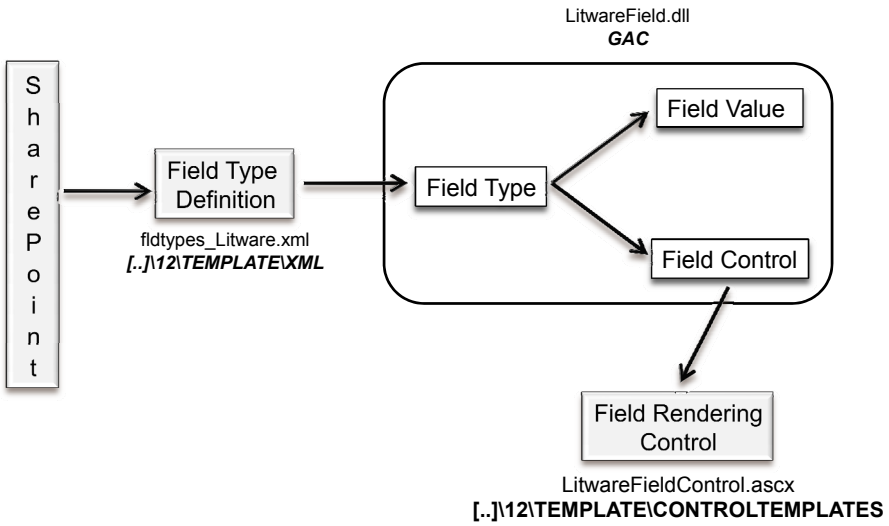
Reviewing the Problem

- At times the included SharePoint field types do not meet business requirements for:
  - Storage of data elements
  - Pull data from external systems & not rely on users to type the customer ID (preferred: select from a list of IDs)
  - Performing validation on data entry / edits

Custom Field Types

- SharePoint provides a way to create custom field types
- Allows developers to define:
  - Structure of the stored data
  - Rendering of the field in display / new / edit mode (as well as alternate rendering based on the context, such as a mobile device)
  - Custom validation
  - Custom default values for new items

Field Type & Control Components



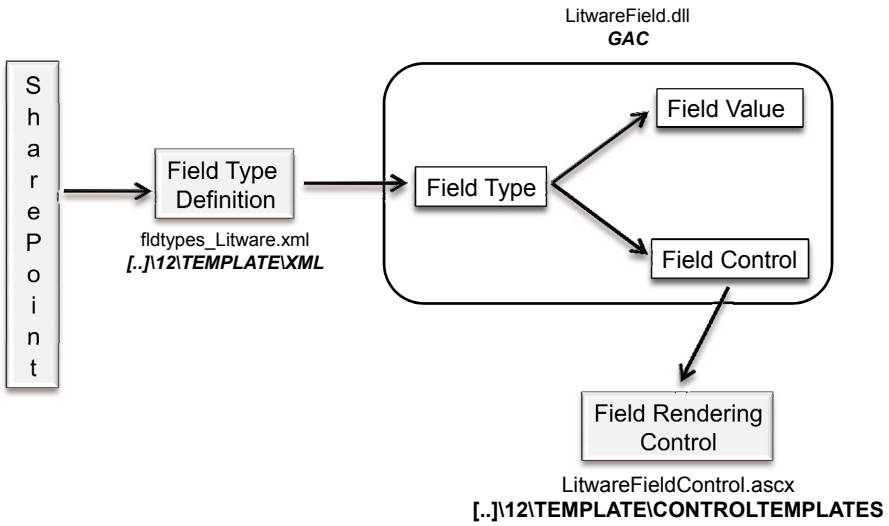
Creating Custom Field Types Requirements

- Create a custom field type class inheriting from a base field type class:

SPFieldBoolean	SPFieldMultiChoice	SPFieldText
SPFieldChoice	SPFieldMultiLineText	SPFieldUrl
SPFieldCurrency	SPFieldNumber	SPFieldUser
SPFieldDateTime	SPFieldRatingScale	SPFieldMultiColumn
SPFieldLookup		

  - Must be deployed to GAC as a strong-named assembly
- Create a field type definition:
  - XML file making SharePoint aware of the field type
  - Contains pointer to class & assembly containing type
  - Contains rendering logic & default values

Field Type & Control Components



Custom Field Values Type

- When base field types do not support a special data structures required by your custom field type, you can create your own
- Requirements:
  - Must be serializable – should implement `ToString()`
  - Must implement two constructors (can simply call the base value type class):

```
public class FieldResourceIconValue : SPFieldMultiColumnValue {
  private const int NUM_OF_FIELDS = 4;

  public FieldResourceIconValue () 2
      : base(NUM_OF_FIELDS) { }

  public FieldResourceIconValue (string value) 2
      : base(value) { }

  properties
}
```

Custom Field Values Type (Part 2)

- Custom field type values require overriding one method on the field type class:
  - `GetFieldValue()`

    Called when SharePoint makes a call to SPListItem["CustomFieldType"]
- Optionally, add server-side validation:
  - `GetValidatedString()`

    Converts the type into a serialized and validated string.

    More on this in a moment…

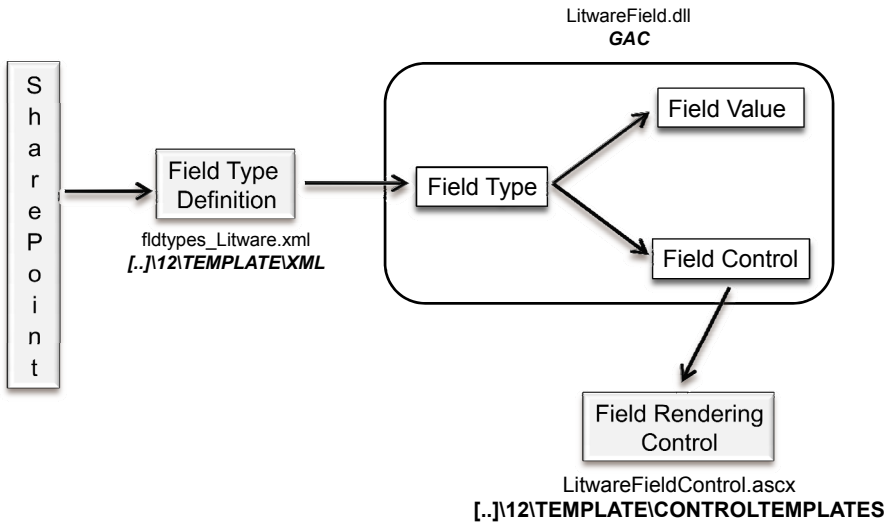
Customizing Behavior of Custom Field Types

- `GetFieldValueAsText()` & `GetFieldValueAsHtml()`
  - Returns the field as either text / HTML
- `GetFieldValue()`
  - Use this when the type contains a custom value type
- `IDesignTimeHtmlProvider.GetDesignTimeHtml()`
  - Used to display the field type in design mode (SPD)
- `OnAdded` / `OnUpdated` / `OnDeleting`
  - Events enabling developers to add custom logic on conditions

Validating Custom Field Types

- Validation can be defined in rendering control using ASP.NET 2.0 validation controls
  - Downside: bypassed when working with the field through code.
- Can override the `GetValidatedString()` method on the custom field class
  - If data is not valid, throw exception of type `SPFieldValidationException`
  - Make sure to include logic if the field is required (indicated by the `Required` property on the base field class)

Field Type & Control Components



Custom Field Controls

- Use custom field controls when the edit experience of a field does not meet business requirements
- Do not need to create a custom field type when creating a custom field control
  - Can leverage an existing field type
- Create an ASCX file for the new / edit experience
- Can include ASP.NET 2.0 validation controls for client-side validation

Creating Custom Field Controls

- Create class that inherits from:
  - `Microsoft.SharePoint.WebControls.BaseFieldControl`
- Override following methods & properties:
  - `DefaultTemplateName` – specifies the of the rendering template
  - `CreateChildControls()` – associate & init any web controls in rendering template
  - `Value` – get / set properties on the web controls
- Create associated ASCX file containing a `<SharePoint:RenderingTemplate>` tag

Creating Custom Field Control (Part 2)

- If custom field control is associated with a custom field type…
  - Associate the field control with a custom field type by overriding the `FieldRenderingControl` property on the field type class
- Deployment:
  - Copy ASCX rendering template to `[..]\12\TEMPLATE\CONTROLTEMPLATES`
  - Deploy assembly to site's `\bin` or WFE's GAC

Leveraging Custom Controls In Page Layouts

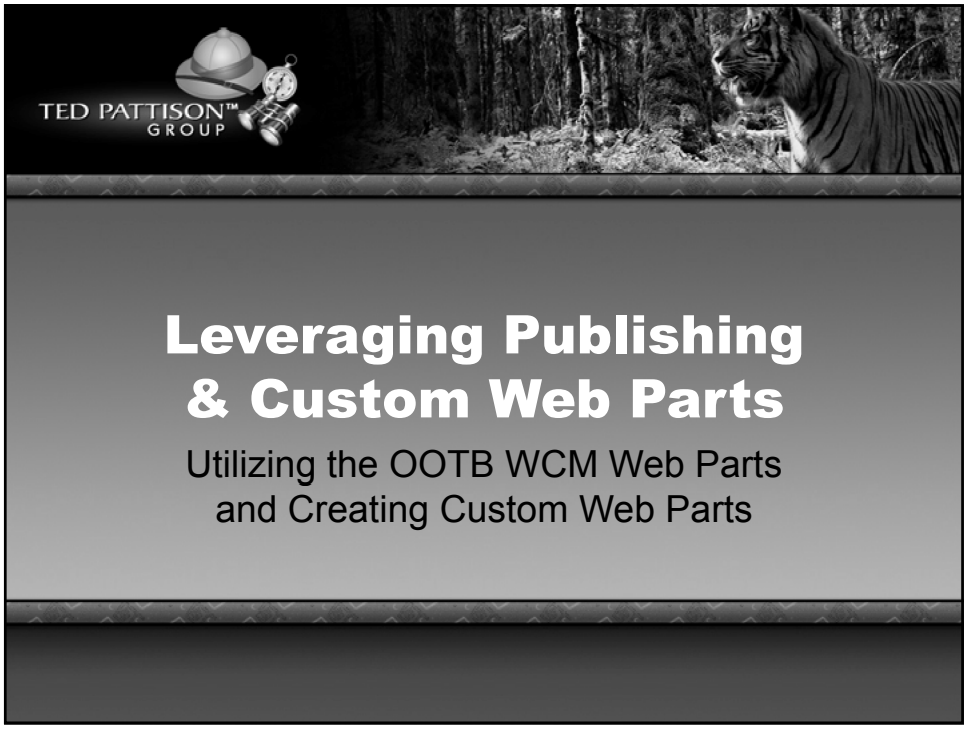
- Treat custom field controls as server controls
- Add `<%@ Register %>` directive at the top of the page layout
- Add field control server control tag
- Make sure to include the `FieldName` attribute
  - This links the field control to an underlying field type in the associated content type

Reviewing all the Moving Parts

- Field type class (ie: `SPFieldText`)
  - Contains definition of custom field type
- Field value class (ie: `SPFieldMultiColumnValue`)
  - Contains custom data structure serialization / deserialization
- Field control class (ie: `BaseFieldControl`)
  - Contains server-side logic for the rendering control
- Field control rendering template (ie: `Litware.ascx`)
  - ASCX file containing new / edit form
- Field type definition (ie: `fldtypes_Litware.xml`)
  - Contains definition and metadata of the field type

Summary

- Understanding the relationship between field types and field controls
  - All those moving parts!!!
- Custom field types
  - Creating & deploying custom field types
- Custom field controls
  - Creating & deploying custom field controls
- Incorporating custom field controls into page layouts



Agenda

- Overview of Web Parts
- Publishing Web Parts
  - Table Of Contents Web Part
  - Summary Links Web Part
  - Content Query Web Part
- Creating custom Web Parts
- Deployment of custom Web Parts

What is a Web Part?

- ASP.NET 2.0 server control
- Collection of controls to perform a specific function
- Enables user to personalize their experience
  - Behavior
  - Content
  - Appearance
- Two Web Parts can be connected to pass data between them
  - Example: List of Reports & Report Viewer

Possible Uses for Web Parts

- Small functional components
  - Display contents of a list
  - Display an image
- Part of a larger application
  - Entering sales orders
- Window into another application
  - Microsoft SQL Server 2005 Reporting Services

Publishing Web Parts

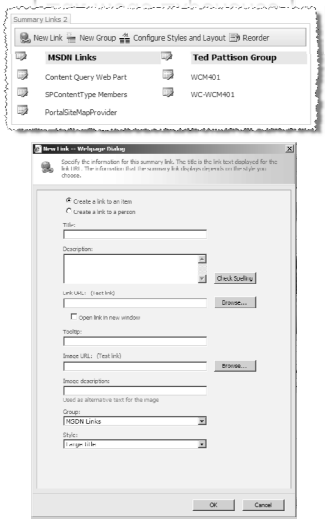
- Publishing sites have three additional Web Parts
  - Summary Links Web Part
  - Table of Contents Web Part
  - Content Query Web Part
- All retrieve their structure as XML
- All render presentation to HTML via XSLT
  - Developers can customize XSLT rendering
- Tip: Content Query Web Part will eliminate need for most custom Web Parts

Summary Links Web Part

- Allows content authors to manually create & manage headings and links
- Supports multiple sorting options
  - Automatic (ascending / descending)
  - Manual
- Also provided as a field control
  - Allows page designers / developers to restrict placement of Summary Links Web Part
- Similar to the Links list (and associated Web Part), except more styling options
- Ideal for providing extra links for a piece of content

Summary Links Web Part (Part 2)

Edit Mode Experience



Display Mode Experience

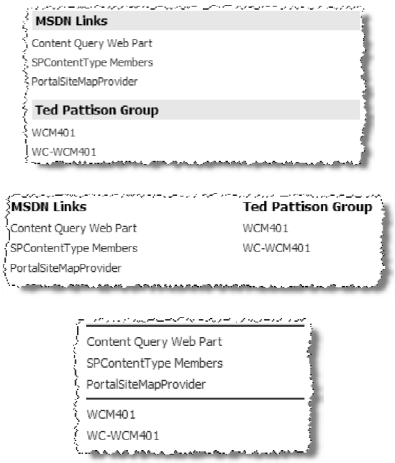
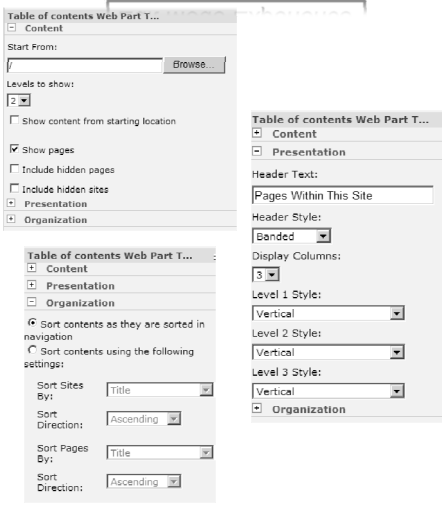


Table of Contents Web Part

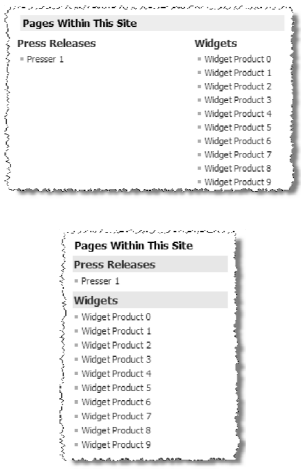
- Displays pages from the current site
  - Optionally can include pages from child sites
- Ideal for section welcome pages (homepage) showing
- Helps keep site primary navigation from becoming cluttered with content pages

Table of Contents Web Part (Part 2)

Edit Mode Experience



Display Mode Experience



Content Query Web Part

- Most powerful and flexible aggregation / rollup Web Part available in Publishing sites
- Allows content author to specify:
  - Scope of query
  - Filtering by specific content type / list type
  - Filtering by field values
  - Grouping & sorting
- Very performant with robust caching techniques leveraged within the Web Part

Advanced Customizing of the CQWP

- Content Query Web Part provides many configuration options via the Web Part Task Pane
- For advanced customization:
  - Export the CQWP to a `*.webpart` file
  - Modify CQWP's public properties directly in the `*.webpart` file
  - Import `*.webpart` file
- Use this technique to specify:
  - Specific CAML query to execute
  - Specify additional content fields to include in result set
  - Additional filtering capabilities

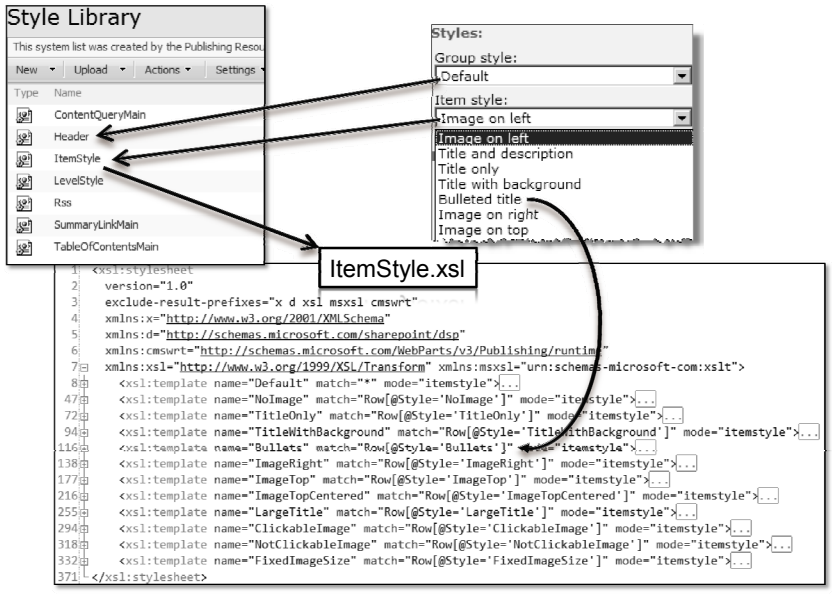
DEMO: Content Query Web Part

- Working with the Content Query Web Part's Task Pane
- Viewing all available properties in the CQWP by exporting it to a `*.webpart` file

Styling the Publishing Web Parts

- All Publishing Web Parts (Summary Links, Table of Contents, Content Query) return query results as XML, passing through XSL to generate HTML
- MOSS includes many OOTB style sheets for use
  - All found in the Style Library in the top-level web:
    `http://.../Style Library/XSL Style Sheets`
- Instead of customizing the provided XSL style sheets, create new ones
  - Benefit: can be packaged in a Feature for easy deployment

Styling Publishing Web Parts (Part 2)



DEMO: Customizing Rendering of the CQWP

- Create a new style sheet
- Define a new rendering style
- Configure Publishing Web Part to use new XSL style sheet

Custom Web Parts

- Why would you need to develop Web Parts?
  - Frequently the provided Publishing Web Parts do not meet unique business needs
  - Want to provide content owners more flexibility in the layout of a page
  - Provide a personalized experience
  - Provide a mini-application (current weather applet)
- Develop custom Web Parts as ASP.NET 2.0 Web Parts
  - WSS 3.0 API includes a Web Part class, but it is primarily there for backwards compatibility

Creating Custom Web Parts

- Build a typical ASP.NET 2.0 server control:
  - Create a new class that inherits from: `System.Web.UI.WebControls.WebParts.WebPart`
  - Override `CreateChildControls()`

    Used to add any child controls to the page such as buttons, textboxes, labels, etc.
  - Override `RenderContents()`

    Renders the contents of the Web Part, inside the outer tags and Web Part chrome
  - Never override `Render()` !!!!

    SharePoint overrides `Render()` to include the Web Part chrome and outer tags

Creating Custom Web Parts (Part 2)

- Few extra steps to get an ASP.NET 2.0 Web Part working in SharePoint:
  - Sign the assembly (need a strongly named assembly)
  - Tell ASP.NET 2.0 & SharePoint Web Part is OK to run

    `System.Security.AllowPartiallyTrustedCallers`

    `<SafeControl />`
  - Create Web Part definition file (`*.webpart`) with the type name (ie: five-part class name) and default public property settings

    `[webroot]\wpcatalog`

    `http://.../_catalogs/wp (Web Part Gallery)`

Real World Deployment Story

- Deploy using WSS solution packages
  - Microsoft Cabinet files with *.WSP extension
- Solution packages contain:
  - Assembly containing Web Part
  - Any resources (images, CSS, etc)
  - Web Part definition file (*.webpart)

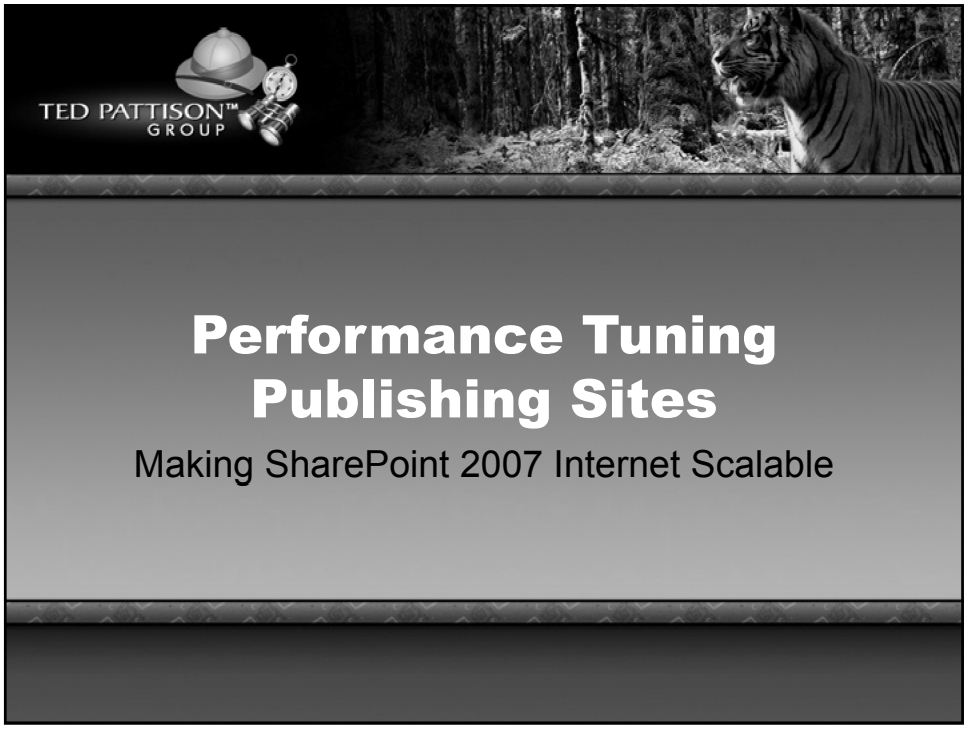
    `Web Part Gallery`
    `wpcatalog`
  - Manifest of the solution contents (manifest.xml)
- Add solution package to SharePoint farm's solution store & deploy (Central Admin / STSADM.EXE)

DEMO: Creating Custom Web Parts

- Demo: Creating a custom Web Part
- Demo: Deploying Web Parts
  - Automatically populate the Web Part Gallery
  - Add Web Part definition file to a single site collection
  - Add Web Part to all site collections in a web application

Summary

- Overview of Web Parts
- Publishing Web Parts
  - Table Of Contents Web Part
  - Summary Links Web Part
  - Content Query Web Part
- Creating custom Web Parts
- Deployment of custom Web Parts



Agenda

- Revisiting Windows SharePoint Services 3.0 architecture
  - What benefits ASP.NET 2.0 brings to the table
- Web Content Management Publishing performance enhancements
  - Page output cache via profiles & policies
  - Object caching
  - Disk-based caching
- IIS Static & Dynamic Compression
- Page Payload Concerns & Techniques
- Prescriptive guidance when coding against the SharePoint API
  - Working with SharePoint collections
  - Proper memory management techniques

Performance Benefits From ASP.NET 2.0

- ASP.NET 2.0's introduction of the `VirtualPathProvider`, adoption of Web Part framework, and others allow WSS 3.0 to be built on top of the .NET Framework 3.0, eliminating the ISAPI filter in WSS v2
- Page output caching heavily leveraged by MOSS 2007

Improving Performance with Caching

- Two biggest improvements WRT performance:
  - Minimize roundtrips to the database
  - Avoid initializing and executing the ASP.NET 2.0 page lifecycle
- Caching helps this in two ways:
  - Avoiding dynamic code execution
  - Retrieve data / files from memory: RAM / HDD (disk)

Page Output Cache

- Fastest type of caching
- After ASP.NET 2.0 page lifecycle generates the HTML markup it is saved in RAM
- Future requests bypass the ASP.NET 2.0 page lifecycle and instead used cached HTML
- Publishing sites provide Web interface to create cache “profiles”
- Profiles assigned to individual sites
- ***Note****: not enabled by default*

Object Cache

- Like output cache, provided by ASP.NET 2.0
- Commonly used objects are stored in RAM
  - Master pages, page layouts, etc.
- Cross-site queries (ie: CQWP queries) stored in RAM to eliminate future roundtrips to the site collection’s content database
  - ***Note****: not enabled by default*
- Amount of memory configurable (100MB = default)
- **Tip**: Use the ASP.NET 2.0 performance counter Cache Hit Ratio to determine effectiveness:
  - *Increase configured memory if not exceeding 90%*

Disk-Based Cache

- Commonly requested static files (images, CSS, JS, etc) stored in libraries in the site are pulled from the content database on each request
- Disk-based cache stores these files on the WFE server's disk to eliminate future round trips
- Can optionally configure the `max-age` HTTP header causing user's browsers to not request the files for a specified duration
- Configured via the Web app's `web.config`

DEMO: MOSS Caching Capabilities

- Page output caching
  - Creating and managing profiles
  - Managing policies
- Object caching
- Disk-based caching
  - Configuring WFE blob caching

Static & Dynamic Compression

- When SharePoint installed, IIS is configured to compress static & dynamic files
  - Static: HTM, HTML, TXT
  - Dynamic: ASP, EXE
- Example: core.js is 257 KB on disk, but IIS compresses to 54 KB before sent to client
- IIS compression level is configurable

IIS 6 vs. IIS 7 Compression

	IIS 6	IIS 7
Default Static Comp.	10	7
Default Dynamic Comp.	0	0
Compress by File Type	YES	NO
Compress by MIME Type	NO	YES
Compress content before adding to page output cache	NO	Configurable

- In IIS 7, compression (static & dynamic) is on / off & is enabled / disabled based on CPU utilization
- In IIS 7, markup can be compressed before inserting into page output cache
  - *Not enabled by default*

Windows 2003 / IIS 6 Compression

- **Tip**: Add .CSS & .JS to static file list:
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/DEFLATE/HcFileExtensions “htm” “html” “txt” “css” “js”
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/GZIP/HcFileExtensions “htm” “html” “txt” “css” “js”
- **Tip**: Add .AXD & .ASPX to dynamic file list
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/DEFLATE/HcScriptFileExtensions “asp” “exe” “axd” “aspx”
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/GZIP/HcScriptFileExtensions “asp” “exe” “axd” “aspx”
- **Tip**: Increase dynamic compression level to 9
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/DEFLATE/HcDynamicCompressionLevel “9”
  - CSCRIPT.EXE ADSUTIL.VBS SET W3Svc/Filters/Compression/GZIP/HcDynamicCompressionLevel “9”
  - ***Note**: higher compression level = higher CPU use*
- **Tip**: Don’t compress images (already compressed)

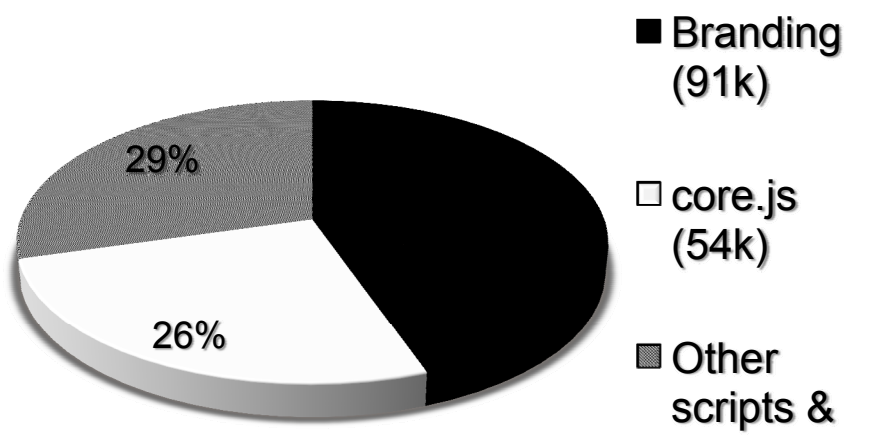
Windows 2008 / IIS 7

- **Tip**: Increase dynamic compression level to 9
  - <httpCompression>

```
<scheme dynamicCompressionLevel=“9” />
</httpCompression>
```

- **Tip**: Change dynamic compression CPU utilization threshold range from 20-75%
  - APPCMD.EXE set config –section:httpCompression /dynamicCompressionDisableCpuUsage:75
  - APPCMD.EXE set config –section:httpCompression /dynamicCompressionEnableCpuUsage:20
- **Tip**: Enable caching before insertion into page output cache
  - APPCMD.EXE set config –section:urlCompression /dynamicCompressionBeforeCache:true

Publishing Portal Page Payload



13

Limiting The Page Payload

- Page payload is the combined file size of a single page request
- OOTB, Publishing Portal homepage is 204k
  - Content + branding = 91k (44.6% of payload)
  - core.js accounts for 54k (26% of payload)

    `Can implement techniques to "lazy load" core.js or even fully suppress it!`
- Configure IIS dynamic compression
- Build accessible (CSS based) sites for up to a ~50% reduction in payload
- MSKB # 933823: Delay loading of core.js
  - http://www.andrewconnell.com/go/163

.NET Framework & IDisposable

- Objects implementing `IDisposable` interface signal developers should call `Dispose()` as early as possible
- Calling `Dispose()` typically releases unmanaged resources not seen by the .NET Garbage Collector

Properly Disposing SharePoint Objects

- `SPSite` & `SPWeb` are small managed code wrappers to much larger unmanaged code blocks
  - .NET Garbage Collector does not release these objects from memory in a timely manner because of the larger unmanaged portion
  - Tip: Don't rely on the .NET Garbage Collector, call `Dispose()` when finished using them
- Under heavy loads, this can cause:
  - High memory use & `OutOfMemory()` exceptions
  - Frequent WSS application pool recycles
  - Application crashes

Properly Disposing SharePoint Objects

- Tip: Utilize `try-catch-finally` statements, disposing within the `finally` block

```
SPSite siteCollections;
SPWeb topLevelWeb;
try
{
  siteCollections = new SPSite("http://wcm.litwareinc.com");
  topLevelWeb = siteCollections.RootWeb;
}
catch
{}
finally
{
  topLevelWeb.Dispose();
  siteCollections.RootWeb.Dispose();
  siteCollections.Dispose();
}
```

Properly Disposing SharePoint Objects

- Tip: Utilize C#'s `using()` statements
  - Automatically calls `Dispose()`

```
string siteTitle;
string siteDescription;
using (SPSite siteCollection = new SPSite("http://wcm.litwareinc.com"))
{
  using (SPWeb site = siteCollection.RootWeb)
  {
    siteTitle = site.Title;
    siteDescription = site.Description;
  }
}
```

Disposing Publishing Objects

- `PublishingSite`
  - No need to dispose or close objects
- `PublishingWeb`
  - Should call `Close()` as early as possible
  - Holds a reference to the associated `SPWeb` object
- `PublishingPage`
  - No need to dispose or close objects

Avoid Unnecessary Creation / Disposal of Objects

- Most SharePoint collections expose indexes
- Using indexes on each call degrades performance
  - SharePoint has to hydrate an internal copy of the object on each call
  - Results in numerous unnecessary:

```
Creation / disposal of objects
Roundtrips to the SharePoint content database
```

Properly Working With Collections

- Tip: Create local copy of the object

```
// BAD way
SPWeb site = SPContext.Current.Web;
string tasksTitle = site.Lists["Tasks"].Title;
string tasksDescription = site.Lists["Tasks"].Description;

// GOOD way
SPWeb site = SPContext.Current.Web;
SPList tasksList = site.Lists["Tasks"];
string tasksTitle = tasksList.Title;
string tasksDescription = tasksList.Description;
```

Aggregating Data in MOSS Sites

- Before you build a custom solution, evaluate the Content Query Web Part
  - Query builder (scope, filtering, grouping, rendering, etc)
  - Returns data as XML, uses XSLT to render as HTML
  - Sophisticated data caching mechanism for queries
- `PortalSiteMapProvider`
  - Primarily used for navigation controls
  - Optimized to query data not only across lists, but across webs as well
  - Scope limited to the current site collection
  - Not to be used for infrequent queries or when underlying data frequently changes as it will be counter productive

Using the PortalSiteMapProvider

```
// get reference to provider for current navigation
PortalSiteMapProvider psmp = PortalSiteMapProvider.CurrentNavSiteMapProvider;

// get specific node in the navigation
PortalWebSiteMapNode node = psmp.FindSiteMapNode("/Division1/PressReleases")
   as PortalWebSiteMapNode;

// get all pages created this fiscal year
SPQuery spQuery = new SPQuery();
spQuery.Query = "<<CAML_QUERY>>";
SiteMapNodeCollection pages =
  psmp.GetCachedListItemsQuery(node, "Pages", spQuery, SPContext.Current.Web);
```

SharePoint Capacity Considerations

- Performance starts to degrade as the number of items in a collection approaches 2,000 items
  - Not a hard limit, just a recommendation
  - Performance degradation primarily limited to browser based list views and upgrades
  - Applies to…

    `Sites: number of subsites within a site`

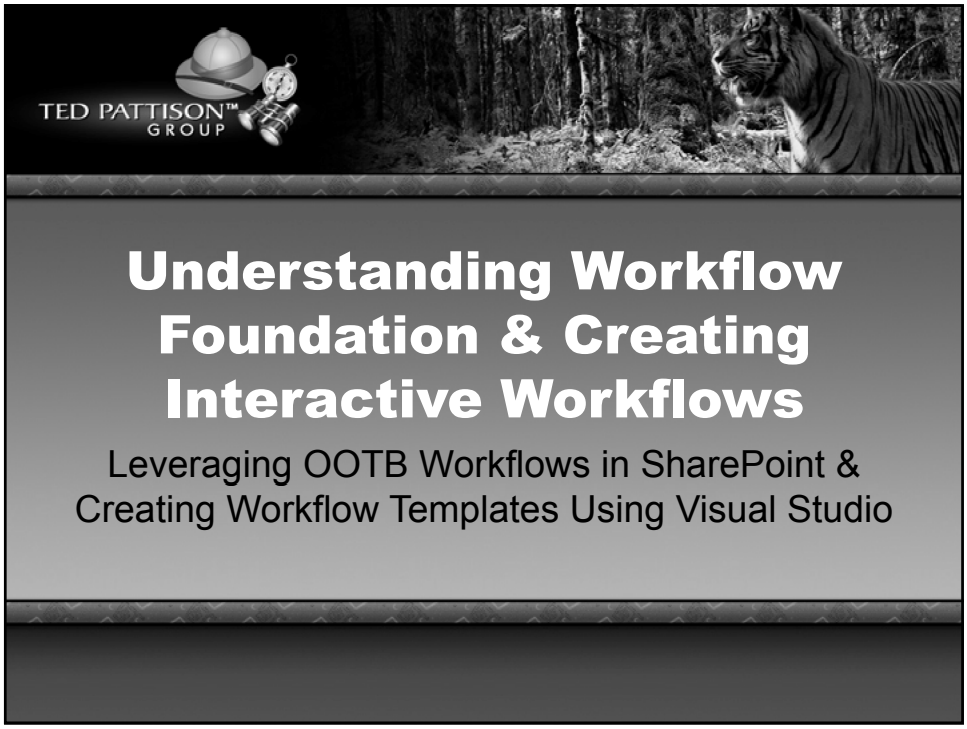
    `Lists: number of items / folders within one level of a list`
  - Mitigate the issue by grouping items within containers

    `Group similar sites within one site`

    `Group list items within folders`
- TechNet: Working with Large Lists in MOSS 2007
  - http://www.andrewconnell.com/go/160

Summary

- Revisiting Windows SharePoint Services 3.0 architecture
  - What benefits ASP.NET 2.0 brings to the table
- Web Content Management Publishing performance enhancements
  - Page output cache via profiles & policies
  - Object caching
  - Disk-based caching
- IIS Static & Dynamic Compression
- Page Payload Concerns & Techniques
- Prescriptive guidance when coding against the SharePoint API
  - Working with SharePoint collections
  - Proper memory management techniques



Agenda

- Windows Workflow Foundation core concepts
- What SharePoint brings to the Workflow Foundation
- Workflow Forms in SharePoint
- Out-of-the-box workflows in SharePoint
- Workflow creation & development in SharePoint
- Debugging workflows

Windows Workflow Foundation

- Windows Workflow Foundation:
  - Development platform for building reactive, episodic programs
  - Runtime components that ship with the .NET Framework 3.0
- Windows Workflow Foundation concepts:
  - Workflow runtime
  - Workflow instance
  - Activities

Workflow Runtime

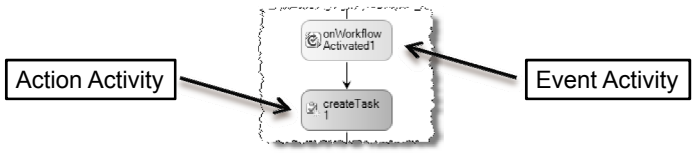
- Extends the capability of the CLR that provides ability to create objects in memory:
  - Adds persistence (serialization / deserialization) of long-running workflow instances
  - Adds scheduling to “wake up” sleeping (persisted) workflow instances
  - Adds concept of activities (like WinForm / ASP.NET UI controls)
- Hosts workflow instances
  - Executing workflow programs

Workflow Activities

- Activities are similar to Windows Forms / ASP.NET 2.0 controls:
  - Encapsulated logic that does a specific task
  - Not dependent upon other activities
  - Can be used in multiple workflows
- Composite activities:
  - Can contain child activities (like composite controls)
  - Controls the execution of child activities
  - Workflow programs are actually composite activities
  - Examples: `IfElse`, `While`, `Sequence`

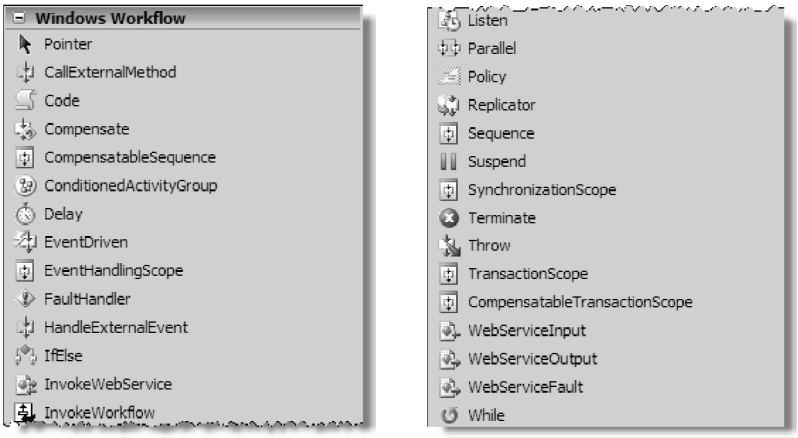
Types of Activities

- Action activities (*indicated as blue in designer*):
  - Do actual work such as creating / updating tasks, executing code, etc.
  - Code event handlers execute BEFORE work is done
- Event activities (*indicated as green in designer*):
  - Wait for an activity, causing the workflow to be persisted & an event is registered when to wake up
  - Code event handlers execute AFTER the event



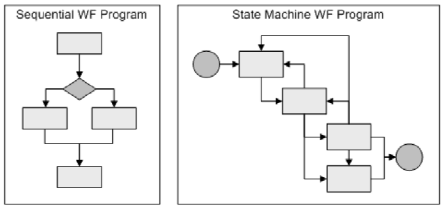
Workflow Foundation Base Activity Library

- Activities included in Windows Workflow Foundation



Types of Workflows

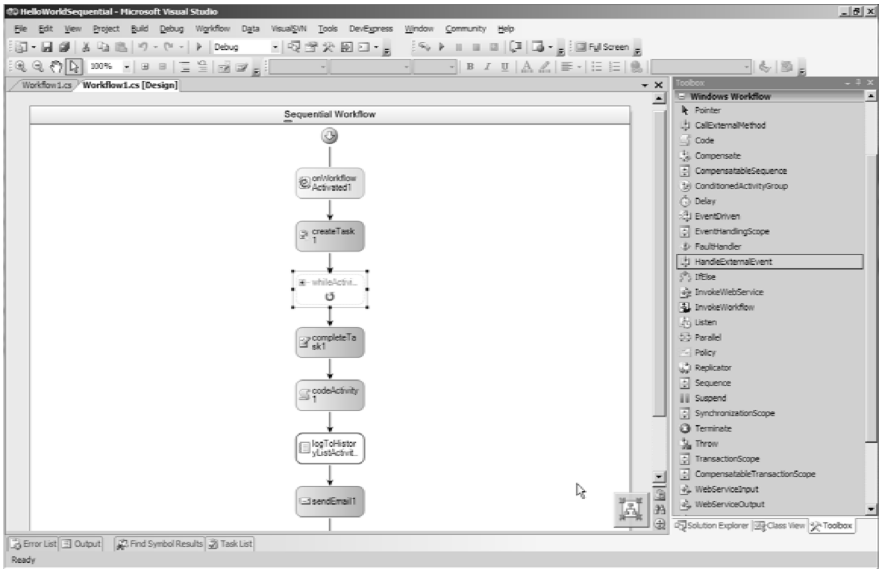
- Sequential workflow:
  - Similar to a flow chart containing predefined paths of execution with a beginning & end
- State machine workflow:
  - Based on concept of conditions & transition
  - No predefined path of execution
  - Path defined by the events during workflow execution



Developing Custom Workflows (VS2005)

- What you need:
  - .NET Framework 3.0
  - Visual Studio 2005 Extensions for Workflow Foundation
    `http://shrinkster.com/QJK`
- Visual Studio 2005 Extensions for Workflow Foundation:
  - Adds the base activity library to Visual Studio 2005
  - Adds a workflow designer to Visual Studio 2005
  - View code behind & create event handlers
  - Set public properties on activities using the Visual Studio Property tool window
  - Adds capability to debug workflows

Visual Studio with VSeWWF Installed



Developing Custom Workflows (VS2008)

- Visual Studio 2008
  - When Office tools (VSTO) optionally included in installation, new project templates added for creating workflows for use in SharePoint
  - Project templates facilitate ability to do F5 debugging
  - Wizard collects parameters for deploying, installing & activating Feature, creating the association and starting the debugger
  - Requires files to be in specific places in the project.
  - Requires .NET Framework 3.5 installed on target SharePoint servers

Workflow Foundation in SharePoint

- WSS 3.0 acts as the hosting application for the Workflow Foundation runtime
- Provides the persistence service to serialize & deserialize workflows to SharePoint content databases
- Adds SharePoint specific activities to workflow engine
- Adds a human element to workflows by incorporating tasks & forms
- Workflows are associated with lists / content types
- Workflow instances execute within the context of a list item / document

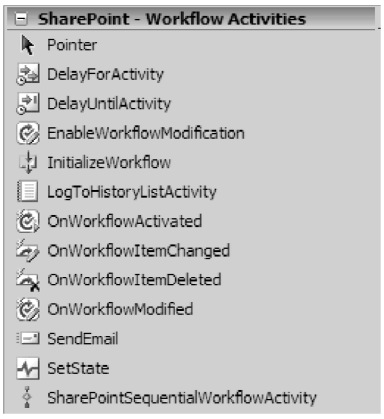
SharePoint Workflows Tasks

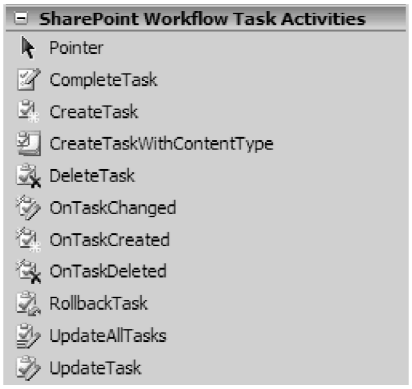
- SharePoint workflows primarily revolve around tasks
  - Tasks created / updated / completed using provided SharePoint specific workflow activities
  - Tasks are visible & editable by users
  - Tasks edited through browser or Office 2007 clients
  - Certain activities listen for certain conditions on tasks
  - Multiple tasks created in one workflow kept in sync with correlation tokens
- SharePoint includes multiple task related activities, including action & event activities

SharePoint Workflows History List

- SharePoint workflows can log messages to a history log
  - Specific history log for each workflow instance
  - SharePoint specific activities make it easy to log messages to the history
  - Add messages to the history list using the `LogToHistoryListActivity`

SharePoint Workflow Activity Library





OOTB SharePoint Workflows

- Approval (*MOSS only*)
  - Used for document routing… approvers can approve / reject a document, reassign task, or request changes.
  - *This is the "Parallel Approval" found on all /Pages lists*
- Collect Feedback (*MOSS only*)
  - Collects feedback by reviewers and sends to document owner when workflow completes
- Collect Signatures (*MOSS only*)
  - Used to collect digital signatures

OOTB SharePoint Workflows (Part 2)

- Disposition Approval (*MOSS only*)
  - Manages expiration & retention allowing participants to retain / delete expired documents
- Three-state (*WSS & MOSS*)
  - Tracks items as they move through various states
  - Can be configured to react in different ways depending on the current state of the workflow
  - Available after activating a Feature

Core SharePoint Workflow Definitions

- Workflow template
  - Assembly created using Visual Studio and deployed with a Feature
  - Contains all workflow logic & references to optional forms
- Workflow association
  - Binding of workflow template to list / content type
  - When binding, specify a name to refer to
  - One template can be associated with the same list / content type multiple times
- Workflow instance
  - Running workflow within the context of a list item / document

Forms in SharePoint Workflows

- SharePoint workflows utilize forms to collect information from people interacting with workflow
- Develop forms in one of two technologies:
  - ASP.NET 2.0 (ASPX pages)
  - InfoPath (rendered in InfoPath 2007 / browser)
- Types of SharePoint workflow forms
  - Workflow association
  - Workflow initiation
  - Workflow modification
  - Workflow task edit

Creating Workflow Forms as ASPX Pages

- Can run on all WSS 3.0 / MOSS 2007 installations
- Should be deployed to subdirectory within `[..]\12\TEMPLATE\LAYOUTS`
  - Developed as application pages
  - Should inherit from `LayoutsPageBase`
- Requires writing a significant amount of custom code in code behind of each form's ASPX page
- Provide no integration with the Office 2007 clients
- You must account for all data passing between workflow & form, as well as creation of any lists

DEMO: LitwareWorkflows

- Demo: **LitwareWorkflows**
  - Custom workflow template
  - Using Features for deployment
  - Workflow forms as ASPX pages
  - Examining the code behind of ASPX workflow forms

Creating Workflow Forms with InfoPath

- Can run only on MOSS 2007 installations
  - Requires Forms Services
- Deployed to a special library within Central Admin.
- Provides a rich RAD experience
- Rendered through browser / InfoPath 2007 client
- Requires significantly less coding
  - Very possible to create zero-code InfoPath forms
  - Only code in the workflow obtaining values in form
- Rich integration with Office 2007 clients

Building Workflows in SharePoint

- SharePoint Designer 2007:
  - Workflows bound at design time… you are creating an on-the-fly association
  - No easy way to copy / export workflows
  - Limited to sequential workflows
  - No debugging support
- Visual Studio:
  - Workflows built as template for deployment and reuse across multiple sites / environments
  - No limitations such as those that exist with SharePoint Designer 2007 developed workflows

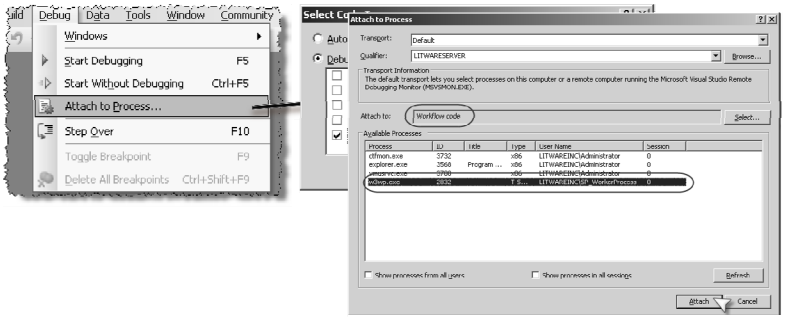
Developing Custom Workflow Templates

- What you need:
  - Everything you need to build workflows in Visual Studio 2005 with .NET Framework 3.0 (*including VSeWWF*)
  - WSS 3.0 SDK and / or MOSS 2007 SDK

    WSS 3.0 SDK: http://shrinkster.com/QJM
    MOSS 2007 SDK: http://shrinkster.com/QJN
- WSS & MOSS SDK installs add Visual Studio workflow project templates specific to SharePoint
  - WSS templates designed for workflows w/o forms / ASPX forms
  - MOSS templates designed for workflows w/ InfoPath forms
- OR Visual Studio 2008
  - No extra installation needed

Debugging Custom Workflows

- VSeWWF add debugging support to VS 2005
- No F5 debugging… *but it isn't that bad*
- Manually attach to the workflow debugger
- VS2008 does provide F5 debugging



Deploying Workflow Templates

- Deployment of custom workflow templates to site collections is done with Features
- Use different Feature definition & workflow element manifest file formats depending upon the forms technology used in the workflow

Deploying Workflow Templates Leveraging InfoPath Forms

- Deployment of custom workflow templates done using Features
- Feature definition - `feature.xml`
  - Uses specific Feature receiver to upload InfoPath forms
  - Property: `GloballyAvailable` = `true`
  - Property: `RegisterForms` = `[path to IP forms]\*.xsn`
- Element manifest – typically `workflow.xml`
  - `CodeBesideAssembly` & `CodeBesideClass`: class that contains the workflow
  - Form URLs: stock ASPX pages that host the IP forms
  - Metadata: URNs of IP forms

Feature.xml for Workflow with IP Forms

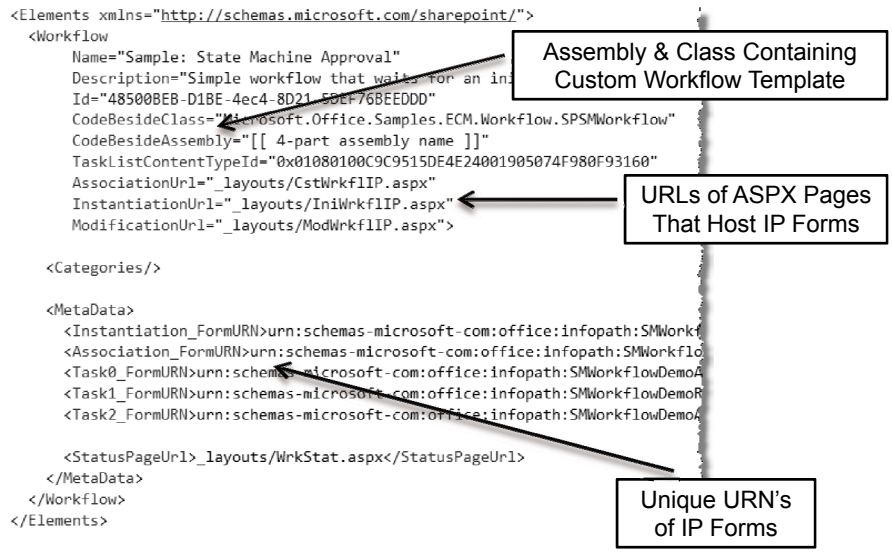
```
<?xml version="1.0" encoding="utf-8"?>
<Feature  Id="1E6D3BDD-9877-41ec-826C-EDE276EB56DD"
          Title="State Machine Workflow Demo"
          Description="Simple workflow that creates a review task and waits for the us
          Version="12.0.0.0"
          Scope="Site"
          ReceiverAssembly="[[ 4-part assembly name ]]"
          ReceiverClass="Microsoft.Office.Workflow.Feature.WorkflowFeatureReceiver"
          xmlns="http://schemas.microsoft.com/sharepoint/">
  <ElementManifests>
    <ElementManifest Location="workflow.xml" />
  </ElementManifests>
  <Properties>
    <Property Key="GloballyAvailable" Value="true" />
    <Property Key="RegisterForms" Value="*.xsn" />
  </Properties>
</Feature>
```

Assembly & Class of Feature Receiver that Uploads InfoPath Forms to Central Admin Library

Tells Feature Receiver Where To Find IP Forms w/I Feature Folder

Tells Feature Receiver IP Forms Should Go Into Central Admin Library

Workflow.xml for Workflow with IP Forms

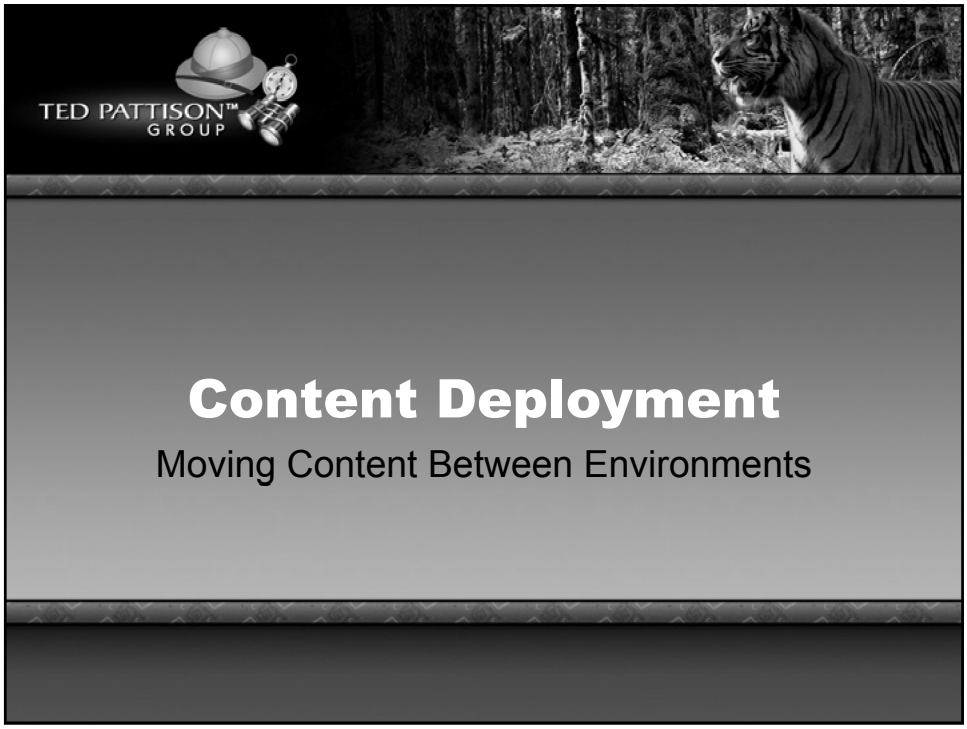


DEMO: MajorityApprovalWorkflow

- Demo: **MajorityApprovalWorkflow**
  - Custom workflow template that requires majority of people within a specific SharePoint group approve
  - Using InfoPath forms for approvers feedback collection
  - Using Features & solutions for deployment

Summary

- Windows Workflow Foundation core concepts
- What SharePoint brings to the Workflow Foundation
- Workflow Forms in SharePoint
- Out-of-the-box workflows in SharePoint
- Workflow creation & development in SharePoint
- Debugging workflows



Agenda

- Overview of content deployment
- Paths & jobs
- Deployment details
- Content deployment process
- Configuring content deployment via the browser-based user experience
- Configuring content deployment via the SharePoint API
  - Connected servers
  - Disconnected servers

Motivation of Content Deployment

- Founded on the concept that a single authoring environment would push site collection content to one (or more) destinations, such as…
  - Internal authoring environment, read only DMZ environment
  - Geo replication solution
- Single master solution… content deployment is a one-way process only
- Moves content, NOT code
  - Should be coordinated with WSS solution package deployment

Content Deployment Scenarios

- Two-tier
  - Authoring -> Production
- Three-tier
  - Authoring -> Staging -> Production
- N-tier
  - Flexible deployment model
- Content deployment is very flexible because the steps are the same for all site topologies

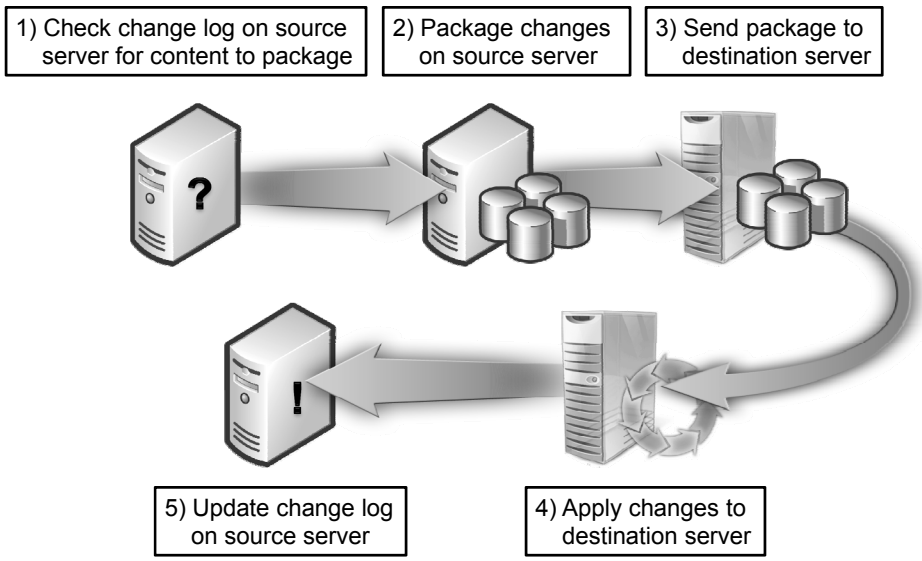
Content Deployment Paths & Jobs

- Path:
  - Defines the relationship between two site collections
  - Site collections can reside in same or different farms
  - Associated with one or more jobs
- Job:
  - Defines the schedule (when) content deployment is executed
  - Defines the scope such as entire site collection vs. collection of subsites

Deployment

- Content deployment only deploys changes since the last successful deployment
- Dependencies are automatically addressed and deployed
  - Page layout & Images
- Quick Deploy Jobs
  - Special type of deployment job created for every path in a site collection that has Publishing feature enabled
  - Runs every 15 minutes (configurable with 5m or 10m)
  - Allows content owners to push a page into production quickly (such as an urgent press release)

Content Deployment Process



Configuring Content Deployment

- Configure the source and destination server
  - Central Administration -> Operations
- Allow incoming content deployment jobs?
- Set import & export servers
- Optionally enable encrypted connection (HTTP / HTTPS) between source & destination servers
  - ALWAYS use encryption!!!

Configuring Content Deployment (Part 2)

- Configure temporary file location
  - Used to build package of exported content
- Configure temp file location with the following permissions:
  - **WSS_WPG & WSS_ADMIN_WPG**: Read, List, and Read & Execute on system temp drive
  - **WSS_ADMIN_WPG**: Full Control on content deployment temp folder
- Number of reports to retain on content deployment jobs

Content Setup and Configuration

- Create a new destination site collection
  - Must create using the Blank Site template
- Within Central Administration / Operations:
  - Create new path (from source -> destination)

    `Optionally deploy versions (or overwrite)`
    `Optionally deploy security info`
  - Create new job (schedule)
- Optionally configure Quick Deployment job
  - Automatically created if source uses Publishing features

DEMO: Setting up Content Deployment

- Demo: Using Content Deployment
  - Configuring Content Deployment settings on the server
  - Creating a Content Deployment Path
  - Creating a Content Deployment Job
  - Execute the Content Deployment Job

Content Deployment via the API

- `Microsoft.SharePoint.Publishing.Administration` namespace contains content deployment classes
- `ContentDeploymentConfiguration`
  - Use to get instances of content deployment on servers
- `ContentDeploymentPath[Collection]`
  - Use to get references and create paths
- `ContentDeploymentJob[Collection]`
  - Use to get references and create new jobs
- `ContentDeploymentJobType`
  - Enumeration used to specify type of job

DEMO: Content Deployment via API

- Demo: **ConnectedServers**
  - Creating & executing a content deployment job via API

API: Deployment Between Two Connected Servers

- For deploying content between servers…
- Configure source & destination farm properties
- Create a new path:
  - Configure source & destination site properties
  - Save
- Create a new job:
  - Configure the job & job type
  - Set the job's `Path` to path previously created
  - Save
- Run job
- Optionally delete the path

API: Exporting Content with Disconnected Servers

- Exporting content:
  - Configure export settings to export package via API
  - Run export to create a file
    (checks for exceptions & runs reports)
- Configure export settings using
  `SPExportSettings()` & `SPExport()`
  - Site to export
  - Where to save the exported file (overwrite existing file?)
  - Include content versions & security in exported objects?

API: Importing Content with Disconnected Servers

- Importing content:
  - Configure import settings to import package via API
  - Run import
- Configure import settings using
  `SPImportSettings()` & `SPImport()`
  - Target site to import
  - Keep object ID's, security, user & date/time info?
  - Update versions & include security?
  - Enable event receivers on import?
    Significant performance hit if enabled

Content Deployment "Tips"

- Ensure MOSS SP1 & Infrastructure Updates installed
  - Should be at least version 12.0.0.6318
- All servers configured as export & import need to host an instance of Central Administration
- Ensure sufficient disk space is available
- Install required Features on destination server
- Don't activate custom Features manually on destination server
- Don't expect incremental deployment will deactivate Features in destination server
- Ensure deployment jobs don't run in parallel
- Create site collection w/o applying template
  - STSADM.EXE –o createsite
    –url "http://foo"
    –owneremail "admin@litwareinc.com"
    –ownerlogin "LITWAREINC\administrator"
    –ownername "LitwareAdmin"

Content Deployment "Gotchas"

- Don't deploy one site collection to another within the same Web application
  - GUIDs are copied, not regenerated, breaking uniqueness within a content database
- Don't edit content on the destination server
  - Content deployment expects the destination server to be exactly as it left it
- Don't rename the "Pages" library

Summary

- Overview of content deployment
- Paths & jobs
- Deployment details
- Content deployment process
- Configuring content deployment via the browser-based user experience
- Configuring content deployment via the SharePoint API
  - Connected servers
  - Disconnected servers

Agenda

- Multilingual capabilities in MOSS 2007 WCM
  - Resource files
  - Variations
- SharePoint language packs
- Understanding variations
  - Variations labels
  - Customizing
- Navigation experience & implications
- Web Part personalization implications
- Search implications

What does "Multilingual" Mean?

- Managing content in various languages
- Navigate a site in preferred language
- Collaborate with people in different regions in different languages from within same application
- Manage and administer personal site using preferred language
- Search and browse content across organization in preferred language

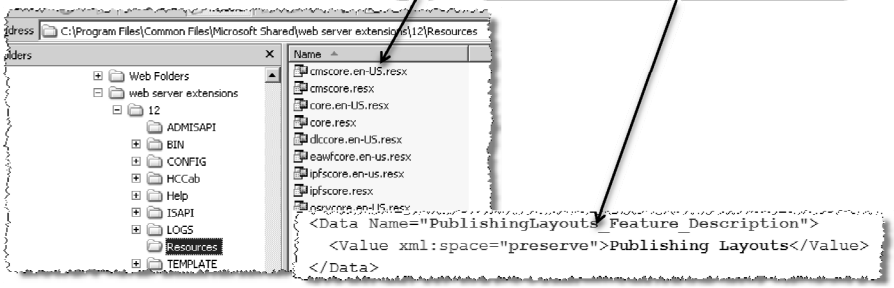
Supported Multilingual Scenarios

- Read & manage content, site navigation & search
  - Full experience OOTB, but customization for search required
- Site admin
  - Fully OOTB
- Site collection admin
  - Single language experience based on the site collection's root web's language setting
- Farm administration
  - Single language experience, depending on installed SharePoint language pack

WSS Resource Files

- Provide an easy way for localizing a solution
- Special tokens used in XML files
- Ability to target a specific token file

```
<Feature  Id="D3F51BE2-38A8-4e44-BA84-940D35BE1566"
          Title="$Resources:cmscore,PublishingLayouts_Feature_Title;"
          Description="$Resources:cmscore,PublishingLayouts_Feature_Description;"
```



SharePoint Language Packs

- SharePoint Language Pack
  - Designed for use in WSS v3 in stand-alone config
  - In farm configuration, same language pack must be installed on all farm servers
- Server Language Pack
  - Designed for use in Office Server System 2007

    `MOSS 2007`
    `Forms Server 2007`
    `Project Server 2007`
    `MOSS 2007 for Search`
  - Includes all resources of SharePoint Language Pack (no need to install both)

SharePoint Language Packs (Part 2)

- Users can specify language upon creation time of site collection or site
  - Sets the Language & Country ID (LCID) for whole site or site collection
  - Defines UI language
  - Defines text input interpretation
  - Defines locale specific data is displayed (currency, numbers, dates, etc.)

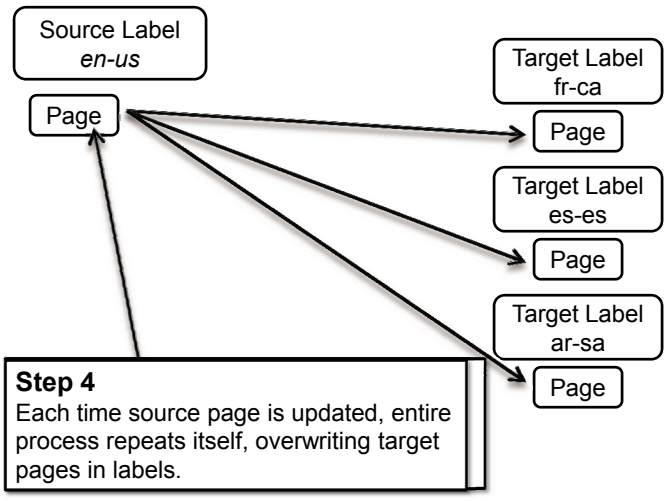
Variations

- Enables site owners / admins to maintain a copy of content in different sites in a site collection
- Each site copy represents a different locale
- Variation involves creating a hierarchy of sites containing copies of the root site

Variation Process

- Once variation configuration & labels are setup…
- Create new page / site within source label
- Once published, SharePoint creates copies in each variation label, but doesn't publish them
- Each site owner then:
  - Translates
  - Change page layout
  - Make other culture specific changes
- Page then goes through it's own workflow process

Variation Process Animation



Configuring Variations in a Site Collection

- From the top-level site within a site collection,
  Site Settings -> Variations
  - Variation Home (Container)
  - Automatic Creation
  - Recreate Deleted Target Page
  - Update Target Page Web Parts
  - Notification
  - Resources
- Limit of only one variation hierarchy per site collection

Creating Variation Labels

- From the top-level site within a site collection,
  Site Settings -> Variation Labels
- Create a label for each language
- Specify:
  - Name (what appears in URL)
  - Display Name (what appears in navigation)
  - Locale (culture label applies to)
  - Hierarchy Creation
    (what to create when hierarchy is created)
  - Source Variation (exactly one label in a Variation config)

Variation Miscellaneous

- Allows site owners to define and implement different master pages, page layouts, and images in each variation label
- Translating content:
  - Not automatic
  - Can occur within site just like content authoring
  - Can be exported, translated by 3rd party, and imported
- Automatic redirection logic:
  - Determined using HTTP header HTTP_ACCEPT_LANGUAGE value
  - Redirection logic can be customized (see MOSS SDK)

Navigation Experience

- Control provided enabling people to navigate to different variation labels
- Navigation control customizable by changing `LabelMenuConfiguration` property on `VariationHierarchyDataSource` control:
  - 1 – user redirected to same page on selected label
  - 2 – user redirected to default page on same path
  - 3 – user redirected to default page of selected label

Web Part Personalization Experience

- Some Web Parts are not aware of variations
  - WSS List Web Part is bound to a specific list by GUID
- Custom Web Parts can also be affected… you need to account for this in your custom Web Parts
- During variation setup & configuration, can set option to not synchronize Web Part to target label

Implications for Search

- When a search is executed, query engine passes query to word breaker for specified language
  - Word breaker: breaks words into logical words / phrases
  - If not word breaker specified for the query's language, neutral word breaker used (white spaces break words)
- Next, results are passed through language specific stemmer (addresses singular / plural, past / present / future tense)
- Search also uses a language specific dictionary and thesaurus

DEMO: Implementing Variations

- Demo: Implementing variations
  - Configuring variation settings within a site collection
  - Create a new variation hierarchy with labels
  - Demonstrate automated content duplication
  - Translate localized content and customize layout

Summary

- Multilingual capabilities in MOSS 2007 WCM
  - Resource files
  - Variations
- SharePoint language packs
- Understanding variations
  - Variations labels
  - Customizing
- Navigation experience & implications
- Web part personalization implications
- Search implications