Sitecore CMS 7.0

Reusing and Sharing Data

Tips and Techniques for Developers

Table of Contents

Chapter 1

Reusing and Sharing Data

Sitecore developers should read this document to determine a strategy for reusing data on multiple pages of a websites or sharing data to multiple managed websites and external applications.[1]

The Sitecore layout engine can employ a number of techniques to reuse data on different pages or share data between managed websites and external applications. Multiple pages and managed websites can reuse and share presentation components that reuse and share data.[2] Presentation components can include data from any number of Sitecore items. You can format data differently based on numerous criteria, such as the type of device requesting an item. You can insert reusable text snippets into Rich Text Editor field values, and you can dynamically replace dictionary entries and simple tokens with reusable character sequences. You can share data between multiple managed websites and external applications, and you can cause items to appear at multiple URLs or locations within the content tree.[3]

This chapter contains the following sections:

- Sharing and Reusing Data with Presentation Components
- Sharing and Reusing Items and Field Values
- Sharing and Reusing Character Sequences

[1] For more information about multiple managed websites, see http://sdn5.sitecore.net/Articles/Administration/Configuring%20Multiple%20Sites.aspx.
[2] For more information about presentation components, see the manual *Presentation Component Reference*.
[3] For more information about item URLs, see the manual *Dynamic Links*.

1.1 Sharing and Reusing Data with Presentation Components

You can use the following techniques to share and reuse data using presentation components.

1.1.1 Devices

The layout engine can invoke different presentation components when different types of devices, such as web browsers or mobile devices, request an item. You can use devices to format data differently based on any information in the HTTP request context.

For more information about devices, see the manual *Presentation Component Reference.*

Important
You must define criteria to activate each device.

For example, two managed websites with the same content and information architecture use different presentation components to render a single hierarchy of Sitecore items.

You can use a device to meet these requirements. Create a device to associate with the second managed website. For each of the shared items, associate the presentation components used by the second managed website with that device in layout details. Specify that device in the managed website definition for the second managed website.

The managed website definitions share a single set of items. Sitecore evaluates the layout details for the appropriate device to determine which presentation components to apply.

For an example of using .NET logic to activate a device and for a presentation component code example that excludes items with no layout details for the context device, see the manual *Presentation Component API Cookbook.*

1.1.2 Rendering Properties

Renderings can generate different output based on rendering properties that you can define at design time and at runtime. You can use rendering properties to reuse character sequences and items in multiple renderings and to share character sequences and items between multiple managed websites.

You can pass a data source item to a rendering. Most renderings retrieve data from the rendering's data source item. You can pass additional items to renderings using additional rendering properties. You can pass the same items to any number of renderings used in any number of managed websites.

For example, multiple managed websites share a common footer appearance, but that footer displays different data for the different sites.

You can use rendering properties to meet these requirements. Create a data template to represent the data in the footer, and create an item based on this data template for each of the managed websites. Pass the appropriate footer item as the data source to the footer rendering for each managed website.

Multiple managed websites reuse the rendering, and multiple web pages within each site can reuse the footer data. You can reuse the footer data item in other renderings, such as the footer rendering for another device.

For more information about rendering properties, see the manual *Presentation Component Reference*.

1.1.3 Conditional Rendering

Conditional rendering applies logic to control renderings at runtime. You can use conditional rendering to control which renderings the layout engine invokes and to apply rendering properties to dynamically reuse and share renderings, items, and character sequences between multiple web pages and managed websites.

For more information about rendering properties, see the section *Rendering Properties*.

For more information about conditional rendering, see the manual *Rules Engine Cookbook*.

Note
If multiple managed websites share data but not markup, layout details can specify a layout containing minimal markup with one or more placeholders, using conditional rendering to populate presentation component hierarchy dynamically.

For example, each page in a solution involving multiple managed websites displays the same message starting 24 hours before the next scheduled event.

You can use conditional rendering to meet these requirements. Configure all of the managed websites to reuse a rendering that presents information about an event. Add a global conditional rendering condition containing the logic to determine the item representing the event to occur within the next 24 hours. Use a conditional rendering action to set the data source of the rendering to the event item. If no such event occurs, set the data source of the rendering to a default item indicating no event in the next 24 hours, or configure the presentation component to generate no output under this condition.

1.1.4 Cascading Style Sheets

CSS files allow the same markup to appear with visually different styling. You can insert references to Cascading Style Sheet (CSS) into markup streams dynamically to present data differently under different conditions.

For example, multiple managed websites share the same data and presentation components, but use different CSS files for presentation.

You can use Cascading Style Sheets to meet these requirements. Configure each of the managed websites to share the same home item. Implement a rendering in the HTML `<head>` element of all pages to include a link to the appropriate CSS file.

The layout engine serves the same HTML for all managed websites, but applies different Cascading Style Sheets.

1.1.5 Inherit Field Values

Presentation components can reuse field values from an ancestor of the context item.

For example, every page on a website can have a logo. If the CMS user does not specify a logo for a page, then the page logo is that of the nearest ancestor of the page for which the user does specify a logo.

You can inherit field values to meet these requirements. Configure all data templates for pages to include a field named Logo of type Image. In the home item for the managed website, populate the Logo field. Create a logo rendering that uses the field value in the context item if that field contains a value, or the nearest ancestor of the context item that contains a value for that field.

The presentation component outputs an HTML `<img>` element using the image referenced in the Logo field in the context item if that field contains a value. Otherwise, the presentation component outputs an

HTML `<img>` element using the image referenced in the Logo field of the nearest ancestor of the context item that contains a value for that field.

Note
In addition to images and text, developers often use field inheritance to allow CMS users to select style sheets, and to control other aspects of presentation.

1.1.6 Reuse and Share with Really Simple Syndication (RSS)

You can use Really Simple Syndication (RSS) to reuse and share data.

Note
This document does not describe RSS.

Note
You can implement presentation components that access external RSS sources to reuse and share data.

For more information about RSS, see http://en.wikipedia.org/wiki/RSS_(file_format).

For more information about Sitecore RSS features, see the manual *Presentation Component Cookbook*.

1.2 Sharing and Reusing Items and Field Values

You can use the following techniques to share and reuse field values in Sitecore items.

Note
You can also use rendering properties to share and reuse items. For more information about rendering properties, see the section *Rendering Properties*.

1.2.1 Aliases

Aliases provide additional URLs for items. You can use aliases to share items between multiple managed websites.

Note
Aliases apply to all managed websites. When the layout engine processes an HTTP request that matches an alias, it sets the context item to the shared item for all managed websites. The aliased item can exist outside of any logical site definition, and all managed sites share that item.

For example, multiple managed websites with different hostnames share a single list of Frequently Asked Questions (FAQ).

You can use an alias to meet these requirements. Configure a single item to represent the shared FAQ page. Configure each child of the FAQ item to represent a FAQ entry, with fields to store a question and its answer. In the FAQ item, configure the alias `faq`.

For all of the managed websites, the layout engine sets the context item to the shared FAQ item when it interprets the URL path `/faq.aspx`.

Note
To prevent questions from appearing on the FAQ for specific sites, you can add a selection field to the data template for questions and answers. If the user does not select any sites in this field, then the question and answer apply to all sites. If the user selects one or more sites, then the question and answer applies only to those sites. Additionally, the FAQ presentation rendering can include FAQ entries stored within each managed website, or elsewhere on the system.

Note
For information about presenting the shared data differently for different managed websites, see the section Sharing and Reusing Data with Presentation Components.

For more information about aliases, see the manual *Content Reference*.

For more information about the context item, see the manual *Presentation Component Reference*.

1.2.2 Selection Fields

Selection fields in data templates allow the CMS user to select one or more Sitecore items when editing an item. You can add any number of a selection fields to a data template, and configure presentation components to retrieve data from the selected items.

You can use the following field types in data templates to allow the CMS user to select a single item:

- Droplink
- Droptree

- Grouped Droplink

You can use the following field types in data templates to allow the CMS user to select zero or more items:

- Checklist
- Multilist
- Treelist
- TreelistEx

Note
You can configure data validation for a selection field to require that the user select a specific number of items.

For example, multiple managed websites share a common content tree. Each site uses a unique set of presentation components. One site provides a prototype from which all others derive both information architecture and the data to populate that information architecture. Each of the derived sites may contain items that do not appear in the prototype site. The prototype site may contain items that do not appear on the derived sites. Derived sites use field values from the prototype site, but CMS users can override field values in the prototype in each derived sites.

You can meet these requirements with a selection field. In each data template, create a Droptree field to allow the user to select an item from the prototype site. In the prototype site, this field is always blank. In presentation components, if an item does not contain a value for a field, and the item specified by the Droptree field contains a value for that field, then use the value from that field of that item.

For more information about data validation, see the manual *Client Configuration Cookbook*.

Item Selection Data Templates

Item selection data templates function as symbolic links from one item to another. You can use selection data templates to reuse an item at multiple locations in the content tree, and to share items between multiple managed websites.

An item selection data template contains a selection field that requires the user to select a single item. When processing an item based on an item selection data template, presentation components retrieve data from the selected item. For more information about selection fields, see the section Selection Fields.

Note
A CMS user could use the Rich Text Editor, rendering properties, or other features to link to an item based on an item selection data template. Account for these conditions in your code. You can configure an `httpRequestBegin` pipeline processor to handle HTTP requests for items based on item selection data templates. The processor can update the context item to the selected item, or log an error and redirect to that item. Alternatively, you can define layout details for the items based on the item selection data template, where those presentation components retrieve some data from the selection item, and other data from the item(s) selected in that field.

Note
Each item has a single URL, regardless of the number of item selection items that with that item selected. For more information about enabling multiple URLs for an item, see the section Aliases.

Note
Unlike rendering properties, which are only visible to users with appropriate CMS access rights, any user that can update an item can update all selection fields in that item. You can use field security to refine who can update selection fields.

For more information about data validation, see the manual *Client Configuration Cookbook*.

For example, a rendering generates navigation links by iterating the content tree. Certain items must appear in multiple locations within the navigation.

You can use an item selection data template to meet these requirements. Create the item selection data template with a field of type Droptree. Create item based on the item selection data template where links must appear in the content tree. In each item based on the item selection data template, select the item to appear at that location. Update the navigation rendering to link to the selected item when processing items based on the item selection data template.

1.2.3 Publishing Targets

Publishing targets represent databases to which Sitecore can publish. You can use publishing targets to scale the content delivery environment, or to segregate published data for different Sitecore instances.

Unless you select publishing targets in publishing restrictions, Sitecore publishes all data to all publishing targets, making it available for reuse and sharing in all content delivery environments.

Important
Sitecore does not publish the descendants of a common ancestor to publishing targets to which it does not publish that ancestor.

For more information about publishing targets, see the manual *Scaling Guide*.

1.2.4 Proxies

Proxies cause items or branches of items to appear at multiple locations in the content tree. You can use proxies to reuse an item or a branch of items in multiple locations in the content tree, or to share a branch of items to multiple managed websites.

Warning
Sitecore 6.4 deprecates proxies and in later versions of Sitecore proxies will become obsolete.

Warning
Enabling proxies has a negative impact on the overall performance of a Sitecore installation. If you do not need to use proxies in a database, do not enable proxies for that database.

The negative performance impact associated with using proxies increases significantly with the following two factors:

- The number of proxy definitions in an installation.
- The size of the content tree referenced by a proxy definition.

In order to avoid performance degradation, Sitecore recommends that you:

- Avoid allowing CMS business users to create proxies.
- Avoid configuring proxies to large branches of items.
- Avoid allowing the number of proxy definitions to grow unbounded in production environments.

- Carefully test the number of expected proxies in a test environment before releasing a solution into production.

Important
Sitecore recommends that you disable proxies in publishing target databases and configure the Master database to publish virtual items.

Note
Depending on how you use them, proxies can result in multiple URLs for a single item, which can adversely affect search engine ranking. You can share and reuse data without proxies, such as to meet the requirements for the example described in the section Devices and the section Item Selection Data Templates.

Note
For each item, each device triggers a single layout. You can use additional devices to apply different layouts to a single item. You can use rendering properties, conditional rendering, and other techniques to format reused and shared data for different circumstances. For information about formatting data differently in different contexts, see the section Sharing and Reusing Data with Presentation Components.

For more information about proxies, see the manual *Content Reference*.

1.2.5 Clones

Clones share similarities with proxies and copies of items. Copying an item is a very similar operation to cloning, although after creation a cloned item behaves differently to a copy. Cloning has more flexibility than copying without the performance overheads of proxies.

A clone is an item with an 'intelligent' standard values link that points to another item called the origin. The origin is the original item that the clone is based on. The clone has the same data template as the origin. Sitecore uses the __Source field in the Advanced section of the standard template to specify the source of the clone. An item with a valid URI in the __Source field is a valid clone. Cloning logic applies when retrieving field values for a cloned item.

Cloned items do not inherit the following fields from their origin:

- Lock
- Workflow
- Workflow State
- Updated
- Created
- Updated By
- Created By
- Source
- Revision

Cloning Items

When you create a clone of an item, Sitecore creates clones for each of the descendants of the item. It does not store any field values in the clones but takes all values from the origins. However when a field is changed, the clone saves the change, creating a local copy of the value.

To clone an item, select an appropriate item in the content tree. In the Content Editor ribbon, Configure tab, Clone group use the Clone and Unclone controls.

To clone multiple items, for example multiple items contained in item buckets, first use Sitecore search to find the items you want to clone. Then in the search drop down, select *Search Operations* and click *Clone Results To*. In the *Clone Item* dialog box, you can choose a location in the content tree to clone your search results or use search to find a suitable location.

Unclone

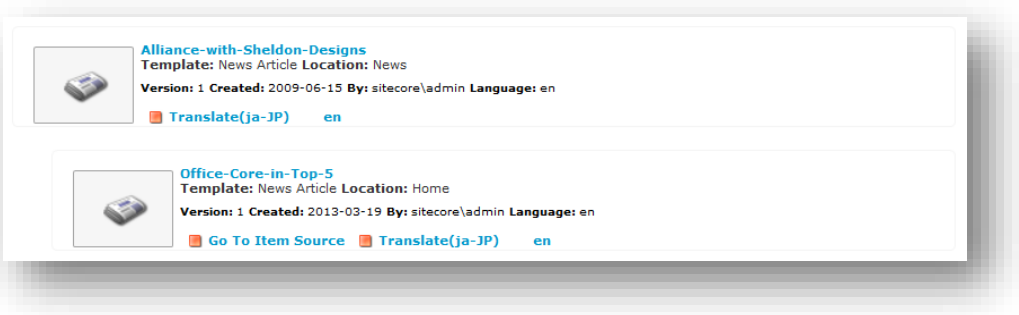
The unclone operation changes the clone into an item. It copies all field values from the origin to the clone and removes the __Source link, effectively creating two independent items.

Notifications

When there are changes to the origin, such as the creation of a new item, field or version, a notification is pushed to the UI allowing the end-user to accept or reject the changes. Each notification is stored as a record in a Notifications table in the Sitecore database containing the item to which the notification applies.

If you create multiple clones using search operations, to store in item buckets, the *Name* field at the bottom of the *Clone Item* dialog box displays the path to the source item.

In the search results, a cloned item has the **Go To Item Source** quick action assigned to it by default. This is a shortcut to the original item.



Deleting Clones

When you delete an item that has clones, the clones become normal content items. This is effectively the same as the unclone operation.

Note
Clones are normal Sitecore items. Unlike proxies, there are no special objects or data structures behind them. They can use workflows, publishing restrictions, reminders, layout definitions like any other Sitecore items.

Note
It is not possible to clone a data template or a data template section.

For more information on item buckets and cloning, see section 4.6 Cloning Items, in the Content Authors Cookbook.

1.2.6 Wildcard Items

Wildcard items map all URL paths that would otherwise map to children of an item to a common child of that item. You can use wildcard items to reuse and share any data, including Sitecore items.

If the name of an item is an asterisk ("*"), then that item is a wildcard. If the directory part of the URL path maps to an item, but that item does not have a child with a name matching the file name part of the URL path, and a wildcard item exists at that location, Sitecore sets the context item to that wildcard item.

For example, multiple managed websites share data in an external Product Information Management (PIM) system. The URL of any product on any managed site is `/product/<ProductID>.aspx`.

You can meet these requirements with wildcard items. Create the `/product` item beneath each website. Beneath each `/product` item, create an item named asterisk ("*"). Configure layout details for the wildcard items to include a rendering that retrieves data from the PIM system using the product identifier in the URL.

The layout engine sets the context item to the wildcard item, and invokes the presentation components specified in the layout details of that item as it would for any other item.

Note
You can use an alias to avoid creating the `/product` item under each website. For more information about aliases, see the section *Aliases*.

Note
Sitecore wildcard items support nesting. For example, if no other item matches the URL `/products/category/product.aspx`, then Sitecore processes the `/Sitecore/Content/Home/Products/*/*` item.

1.2.7 Web Services, AJAX, and other Internet Interfaces

Interfaces such Web Services expose data, APIs, and other services over network connections. You can use web services, AJAX, and other Internet technologies to reuse and share data in Sitecore with other applications, including AJAX in the client.

Important
Always account for the security context in which a web service runs.

Note
Sitecore exposes default web services at the URL `/sitecore/shell/webservice/service.asmx`. You can implement a custom web service to expose any Sitecore data or API.

Note
You do not always need web services to syndicate data over the Internet. The client application can request the URL of an item on the Sitecore server, using a query string parameter to activate a specific device. The layout for that device could format data as required for the external application. For more information about devices, see the section Devices.

1.3 Sharing and Reusing Character Sequences

You can use the following techniques to share and reuse sequences of characters that you do not store in field values of Sitecore items.

Note
You can also use rendering properties to reuse character sequences. For more information about rendering properties, see the section Rendering Properties.

1.3.1 Snippets

Snippets let CMS users insert predefined sequences of characters into Rich Text Editor field values. You can use snippets to let CMS users easily insert reusable phrases, tokens, and other character sequences into Rich Text Editor field values.

Important
If you update a snippet, Sitecore does not update all field values that contain that snippet.

For example, an organization frequently enters its slogan into the value of a Rich Text Editor field.

To meet these requirements, you can configure a snippet to allow the user to insert the slogan with a few mouse clicks rather than by typing the slogan on the keyboard each time.

Sitecore inserts the snippet into the field value when the user selects that snippet from a drop-down menu in the Rich Text Editor.

For more information about snippets, see the manual *Client Configuration Cookbook*.

1.3.2 Dictionary Translation

The Sitecore dictionary translates specific character sequences into different languages using items to store the terms and translations. You can use the dictionary to translate phrases and sequences of characters to reusable values for each language.

You can use the `Sitecore.Globalization.Translate.Text()` method to translate a token. If the value of the Key field of an dictionary entry item matches the character sequence passed to the first parameter of the `Sitecore.Globalization.Translate.Text()` method, then this method returns the value of the Phrase field in that dictionary entry. Otherwise, `Sitecore.Globalization.Translate.Text()` returns the value passed to the method.

Note
The `Sitecore.Globalization.Translate.Text()` method is case-sensitive.

For an example that uses dictionary translation to reuse data, see the section Token Substitution.

How to Create a Dictionary Entry

To create a dictionary entry:

1. In the Content Editor, select the `/Sitecore/System/Dictionary` item.

   If the dictionary will have a small number of entries over the life of the project, you can insert dictionary entries directly beneath the `/Sitecore/System/Dictionary` item. Otherwise, insert dictionary folders using the `System/Dictionaries/Dictionary Folder` data

template. For example, create a dictionary folder named "a" to contain terms that begin with the letter a. Create nested dictionary folders to prevent any folder containing hundreds of items.

2. In the Content Editor, with the /Sitecore/System/Dictionary item or the dictionary folder item selected, insert a dictionary entry item using the System/Dictionaries/Dictionary Entry data template.

   Give the dictionary entry item a name that will help CMS users identify the term when they maintain its value.

   Note
   If CMS users cannot access the /Sitecore/System branch in order to access the /Sitecore/System/Dictionary folder, then you can use a proxy to cause this branch to appear elsewhere, such as /Sitecore/Content/Dictionary. For more information about proxies, see the section *Proxies*.

3. In the new dictionary entry item, in the **Data** section, in the **Key** field, enter the token to translate.
4. In the new dictionary entry item, in the **Data** section, in the **Phrase** field, enter the value for the current language.

Important
For each dictionary entry item, for each language, enter a value for the Phrase field in the Data section.

Important
Remember to publish dictionary folders and dictionary entries.

1.3.3 Token Substitution

Token substitution replaces character sequences in data with alternate values. You can use token substitution in presentation components to reuse data among pages and possibly languages, and to share this data between multiple managed websites.

Important
Choose tokens that will not otherwise appear in field values.

Important
Do not replace tokens for fields that support inline editing if the user is inline editing in the Page Editor.

Note
You can replace tokens with values stored in fields of Sitecore items, configuration files, .NET resource files, or other information stores.

For example, an organization's slogan changes periodically. That slogan must appear as plain text in Rich Text Editor fields.

You can meet these requirements with token substitution. Insert a token such as $slogan in the field value, and replace that token with the slogan at runtime using the renderField pipeline as described in the following section. Because a slogan is a single value, you can use dictionary translation for the slogan.

How to Implement Token Substitution: the renderField Pipeline

The layout engine invokes the renderField pipeline to process field values, such as converting internal representation to HTML and expanding dynamic links. You can implement token substation by adding a

processor to the `renderField` based on the following example that meets the requirements defined in the previous section.

1. Add a class based on the following example to the Visual Studio project:

```
namespace Sitecore.Sharedsource.Pipelines.RenderField
{
  using System;

  public class ApplySlogan
  {
    public string Token
    {
      get;
      set;
    }

    public void Process(Sitecore.Pipelines.RenderField.RenderFieldArgs args)
    {
      if (Sitecore.Context.PageMode.IsPageEditorEditing)
      {
        return;
      }

      Sitecore.Diagnostics.Assert.IsNotNullOrEmpty(this.Token, "Token");
      string slogan = Sitecore.Globalization.Translate.Text(this.Token);
      Sitecore.Diagnostics.Assert.IsNotNullOrEmpty(slogan, "slogan");

      if (!String.IsNullOrEmpty(args.Result.FirstPart))
      {
        args.Result.FirstPart = args.Result.FirstPart.Replace(
          this.Token,
          slogan);
      }

      if (!String.IsNullOrEmpty(args.Result.LastPart))
      {
        args.Result.LastPart = args.Result.LastPart.Replace(
          this.Token,
          slogan);
      }
    }
  }
}
```

2. Add the processor to the `renderField` pipeline immediately before the `Sitecore.Pipelines.RenderField.AddBeforeAndAfterValues` processor:

```
<renderField>
...
  <processor type="Sitecore.Sharedsource.Pipelines.RenderField.ApplySlogan, assembly">
    <token>$slogan</token>
  </processor>
  <processor
    type="Sitecore.Pipelines.RenderField.AddBeforeAndAfterValues, Sitecore.Kernel" />
...
</renderField>
```

For this example, create a dictionary entry as described in the section How to Create a Dictionary Entry. In the dictionary entry item, in the Data section, in the Key field, enter the token `$slogan`. In the Data section, in the Phrase field, enter the current slogan. For more information about dictionary translation, see the section Dictionary Translation.

Unless the user is inline editing in the Page Editor, this processor replaces the token `$slogan` in field values rendered using the `renderField` pipeline the value from the corresponding dictionary entry.

Note
You can create individual data templates for different slogans, and store the slogan in the standard values of that data template.

Note
To reduce the need for CMS users to memorize tokens, you can create snippets that contain tokens. For example, you could define a snippet to insert the token `$slogan`. For more information about snippets, see the section Snippets.