

Module 6: Extending Web Content Management

Date published:

February 2009

Summary:

This module describes how to customize Microsoft® Office SharePoint® Web Content Management so that you can include customizations in solution architectures.

See [Web Content Management Training Modules](http://go.microsoft.com/fwlink/?LinkId=141931) (http://go.microsoft.com/fwlink/?LinkId=141931) for a complete list of the available downloads.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, SharePoint, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

Module 6: Extending Web Content Management 1

Module 6 Overview 1

Objectives 1

Lesson 1: Customizing Content Presentation 1

Objectives 1

Creating a Custom Field Control 1

Field Types and Field Controls 2

Creating a Custom Field Control 2

Deploying Custom Field Controls 2

Customizing Navigation 3

Built-in Navigation Controls 3

Navigation Model 4

Creating a Custom Navigation Control 4

When to Consider a Custom Navigation Control 4

Working with Site Map Providers 5

Considerations for Developers 5

Customizing the Site Variation Redirection Logic 5

Default Site Variation Logic 6

How Developers Modify the Site Variation Redirection Logic 6

Using the Variation Navigation Control 7

Lesson 2: Customizing Content Editing 7

Objectives 7

Customizing the Page Editing Toolbar 7

Facilities on the Page Editing Toolbar 8

How to Customize the Page Editing Toolbar 8

Customizing the HTML Editor Field Control 8

Controlling Editor Actions 9

Controlling Table Formats 9

Adding Extra Buttons 10

Lesson 3: Customizing Content Deployment 10

Objectives 10

Capabilities of the Content Deployment and Migration API 11

Exporting Office SharePoint Content 11

Special Features 11

Importing Content 12

Custom Content Deployment Scenarios 13

Review of Module 6 13

Module 6 Overview

As the architect, you might find that your organization has unusual or unique Web Content Management (WCM) requirements that require customization of Microsoft® Office SharePoint® Products and Technologies. You can customize many aspects of Office SharePoint WCM to suit your situation. For example, you can create custom controls for presenting and editing items on your pages.

This module describes the ways in which you can customize Office SharePoint WCM so that you can include customizations in solution architectures.

Objectives

After completing this module, you will be able to:

* Describe how developers can customize the way Office SharePoint Products and Technologies present content to site visitors ([Lesson 1](#Lesson1))
* Describe how developers can change the way that users edit content in Office SharePoint sites ([Lesson 2](#Lesson2))
* Include custom content presentation in WCM solution architecture ([Lesson 3](#Lesson3))

Lesson 1: Customizing Content Presentation

Office SharePoint Products and Technologies include versatile facilities that present data to site visitors. You can often create the site that you want by using these facilities without any customization. However, for those with unusual or unique requirements, Office SharePoint provides several opportunities to customize content presentation.

Objectives

After completing this lesson, you will be able to:

* [Describe custom field controls and situations in which you should consider creating them](#CreatingCustomFieldControl)
* [List the alternative navigation controls and providers that Office SharePoint Products and Technologies include](#CustomizingNavigation)
* [Understand how developers create custom navigation controls](#CreatingCustomNavigationControl)
* [Describe situations in which you should consider customizing the site variation redirection logic](#CustomizingSiteVariationRedirect)

Creating a Custom Field Control

You can store many types of data in Office SharePoint Products and Technologies WCM solutions by selecting appropriate field types. Each field type is rendered on a Web page by a field control. Occasionally, you might require a custom field type in your site and in that case you should plan a custom field control to display your data to users.

This section contains the following topics:

* [Field Types and Field Controls](#FieldTypesandFieldControls)
* [Creating a Custom Field Control](#CreatingaCustomFieldControl)
* [Deploying Custom Field Controls](#DeployingCustomFieldControls)

Field Types and Field Controls

Developers create custom field types and custom field controls by writing managed code. Custom field types and controls can extend the functionality of built-in fields and controls.

Before you plan for a custom field type and a custom field control in your architecture, consider first whether you can satisfy your requirements by using a built-in field type. For example, to store a postal address you might use a multiline text field.

Consider using a custom field type in the following circumstances:

* Similar data appears many times throughout your site, or across multiple sites. You can reuse custom field types to maximize the return on investment in custom code.
* You require custom data validation. You can build data validation into your custom field type to ensure that users enter sensible, valid data.
* The built-in controls do not display data to your satisfaction. You have precise control over the display of data when you create a custom field control.

Creating a Custom Field Control

When a developer creates a custom field type and a custom field control, they must:

* Create a class library project in the Microsoft® Visual Studio® 2005 or Microsoft® Visual Studio® 2008 development system. This kind of project compiles into a .dll file.
* Add a custom field type to the project. The custom field type is a class that extends one of the built-in field types. The developer can extend the built-in field type by overriding methods. For example, to add custom data validation, the developer could override the **GetValidatedString** method.
* Add a custom field control to the project. The custom field control is a class that extends one of the built-in field controls. The developer can customize the display of information by overriding methods.
* Create a field type definition file. The field type definition file is an XML file that contains information that the Office SharePoint server uses to instantiate and run the field type.

Deploying Custom Field Controls

When the developer has completed and compiled the field control and field type, you must complete the following steps to deploy them (ideally thorough an Office SharePoint solution):

► Deploy the custom field controls

1. Install the .dll file into the Global Assembly Cache (GAC) on all the front-end Web servers in your server farm.
2. Copy the field type definition file to the following location:

C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\XML

1. Add a <SafeControl> tag to the Web.config files for the appropriate Web applications for the custom field control.

Customizing Navigation

A clear navigation design is crucial to all Web sites because it enables users to locate information rapidly and intuitively. For this reason, you might want to customize the way that Office SharePoint Products and Technologies formulate and present navigation information. The following table shows aspects of navigation customization.

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| [Built-in navigation controls](#BuiltinNavigationControls) | Global  Current  Breadcrumb  Summary links  Tree view  Content by query |
| [Navigation model](#NavigationModel) | Data sources  Site map providers  Navigation controls |

Built-in Navigation Controls

The navigation controls built into Office SharePoint Products and Technologies provide advanced functionality and a great deal of flexibility. They should be carefully considered before you consider including a custom navigation solution in your architecture. The controls are as follows:

* + **Global navigation control**:   Appears as top link bars or tabs in default site templates. Administrators can specify what is displayed here. By default, the control displays all subsites one level below the top-level site. This navigation is implemented with the Office SharePoint menu control that you can use to create other navigation elements.
* **Current navigation control**: Also known as the Quick Launch and displayed on the left in default site templates. Administrators can extensively control its content. For example, they can select whether the navigation displays a menu or a tree view of the site.
* **Breadcrumb navigation control**: Displays the user’s current position in the site hierarchy, with links to all higher levels.
* **Tree view control**: Displays a hierarchy of the site. You can configure the number of levels to show and make other adjustments.
* **Content Query Web Part**: Displays a dynamic view of content in your Office SharePoint site that satisfies a query that you specify. For example, you could use this Web Part to display recently published pages or pages by a specific author.

These controls can be used anywhere in your site, but they are commonly placed on master pages so that all pages display a consistent navigation interface. You can adjust the appearance of all these controls by using style sheets.

Navigation Model

Windows® SharePoint® Services 3.0 and Microsoft® Office SharePoint® Server 2007 use the same modular navigation model available in ASP.NET 2.0 as follows:

* **Data sources**. ASP.NET 2.0 sites often use an XML file as the source of site hierarchy data. In Office SharePoint Products and Technologies, the content database is used as the source.
* **Site map providers**. Site map providers query the data source and build a hierarchy of site map nodes that represent the pages in the site. You can configure these providers by specifying, for example, the number of levels to display.
* **Navigation controls**. The navigation controls described above are the top level in the navigation model.

If you want to use a data source other than XML files or the Office SharePoint content database, your developers could implement a custom site map provider. However, if you do this, the site map does not automatically reflect changes in the content database and you must take steps to ensure that your navigation data source constantly reflects your actual content.

Creating a Custom Navigation Control

Office SharePoint Products and Technologies include highly functional and customizable navigation controls. However, if these do not satisfy your requirements, you can plan a custom navigation control as part of your architecture and implement it in managed code.

This section contains the following topics:

* [When to Consider a Custom Navigation Control](#WhenConsiderCustomNavControl)
* [Working with Site Map Providers](#WorkingwithSiteMapProviders)
* [Considerations for Developers](#ConsiderationsforDevelopers)

When to Consider a Custom Navigation Control

Developing a custom navigation control involves a significant investment in development time. Before beginning this development effort, ensure that the existing controls, with their broad scope for customization, do not satisfy your requirements.

Consider the following cases:

* If the rendering of data in a built-in control does not satisfy requirements, can you tweak the control by editing the relevant style sheet?
* If the nodes displayed by a built-in control do not satisfy requirements, can you adjust the settings in the user interface; for example, by changing the number of levels displayed?
* If the nodes displayed by a built-in control do not satisfy requirements, does this reflect an illogical site structure that you should reconsider?
* If the nodes displayed by a built-in control do not satisfy requirements, can you display the right nodes by using the right query in the Content Query Web Part?

If your answer to all of these questions is “No,” consider a custom navigation control.

Working with Site Map Providers

Site map providers evaluate the contents of the Office SharePoint content database and construct a hierarchy of site map nodes that describe the site. Any custom navigation control that you design must evaluate these site map nodes and render them to the user during page load.

Site map providers handle security trimming for Office SharePoint Products and Technologies. They remove from the hierarchy any nodes that the current user does not have permission to read. Therefore, your developers do not have to write code that evaluates security permissions when they create a custom navigation control.

Considerations for Developers

If you have decided to use a custom navigation control, your developers should bear the following points in mind as they write the code:

* If your proposed custom navigation control is similar to any of the existing controls, you should consider extending them — developers can do this by creating a control that inherits from a built-in control.
* Developers should override the **Page\_Load** event in their custom code and, within this event, traverse the **SiteMapNode** objects supplied by the site map provider and render the output.
* The most important **SiteMapNode** properties to use in your rendering code are:
* **Title**. This is the most appropriate text to render for your node.
* **URL**. Use this to construct the link so that users can click the node and view the page.
* **Description**. This is often used to construct a Tool Tip that appears when the mouse hovers over the node in the navigation.

Customizing the Site Variation Redirection Logic

As you saw in Module 2, site variations enable you to redirect visitors to an appropriate variation of your site based on headers in their request. Site variations are used to create versions of the site in alternate languages or for alternate devices such as mobile phones and personal digital assistants (PDAs).

This section contains the following topics:

* [Default Site Variation Logic](#DefaultSiteVariationLogic)
* [How Developers Modify the Site Variation Redirection Logic](#HowDevelopersModifytheSiteVariation)
* [Using the Variation Navigation Control](#UsingVariationNavigationControl)

Default Site Variation Logic

When you configure site variations, Office SharePoint Server 2007 creates a new page called **VariationRoot** in the source site’s Pages library. This page is based on the VariationRootPageLayout.aspx page layout that includes the **VariationsRootLanding.ascx** control. This page is set as the default page for the site.

When a request arrives from a user, the **VariationRoot** page loads and runs the **VariationsRootLanding.ascx** control. This control checks the browser’s accept lang setting in the request and matches it to the appropriate site variation. It forwards the user to the site variation that the control selects.

You might want to modify this logic so that the selection is based on some field other than the accept lang setting. For example, for mobile devices, you might wish to forward the browser to a site variation that is designed to display on small screens. In such cases, you must modify the site variation redirection logic.

How Developers Modify the Site Variation Redirection Logic

Developers can modify the site variation redirection logic in three ways:

* **By editing the VariationsRootLanding.ascx file**. In this simple approach, developers edit the redirection logic in the ASP.NET control where it is stored. This approach is rapid, but you must modify the file on all front-end Web servers in your farm. You should also consider that modifying the native SharePoint files in this way is not supported by Microsoft, and know that this kind of modification can be overwritten by service packs and hotfixes.
* **By creating your own copy of the VariationsRootLanding.ascx file.** In this other simple approach, developers make a new copy of the original ASP.NET control, place it in the 12\TEMPLATE\CONTROLTEMPLATES directory, and modify the VariationRootPageLayout.aspx in your master page gallery to reference the new version of the control. This approach is also relatively quick, but you must install the file on all front-end Web servers in your farm and update the page layout on all site collections to which you’d like to apply the new logic. This method sidesteps any support issues.
* **By copying the redirection logic into the page layouts**. In this approach, developers copy the redirection logic into the page layout file for the home page and modify it there. In this way, they can modify the logic for the whole site collection in a single location. To complete this process, you also must update the Web.config file on each of the front-end Web server computers to allow the script to execute.

Note: If you take this approach, use caution when setting the **AllowCompilation** flag for the VariationRootPageLayout.aspx file. Using inline code means that someone could inject malicious code into the page.

* **By creating a compiled assembly**. In this approach, developers create a new compiled .dll file and copy the redirection logic into it, where they modify it. As with the method that involved making a copy of the VariationsRootLanding.ascx file, you would then reference this assembly in the page layout file to put that logic to work. There is a smaller risk that the logic can be modified by malicious users, because it is compiled into a .dll file, but the assembly must be deployed to all front-end Web servers before it can be used.

For any of the preceding approaches, you would want to use the solutions and features frameworks to deploy the customization.

Using the Variation Navigation Control

In most cases, users are redirected to the appropriate site variation without their knowledge. If they want to see an alternate variation of the site, they must select their preferred language in the browser before they visit your site. If you want to enable users to choose the variation easily after they arrive, you can use the variations label menu control. You should place the control on a master page to ensure that it renders on all pages in your site.

Lesson 2: Customizing Content Editing

Office SharePoint Server 2007 provides opportunities for customization to the content-editing facilities. You might need to consider such modifications if the authors in your organization work in an unusual or unique way.

Before you consider the techniques described in this lesson, bear in mind that custom workflows, the approval process, and other Office SharePoint features enable a wide variety of configurations with no custom code development. Only consider customizing content editing if you cannot satisfy your requirements with out-of-the box features.

Objectives

After completing this lesson, you will be able to:

* [Architect a content editing system with a customized Page Editing toolbar](#CustomizingPageEditingToolbar)
* [Design a WCM system with a customized HTML Editor field control](#CustomizingHTMLEditorFieldControl)

Customizing the Page Editing Toolbar

When content authors and editors make changes to a Web page in Office SharePoint Products and Technologies, the front-end Web server displays the page to them in edit mode. Controls such as the HTML Editor field control render in edit mode and let editors change the content of the page. In addition, the Page Editing toolbar is displayed at the top with tools that are essential to the editing process:

* [Facilities on the Page Editing toolbar](#FacilitiesonPageEditingToolbar)
* [How to customize the Page Editing toolbar](#HowCustomizePageEditingToolbar)

Facilities on the Page Editing Toolbar

The Page Editing toolbar consists of three sections:

* **Page Status bar**. Located along the top of the Page Editing toolbar, the page status bar displays version information, check-out status, and the publication start date for the page.
* **Page editing menu**. Located in the lower left area of the Page Editing toolbar, the page editing menu has three drop-down lists. The Page list enables authors to save, check in, or delete the page. The Workflow list enables authors to start workflows and move the page through the steps. The Tools list includes a spelling checker, preview links, and other facilities.
* **Quick access buttons**. Located in the lower right area of the Page Editing toolbar, the quick access buttons enable authors to check in the document, publish it, and take other actions, depending on the document’s approval status.

How to Customize the Page Editing Toolbar

The Page Editing toolbar is an ASP.NET user control, implemented in the C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\CONTROLTEMPLATES\publishingconsole.ascx file.

This control formulates its user interface by reading two XML files called **EditingMenu.xml** and **QuickAccess.xml**, which are both located in the C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\LAYOUTS\EditingMenu folder.

However, to ensure compatibility with other sites and to avoid supportability issues, you should not edit these files directly. Instead, place your modifications in two custom markup XML files. These files are called CustomEditingMenu.xml and CustomQuickAccess.xml and you can find them in the Editing Menu folder within the master page gallery for the site collection. In this location, you can check the files out and publish them when you have completed your modifications.

The XML elements you place within these files must conform to the XML schema at C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\LAYOUTS\EditingMenu\ConsoleConfigurationSchema.xsd.

Customizing the HTML Editor Field Control

When editors or authors enter edit mode in a Web page in an Office SharePoint Products and Technologies site, they see, in addition to the Page Editing toolbar, the HTML Editor field control. When authors click the **Edit Content** option, they can enter new text and choose from a wide variety of formatting tools.

This section contains the following topics:

* [Controlling Editor Actions](#ControllingEditorActions)
* [Controlling Table Formats](#ControllingTableFormats)
* [Adding Extra Buttons](#AddingExtraButtons)

Controlling Editor Actions

To use the HTML Editor field control, you must use a content type that includes the a column of the Publishing HTML type (such as Page Content or Image Caption). If this is the case, you can add the editor to the content type’s page layout by including the markup such as the following for the Image Caption column.

<PublishingWebControls:RichHtmlField id="Caption"

FieldName="PublishingImageCaption" AllowExternalUrls="false"

AllowFonts="true" AllowReusableContent="false" AllowHeadings="false"

AllowHyperlinks="false" AllowImages="false" AllowLists="false"

AllowTables="false" AllowTextMarkup="false"

AllowHTMLSourceEditing="false" DisableBasicFormattingButtons="false"

runat="server"/>

You can control the facilities that are presented to the authors and editors by using attributes such as **AllowImages** and **AllowTables**. In the previous example, most facilities are disabled by setting attributes to **false**, but users can change fonts because **AllowFonts** is set to **true**.

When you plan the use of the HTML Editor field control, you should consider actions that might disrupt your site. For example, if you enable users to set their own fonts, they might write content that deviates from the look and feel imposed by your style sheets.

Controlling Table Formats

If you set the **AllowTables** attribute to **true**, users can add an HTML table to content. When authors click the New Table button, a dialog box enables them to choose table properties, including the format. Office SharePoint Products and Technologies have five predefined table formats to choose from, but you can also create your own styles by using style sheets.

To do this, you must add classes to the master page style sheet or an inline style sheet within the page layout. The classes that you add must be named in the format **PREFIXTableXXX-NNN** where:

* **PREFIX** is either **ms-rte** or the value that you specify in the **PrefixStyleSheet** attribute of the field control.
* **XXX** is the table section such as **EvenRow** or **HeaderFirstCol**.
* **NNN** is a name to identify the style. This is visible to users when they choose the style.

The following is an example style sheet.

.ms-rteTable-1

{border-collapse:collapse; border-top:gray 1.5pt; border-left:gray 1.5pt;

border-bottom:gray 1.5pt; border-right:gray 1.5pt;border-style:solid;}

.ms-rteTableHeaderRow-1

{color:Green;background:yellow;text-align:left}

.ms-rteTableHeaderFirstCol-1 {padding:0in 5.4pt 0in 5.4pt;}

.ms-rteTableHeaderLastCol-1 {padding:0in 5.4pt 0in 5.4pt;}

.ms-rteTableHeaderOddCol-1 {padding:0in 5.4pt 0in 5.4pt;}

.ms-rteTableHeaderEvenCol-1 {padding:0in 5.4pt 0in 5.4pt;}

Adding Extra Buttons

You can also add to the facilities in the HTML Editor field control. Consider using this technique if users have a common need to add specific content quickly. For example, you might add a button that inserts the name of a product, properly formatted and trademarked. Developers add buttons to the HTML Editor field control in the following way.

► Add buttons to the HTML Editor field control

1. Edit the RTE2ToolbarExtension.xml file. This is found in the Editing Menu folder in the master page gallery for the site collection. Developers add an <RTE2ToolbarExtraButton> tag to the <RTE2ToolbarExtensions> element in this file.
2. Create a JavaScript file that runs when a user clicks the button. This file must implement the **onClickCallback** method and the **onResetStateCallback** method.
3. Upload the JavaScript file to the location specified in the <RTE2ToolbarExtraButton> tag. This must be done on every front-end Web server in the server farm — ideally, this is performed by means of a solution.

Lesson 3: Customizing Content Deployment

Office SharePoint Server 2007 includes an advanced and versatile content deployment system (CDS). However, occasionally you might find that unusual or unique requirements cannot be satisfied using the out-of-the-box CDS. In this case, you should consider a custom solution. The Content Deployment and Migration application programming interface (API) enables developers to create deployment tools that are targeted closely to the needs of their organization.

Objectives

After completing this lesson, you will be able to:

* [Describe the capabilities of the Content Deployment and Migration API and architect custom solutions that use it](#CapabilitiesofContentDeployMigrnAPI)
* [Describe scenarios in which you must customize the CDS in Office SharePoint Products and Technologies](#CustomContentDeploymentScenarios)

Capabilities of the Content Deployment and Migration API

The Content Deployment and Migration API is a hierarchy of Microsoft .NET Framework classes that can export and import content from one Office SharePoint technologies location to another; for example, from site to site or farm to farm. Office SharePoint features, such as the CDS, variations, and the export and import tools in STSADM.exe, are built on objects in the Content Deployment and Migration API. Your SharePoint developers can access this API in their managed code, so they can construct custom content deployment solutions with it. To design such custom solutions, you must understand the capabilities of the API. The following table will guide you through this section.

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| [Exporting Office SharePoint content](#ExportingOfficeSharePointContent) | Site collections  Sites and subsites  Lists and document libraries  Lists items and documents |
| [Special features](#SpecialFeatures) | Dependent objects  Compression  Incremental export |
| [Importing content](#ImportingContent) | Preserving object identity  Fixing links |

Exporting Office SharePoint Content

All operations with the Content Deployment and Migration API are either exports or imports. You must therefore begin by exporting content to a file. You can export the following objects:

* **Site collections**: All content in the site and subsites is exported.
* **Sites and subsites**: When you export a site or subsite, you can choose whether descendant objects such as subsites, lists, and libraries are exported.
* **Lists and document libraries**: By selecting a list or document library, you can export all of the entries in a single operation.
* **List items and documents**: You can create very granular export operations by selecting a single item or document.

Special Features

When you export an object such as a Web page in a document library, you might need to include dependent content, such as the images that display on that Web page. By using export settings, the developer can ensure that such content is included, without a specific export operation for each dependent item.

If you export a large site collection or a document library with large documents, you might find that the export file is large. In this case, the developer can specify a compression option, so that all exported objects are saved in a single compressed cabinet file.

Developers can also export only those items that have changed since the last export. This is called an incremental export. To perform an incremental export, developers must save a change token during an export. This string can be saved to a file or database field. During a subsequent incremental export, developers supply this change token to indicate the last change that was exported. All subsequent changes are exported to the file.

Importing Content

To import content to a new location, code must supply the location of the export file, the location to import to, and optionally other settings.

One possibility is to maintain the identity of the imported objects. When you use this option, each object will have the same globally unique identifier (GUID) and URL as the object in the source location. This method is used to move an entire site collection to another farm, usually the staging or production environment.

Note: The import option can only be used were the parents of all objects are present in the destination. Therefore, it is not possible, for example, when you move list items into a different list.

When you import content into a new location, you might experience a problem with hyperlinks in that content. If, in the original location, the hyperlinks linked to items also in the original location, they will point to the wrong location after import. So, for example, a user in the production environment might be forwarded to a page in the staging environment, to which the user does not have access. To avoid this issue, developers can write code that substitutes the original links for correct links for each imported item.

Custom Content Deployment Scenarios

The content development system (CDS) is a highly versatile system that can be used to deploy content to many locations. It is also simple to schedule CDS jobs to deploy content at desirable times, such as those with low user demand. Before you consider designing a custom CDS, verify that the current system is not capable of satisfying your requirements. The following table lists some scenarios in which a CDS might be required.

|  |  |
| --- | --- |
| [Custom tools](#CustomAdministrativeTools) | Give administrators or authors simple tools with specific capabilities |
| [Granular deployment](#GranularDeployment) | Deploy individual items or documents |
| [Filtered deployment](#FilteredDeployment) | Deploy only objects that pass tests |

* **Custom Administrative Tools**: Although the CDS is versatile, it might be too complex for authors or inexperienced administrators. By building your own deployment tool, you can enable such personnel to deploy content rapidly, while enforcing the content policies of the organization. You can also integrate this functionality with other tasks such as content approval, so that users have a single tool that can be used for all their tasks.
* **Granular Deployment**: The Content Deployment and Migration API can select individual items or documents for export. It is therefore possible to be extremely specific about the items that you deploy.
* **Filtered Deployment**: The **import** operation creates new instances of all the objects in the export file. However, it is possible for code to interrupt the importation of an object. In this case, you could run tests on the object, for example, to enforce an organization policy, and import only those objects that pass.

Review of Module 6

* Customizing content presentation
* Customizing content editing
* Customizing content deployment

References

The following videos provide supplemental information to these modules. There is no one-to-one correspondence between the modules and the videos:

* [Video 1](http://go.microsoft.com/fwlink/?LinkId=140097&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140097&clcid=0x409)
* [Video 2](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%202) (http://go.microsoft.com/fwlink/?LinkID=140063&clcid=0x409)
* [Video 3](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%203) (http://go.microsoft.com/fwlink/?LinkID=140068&clcid=0x409)
* [Video 4](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%204) (http://go.microsoft.com/fwlink/?LinkId=140101&clcid=0x409)