Sitecore CMS 6.4 or later

Client Configuration Cookbook

Features, Tips and Techniques for CMS Architects and Developers

Table of Contents

Chapter 1

Introduction

This cookbook provides information, tips, and techniques for CMS architects and developers to optimize Sitecore client user interface usability.

For client hardware requirements, see the *Sitecore Installation Guide* on SDN.

For information about configuring Internet Explorer, see the manual *Internet Explorer Configuration Reference*.

This document contains the following chapters:

- Chapter 1 — Introduction
- Chapter 2 — Common Procedures
- Chapter 3 — Data Templates and Items
- Chapter 4 — Data Validation
- Chapter 5 — The Page Editor
- Chapter 6 — Security Based Configuration Features
- Chapter 7 — Sitecore Client RSS Feeds

Chapter 2

Common Procedures

This chapter provides instructions for common procedures required by other procedures described in this and other documents.

This chapter contains the following section:

- Common Procedures

2.1 Common Procedures

This section provides instructions for common procedures required by other procedures described in this and other documents.

2.1.1 How to Select a Database in the Sitecore Desktop

To select a database in the Sitecore desktop:

1. In the browser, access the Sitecore login page, such as http://localhost/sitecore.
2. In the Sitecore login page, enter your Sitecore username and password.
3. Click **Options**, and then double-click **Desktop**. The Sitecore desktop appears in the browser.
4. In the Sitecore desktop, in the lower right corner, click the database icon, and then click the database name. The Sitecore desktop refreshes, closing any open applications. If you open the **Content Editor** or other applications within the Sitecore desktop, they will access the database that you selected until you log out or select a different database.
5. In the Sitecore desktop in the lower left corner, click **Sitecore**, and then click **Content Editor**, the **Content Editor** appears, allowing you to navigate the selected database.

Important
To reduce the potential for making changes to a database unintentionally, always select the *Master* database after working with one of the other databases.

Note
Unless otherwise specified, all procedures in this and other Sitecore documentation assume that you have selected the *Master* database.

2.1.2 How to Show or Hide the Standard Fields

To show or hide the standard fields:

1. In the **Template Manager** or the **Content Editor**, click the **View** tab.
2. On the **View** tab, in the **View** group, enable or disable **Standard Fields**.

Note
The visibility of the standard fields can affect Sitecore client performance.

2.1.3 How to Show or Hide Raw Values

You can investigate the text value of a field by viewing its raw value. For example, you may view the raw value of a field to determine the attributes that Sitecore stores for an element.

Note
Whenever possible, use constructs that abstract field values rather than accessing raw field values directly.

To show or hide raw values:

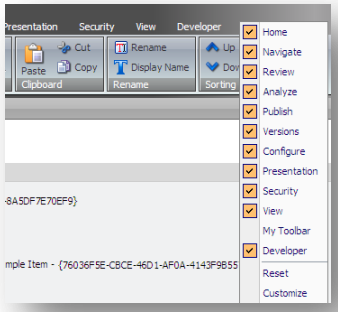
1. In the **Template Manager** or the **Content Editor**, click the **View** tab.
2. On the **View** tab, in the **View** group, enable or disable **Raw Values**.

2.1.4 How to Show or Hide the Developer Tab

The Developer tab provides convenience features for developers working with Sitecore solutions.

To show or hide the Developer tab:

- In the **Content Editor**, right-click the ribbon, and then enable or disable **Developer**.



2.1.5 How to Copy the ID or Path of an Item to the Windows Clipboard

To copy the path to an item to the Windows clipboard:

1. In the **Template Manager** or the **Content Editor**, select the item, and then click the **Content** tab.
2. In the **Quick Info** section, click and drag the mouse across the value of the **ID** or **Item Path**. To copy the path to the Windows clipboard, press CTRL-C, or right-click the selected text, and then select **Copy**.

Tip
You can also copy the ID or path to an item by using the **ID** and **Path** commands in the **Show** group on the **Developer** tab. For more information about the **Developer** tab, see the section *How to Show or Hide the Developer Tab*.

2.1.6 How to Enter a Class Signature

A class signature identifies a class in a .NET assembly (`.dll` file).

To enter a class signature:

1. Enter the following prototype:

```
Namespace.Class, Assembly
```

2. Replace `Namespace` with the namespace containing the class.
3. Replace `Class` with the name of the class.
4. Replace `Assembly` with the name of the assembly containing the class, without the `.dll` extension.

Chapter 3

Data Templates and Items

This chapter provides tips and describes techniques for configuring data templates and items, including controlling item, section, and field appearance, custom item editors, and insert options.

This chapter contains the following sections:

- Item Appearance
- Data Template Sections
- Data Template Fields
- Custom User Interfaces in Data Templates
- Insert Options

3.1 Item Appearance

You can use the following features to control the appearance of items in the content tree. A field in the standard template defines the icon for each item.

3.1.1 Icons

Sitecore user interfaces, including the content tree in the Content Editor, display an icon next to each item.

Developers use icons:

- To differentiate specific items visually.
- To differentiate items based on specific templates visually.

If an item does not specify an icon, the content tree displays the icon specified in data template associated with the item. If that data template does not specify an icon, then the content tree displays the icon specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name DefaultIcon`.

Note
Sitecore supports theming. Sitecore interprets relative icon paths as relative to the `/sitecore/shell/themes/standard` directory. For example, the value `applications/16x16/star_yellow.png` is equivalent to `/sitecore/shell/themes/standard/applications/16x16/star_yellow.png`.

Note
An icon can be an arbitrary URL of an image.

How to Set the Icon for an Item

To set the icon for an item:

1. In the **Content Editor** or the **Template Manager**, select the item.
2. Click the **Configure** tab and in the **Appearance** group, click **Icon**.
3. In the drop down menu, select an icon or click **More Icons** and then in the **Icon** dialog box select an icon.

Note
The **Icon** dialog box may load slowly when presenting a large number of images in a single directory.

Tip
As a shortcut to access the Icon dialog box, in the editing pane, click the **Content** tab, and then click the icon in the item title bar.

Important
Rather than setting the icon for individual items, set the icon for each data template. The icon of a data template is the default icon for items based on that template, including the standard values item.

How to Set the Default Icon for All Items Based on a Data Template

To set the default icon for all items based on a data template:

1. In the **Template Manager** or the **Content Editor**, select the data template definition item.
2. Set the icon in the data template definition item.

   For more information about setting the icon, see the section *How to Set the Icon for an Item*.

Important
Set the icon in the definition item for each data template, not in the standard values item for the data template. If you only set the icon in the standard values item for the data template, the icon does not apply to the data template itself; the icon shown for the data template in the content tree remains the default icon.

How to Change the Default Icon for All Items

To set the default icon for all items for which the item, its data template, and the standard values item associated with its data template do not define an icon, in the `web.config` file, in the `/configuration/sitecore/settings/setting` with the name `DefaultIcon`, set the `value` attribute to the default icon path.

How to Configure the Icon Selection Menu

To configure the icons that appear in the icon selection menu, edit the file `/App_Config/Icons.config`. Each collection element defines a row, and contains a pipe-separated list of absolute or relative paths to icon files.

3.1.2 Hidden Items

Hidden items do not appear in the content tree for users who do not have permission to show hidden items, or have not chosen to show hidden items.

How to Show or Hide Hidden Items

To show hidden items, the user must be an administrator, or a member of either the Sitecore Client Developing role or the Sitecore Client Maintaining role.

To show hidden items:

1. In the **Content Editor** or the **Template Manager**, click the **View** tab.
2. In the **View** group, enable or disable **Hidden Items**.

How to Show or Hide an Item

To hide or show an item, the user must be an administrator, or a member of the Sitecore Client Configuring role.

To show or hide an item:

1. In the **Content Editor** or the **Template Manager**, select the item.
2. Click the **Configure** tab and in the **Attributes** group, click **Hide Item** to hide the item, or click **Make the Item Visible** to show the item.

3.1.3 Protected Items

No user can edit protected items through Sitecore user interfaces.

How to Protect or Unprotect an Item

To protect or unprotect an item, the user must be an administrator or a member of the **Sitecore Client Configuring** role.

To protect or unprotect an item:

1. In the **Content Editor** or the **Template Manager**, select the item.
2. On the **Configure** tab, in the **Attributes** group, click **Protect Item** to protect the item, or click **Unprotect Item** to unprotect the item.

3.1.4 Item Styles

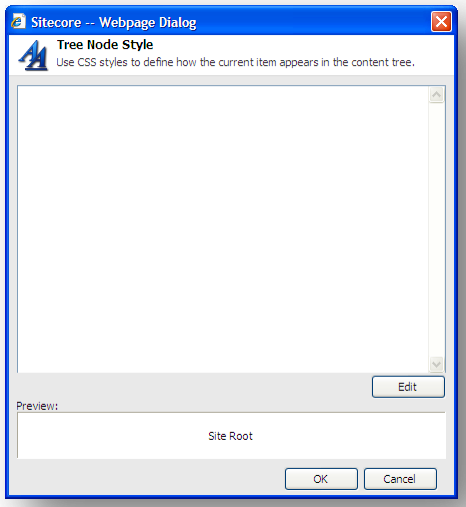
Item styles control the appearance of the names displayed for each item in the content tree. Developers use item styles to highlight and differentiate visually specific items or items based on specific templates. For example, certain system items appear in green text by default, and proxy items to appear grayed out.

To style individual items, developers apply item styles to items. More commonly, developers apply items styles to data templates to style all items based on those templates.

How to Configure the Style of an Item Name in the Content Tree

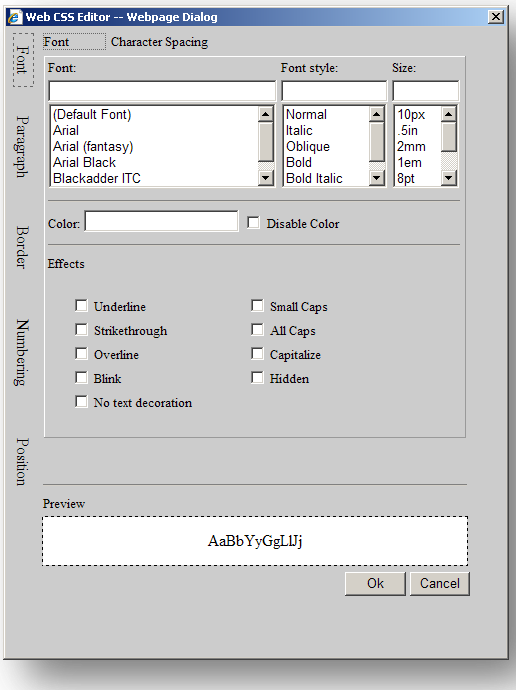
You can use the tree node style to configure the styling of item names in the content tree. To configure the tree node style for item names:

1. In the **Template Manager** or the **Content Editor**, select the standard values item or the individual item.
2. Click the **Configure** tab and in the **Appearance** group, click **Tree Node Style**.



3. In the **Tree Node Style** dialog box, enter CSS styles, or click **Edit** to use the **Web CSS Editor** dialog.

For example, enter `COLOR:red;` to set the color of the font to red.



Note
Sitecore uses `COLOR:green` for system items and `COLOR:gray` for proxy shadow items. Avoid using these tree node styles for other purposes.

3.1.5 Display Name

If defined, the display name of an item appears in user interfaces such as the content tree instead of the name of the item.

Developers use display names:

- To provide helpful item names in the content tree without affecting the default URL for content items.
- To allow characters not allowed in item names to appear in the content tree.

Warning
Use display names in limited cases, such as where you must display characters not allowed in item names.

For information about using display names in URLs, see the Dynamic Links guide at http://sdn.sitecore.net/Reference/Sitecore%206/Dynamic%20Links.aspx

How to Set the Display Name for an Item

To set the display name for an item:

1. In the **Content Editor** or the **Template Manager**, select the item.
2. On the **Home** tab, in the **Rename** group, click **Display Name**.

3. In the dialog box that appears, enter the display name for the item.

3.1.6 Context-Sensitive Help

Editorial interfaces can display helpful information about the selected item and its data template.

Developers use context sensitive help:

- To provide useful information to users working with items based on a specific data template.
- To provide useful information to users working with a specific item.

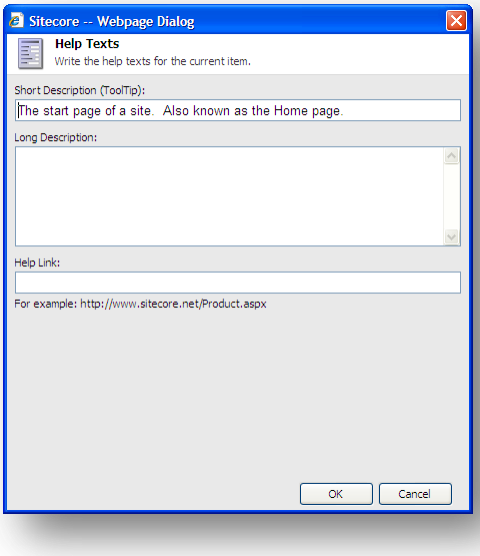
The item title bar displays the short description defined in the help properties of the item or its data template. When you move the mouse over an item in the content tree, Sitecore displays the long description of the item or its template as a tooltip.

If the developer has not defined any help properties for the item, Sitecore uses the help properties defined in the item's data template.

How to Set Item Context-Sensitive Help

To set context-sensitive help for a data template or an individual item:

1. In the **Template Manager** or the **Content Editor**, select the standard values item or the individual item.
2. On the **Configure** tab in the **Appearance** group, click **Help** and the **Help Texts** dialog box appears.



3. In the **Short Description** field, enter a short description of the type of item or the individual item.
4. In the **Long Description** field, enter a long description of the type of item or the individual item.

3.2 Data Template Sections

Each data template contains one or more sections, and each section contains one or more fields. Data template sections allow developers to organize fields into related groups. Organizing fields into sections avoids the appearance of monolithic data entry forms, and allows users to collapse sections of the form that they are not currently using. Developers also use data template sections to contain groups of fields that you can reuse in multiple data templates.

Data templates with large number of sections, and sections with large numbers of fields, can reduce usability and performance. To improve performance, it is especially important to avoid creating data templates with large numbers of Rich Text Editor (RTE) fields. Instead, developers use other field types, or represent the data using a hierarchy of items associated with different data templates.

Logical sorting of data template sections and fields helps users locate the fields they wish to update. Consider sorting sections and fields in the same order that presentation components render field values in page views. Alternatively, place the sections and fields that are most important or updated most frequently towards the top of the data template.

Using appropriate data template section and field names helps users locate the fields they wish to update. For example, if the data template for news articles defines only a few fields such as Author and Date, consider placing those fields in a section named News, and do not use this section name for any fields that are not specific to news articles. Overly general section names such as Data do not provide the same contextual assistance in templates that contain numerous fields.

If a data template and any of its base templates define sections with a common name, the Content Editor renders all the fields in those sections as a single visual section. The sort order property of sections and fields controls their order relative to other sections and fields in the template.

3.2.1 Data Template Section Icons

Data template section headers in the Content Editor display the icons associated with each data template section. Developers specify icons for data template sections to help users locate the fields they want to edit.

Note
If a data template section definition item does not specify an icon, then Sitecore uses the icon specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with name `DefaultIcon`.

3.2.2 How to Set a Data Template Section Icon

To set the icon for a data template section:

1. In the **Template Manager** or the **Content Editor**, select the data template section definition item.
2. Set the icon for the data template section definition item.

   For information about setting the icon for an item, see the section *How to Set the Icon for an Item*

3.2.3 Data Template Section Sort Order

You can sort data template sections using Template Builder, or by setting the sort order property of a data template section definition item.

Note
To control the order of sections when a data template and its base templates define different data template sections, set the sort order property of data template section definition items.

How to Sort Data Template Sections

To sort the data template sections in a data template:

1. In the **Template Manger** or the **Content Editor**, select the template and then click the **Builder Options** tab.
2. In the **Template Builder**, click the section name to select that section.
3. In the **Sorting** group, click **Up**, **Down**, **First**, and **Last** to sort the section relative to the other sections in the data template.

How to Set the Sort Order Property of a Data Template Section Definition Item

To set the sort order property of a data template section definition item:

1. In the **Template Manager** or the **Content Editor**, select the data template section definition item.
2. Show the standard fields.
3. In the **Appearance** group, in the **Sortorder** field, enter a numerical value.
4. In the **Template Manager** or the **Content Editor**, hide the standard fields.

For more information about showing and hiding the standard fields, see the section *How to Show or Hide the Standard Fields*.

3.3 Data Template Fields

You can use the features described in this section to optimize the usability of data template fields.

3.3.1 Data Template Field Headers and Context-Sensitive Help

The Content Editor displays field headers above each field. Developers use field headers to provide instructions and contextual information to users working with a field value.

Each field header consists of a number of elements, including:

- The field title, which defaults to the field name.
- The token `[standard value]`, if the field contains its standard value.
- The token `[shared]`, if the field contains a value shared to all versions of the item in all languages.
- The token `[unversioned]`, if the field contains a value that is not versioned, but may differ by language.
- Field editing controls appropriate for the field type.

The long description of the field appears when the user moves the mouse over the field title.

For fields that define a help link, the title and short description in the field header link to the specified URL.

How to Set the Title for a Data Template Field

To set the title for a data template field, causing the field label in the Content Editor to differ from the name of the field definition item:

1. In the **Template Manager** or the **Content Editor**, select the data template field definition item.
2. In the **Data** section, in the **Title** field, enter the title of the field.

How to Set Context-Sensitive Help for a Data Template Field

To set context-sensitive help for a data template field:

1. In the **Template Manager** or the **Content Editor**, select the data template field definition item.
2. Click the **Configure** tab and in the **Appearance** group, click **Help** and the **Help Texts** dialog box appears.
3. In the **Help Texts** dialog box, in the **Short Description** field, enter a short description of the field.
4. In the **Long Description** field, enter a long description of the field.
5. In the **Help Link** field, enter the URL of a resource containing helpful information about the field.

3.3.2 How to Style a Data Template Field

To style a data template field:

1. In the **Template Manager** or the **Content Editor**, select the data template field definition item.
2. Click the **Configure** tab and in the **Appearance** group, click **Tree Node Style** and the **Tree Node Style** dialog box appears.
3. In the **Tree Node Style** dialog box, enter CSS styles, or click the Edit button to use a CSS style wizard.

   For example, enter `HEIGHT:600px;` to set the height of a Rich Text Editor, or `FONT-WEIGHT:bold;` to style the text in a single-line text field.

3.3.3 Data Template Field Sort Order

Explicitly set the sort order property of data template field definition items to control the order of fields when a data template and its base templates define the same data template sections.

How to Sort Data Template Fields

To sort data template fields:

1. In the **Template Manager** or the **Content Editor**, select the data template definition item.
2. Click the **Builder** tab and the **Template Builder** appears.
3. Click in a field name to select that field.
4. In the **Sorting** group, click **Up**, **Down**, **First**, and **Last** to sort the field relative to the other fields in the section.

How to Set the Sort Order Property of a Data Template Field Definition Item

To set the sort order property of a data template field:

1. In the **Template Manager** or the **Content Editor**, edit the data template field definition item.
2. Show the standard fields.
3. In the **Appearance** section, in the **Sortorder** field, enter a numerical value.
4. Hide the standard fields.

For more information about showing and hiding the standard fields, see the section *How to Show or Hide the Standard Fields*.

3.3.4 Rich Text Editor (RTE) Configuration

Rich Text Editor (RTE) profiles control the features available in the Rich Text Editor (RTE) fields.

Developers use RTE profiles:

- To remove features from RTE fields.
- To enable default features in RTE fields.
- To expose different features to different users of RTE fields.

- To configure drop-down menus and other options in RTE fields.
- To add custom features to RTE fields.
- To make different features available in different data templates and different RTE fields in a single template.

Developers may reference an RTE profile using the source property of each RTE field definition item. Sitecore stores RTE profiles under `/Sitecore/System/Settings/Html Editor Profiles` in the *Core* database. If the developer does not define the source property for an RTE field, Sitecore applies the `/Sitecore/System/Settings/Html Editor Profiles/Rich Text Default` RTE profile.

Only the definition items that exist in each RTE profile appear in fields using that profile. Access rights assigned to definition items in the RTE profile control access to RTE features for different users.

Tip
For consistency, accessibility, and reusability, separate design from presentation. RTE fields defeat this purpose by merging content with styling, such as CSS styles, and presentation, such as HTML tables and images. In addition to minimizing use of the RTE field type, minimize features available in each RTE field using RTE profiles.

RTE Profile Definition Item Reference

The following table describes items that can appear within an RTE profile definition item.

Path	Function
`/Buttons/HTML View`	Controls whether or not the **HTML** tab appears at the bottom of the RTE.
`/Class Translation`	Maps styling information in content pasted into the pasted into the RTE to corresponding CSS styles.
`/Font Names`	Populates the font names drop-down menu.
`/Font Sizes`	Populates the font sizes drop-down menu.
`/Links`	Populates the Custom Links drop-down menu.
`/Paragraphs`	Populates the elements drop-down menu.
`/Ribbon`	Controls the formatting ribbon in the Page Editor.
`/Snippets`	Populates the snippets feature.
`/Tidy`	This path has been discontinued.
`/Toolbar 1`	Controls options on the primary toolbar.
`/Toolbar 2`	Controls options on the secondary toolbar.
`/Toolbar 3`	Controls options on the tertiary toolbar.
`/WebEdit Buttons`	Controls formatting buttons shown beneath the field when inline editing in the Page Editor.
`/Zoom`	Populates the Zoom drop-down menu.

How to Configure RTE Profiles

To configure RTE Profiles:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the `/Sitecore/System/Settings/Html Editor Profiles` item.
2. In the **Content Editor**, beneath each custom RTE profile definition item, remove options that are unnecessary for all users, and control the visibility of other options using the `item:read` access right.
3. In the Sitecore desktop, select the *Master* database.

   For more information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

Important
Do not edit the default RTE profiles under the `/Sitecore/System/Settings/Html Editor Profiles` item in the *Core* database. Instead, duplicate an existing RTE profile, and configure RTE data template field definition items to use that profile.

Tip
To configure the default RTE profile without editing it or specifying that profile in the Source property of each RTE field definition, duplicate the default RTE profile to create a backup, and then edit the original.

How to Determine the Path to an RTE Profile

To determine the path to an RTE profile:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the RTE profile definition item.
2. In the **Content Editor**, copy the path to the RTE profile definition item to the Windows clipboard.
3. In the Sitecore desktop, select the *Master* database.

For information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

For information about copying an item path to the windows clipboard, see section *How to Copy the ID or Path of an Item to the Windows Clipboard*.

How to Set the Profile for an RTE Template Field Definition Item

To set the RTE profile for an RTE template field definition item:

1. In the **Template Manager** or the **Content Editor**, select the RTE field definition item.
2. In the **Data** group, in the **Source** field, enter the path to the RTE profile.

How to Add Features to an RTE Profile

To add features to an RTE profile:

- Copy features from the `/Sitecore/System/Settings/Html Editor Profiles/Rich Text Full` RTE profile definition.

How to Set the CSS Used by RTE Fields

To set the Cascading Style Sheet (CSS) file used to populate the Apply CSS Class drop-down in all RTE fields:

1. Close the Rich Text Editor.
2. In `web.config`, in the `/configuration/sitecore/settings/setting` element with `name WebStylesheet`, set the `value` attribute to the path to a CSS file relative to the document root of the IIS web site.
3. Open the Rich Text Editor to see the change.

For information about loading different CSS files into the RTE, see http://sdn.sitecore.net/Scrapbook/Dynamically%20loading%20web%20stylesheets%20in%20RTE.aspx.

How to Limit the CSS Styles Visible in RTE Fields

You can use the CSS `import` directive to prevent styles from appearing in the Apply CSS Class drop-down menu in the RTE.

This Apply CSS Class drop-down includes styles defined in the CSS file specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name WebStylesheet`.

Most web sites use CSS styles that are not relevant to content in the RTE. You can use the CSS `import` directive to prevent these styles from appearing in the Apply CSS Class drop-down in the RTE.

For example, place the styles that users can apply to content in the RTE in the file `contentstyles.css`, and set the `WebStylesheet` setting to the path to that file relative to the document root. Place the styles that should not appear in the RTE in the file `sitestyles.css` in the same directory, and reference that file in your layouts. In `sitestyles.css`, import `contentstyles.css` with a CSS `import` directive such as the following:

```
@import url(contentstyles.css);
```

Only styles defined in `contentstyles.css` appear in the RTE, while presentation components can use styles defined in both `sitestyles.css` and `contentstyles.css`.

Warning
Due to various levels of caching, changes to CSS files may not be visible in the browser immediately. If changes to CSS files do not appear, perform the following operations in order until the change appears: reload the Rich Text Editor, clear the browser cache, restart IIS, and change the value of the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name WebStylesheet`.

How to Configure the RTE HTML Element Types Drop-Down Menu

To configure the HTML element types listed in the RTE:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the RTE profile definition item.
2. Under the RTE profile definition item, select `/Paragraphs`.
3. Insert an HTML element definition item using the `/System/Html Editor Profiles/Html Editor List Item` data template.
4. Select the HTML element definition item.

5. In the **Data** section, in the **Header** field, enter the value to display in the elements drop-down.
6. In the new item, in the **Data** section, in the **Value** field, enter the HTML element to insert when the user chooses this element type.
7. In the Sitecore desktop, select the *Master* database.

For more information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

How to Enable Snippets in an RTE Profile

To enable snippets in an RTE profile:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the `/Sitecore/System/Settings/Html Editor Profiles/Rich Text Full/Toolbar 1` item.
2. In the **Content Editor**, copy the `Insert Snippet` item to the corresponding location in the RTE profile.
3. In the Sitecore desktop, select the *Master* database.

How to Add a Snippet to an RTE Profile

To add a snippet to an RTE profile:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the **Snippets** item beneath the relevant RTE profile definition item within the `/Sitecore/System/Settings/Html Editor Profiles` branch.
2. With the Snippets item within the relevant RTE profile definition item selected, insert a snippet definition item using the `/System/Html Editor Profiles/Html Editor Snippet` data template.
3. In the snippet definition item, in the **Data** section, in the **Header** field, enter the text to display in the snippets drop-down list.
4. In the **Value** field, enter markup to insert when the user chooses the snippet.
5. In the Sitecore desktop, select the *Master* database.

How to Control the Markup Inserted by the Enter Key

The `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name HtmlEditor.LineBreak` controls the markup that the Rich Text Editor inserts when a user presses the ENTER key. By default, the `value` of this `setting` is `p`, causing the Rich Text Editor to surround text with paragraph elements ("`<p>...</p>`"). To insert a line break element ("`<br>`") at the end of the line instead of wrapping the text with paragraph elements, change the value of this setting to `br`.

Note
If the `value` of the `/configuration/sitecore/settings/setting` element with `name HtmlEditor.LineBreak` is `p`, you can insert a line break element (`<br>`) by pressing CTRL-ENTER. If the value of this setting is `br`, you can insert paragraph elements (`<p>...</p>`) by pressing CTRL-M.

How to Disable the RTE HTML Tab for Some or All Users

You can remove the HTML tab from the RTE for all CMS users, or use access rights to control which users can access the HTML tab.

To remove or restrict the HTML tab from the RTE:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the RTE editor profile definition item.
2. Delete or restrict read access rights to the `/Buttons/HTML View` item.
3. In the Sitecore desktop, select the *Master* database.

For more information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

3.4 Custom User Interfaces in Data Templates

If Sitecore does not provide the data template user interface elements that you need, you can implement custom user interface in data templates.

3.4.1 The IFrame Field Type

The IFrame data template field type enables developers to cause arbitrary web applications to appear as data template fields when working with items in the Content Editor.

Developers use the IFrame field type:

- To add simple custom field editors to data templates.
- To add reports and other custom applications to data templates.

Fields of type IFrame load the URL specified in the source property of the field definition item into an IFrame in the Content Editor. Sitecore adds the following query string parameters to the URL:

Parameter	Function
`id`	The GUID of the item selected by the user.
`la`	The language code selected by the user.
`vs`	The version number selected by the user.

Developers can pass additional variables by adding query string parameters to the URL specified in the source property of the IFrame field definition item.

How to Create an IFrame Data Template Field

To create an IFrame data template field:

1. In the Visual Studio web application project, create a web page that contains the user interface to display in the iFrame.

   For information about the query string parameters passed to the IFrame field, see the section *The IFrame Field Type*.
2. In the **Template Manger** or the **Content Editor**, select the data template definition item.
3. In the editing pane, click the **Builder** tab.
4. In the **Template Builder**, add a field of type IFrame, and then save the data template.
5. Expand the section definition item, and select the IFrame data template field definition item.
6. In the **Data** section, in the **Source** field, enter the URL of the web page created previously containing the IFrame user interface.

For information about adding a field, see the manual *Data Definition Reference*.

3.4.2 Item Editors

Item editors define alternate user interfaces for working with items in the Content Editor. Developers implement custom item editors:

- To provide arbitrary applications for working with specific items or items based on specific data templates.
- To provide summaries, reports, or other interfaces for individual items or items based on specific data templates.
- To provide a single user interfaces to update a hierarchy of items.

Item editors appear as tabs in the Content Editor when a user selects an item. Each tab in the Content Editor activates a different item editor. Developers can associate any number of item editors with an item or a data template. The standard values item for each data template defines default item editors for all items associated with that template.

The default editor for all items appears as the **Content** tab in the Content Editor. This interface displays field editing controls grouped in sections as defined by the item's data template and its base templates.

Sitecore associates various item editors with specific types of items by default. The Folder item editor for folders defines the **Folder** tab listing existing child items and options to create new child items. The Builder item editor for data templates defines the **Builder** tab for working with the data template sections and fields comprising the template.

Item editors are URL-addressable web applications. Sitecore manages item editors under `/Sitecore/Content/Applications/Content Editor/Editors` in the *Core* database.

Sitecore passes the following query string parameters to item editors:

Parameter	Function
`id`	The GUID of the item selected by the user
`language`	The language code selected by the user.
`version`	The version number selected by the user.
`database`	The name of the database containing the item selected by the user.

Note
You can pass additional variables by adding query string parameters in the URL specified in the **URL** field in the **Data** section of the editor definition item.

Note
Item editors supersede the **Editor** property in previous versions of Sitecore.

How to Configure Item Editors

To configure item editors:

1. In the **Template Manager** or the **Content Editor**, select the standard values item or the individual item.
2. Click the **Configure** tab and in the **Appearance** group, click **Editors** and the **Custom Editors** dialog box appears.
3. In the **Custom Editors** dialog box, in the **All** field, expand the branches and double-click item editors to add them to the **Selected** list.

4. To change the order of item editor tabs, in the custom editors dialog, in the **Selected** field, select an item editor, and then click the arrows at the right to sort the selection.
5. To remove an item editor, in the **Custom Editors** dialog box, in the **Selected** list, double-click the item editor.

How to Create a Custom Item Editor

To create a custom item editor:

1. In the Visual Studio web application project, create a web page containing the custom item editor user interface.
2. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the `/Sitecore/Content/Applications/Content Editor/Editors/Items` item.
3. In the **Content Editor**, insert an item editor definition item using the `/Sitecore Client/Content Editor/Editor` data template.
4. In the item editor definition item, in the **Data** section, in the **Header** field, enter a value that **Content Editor** should display in the tab that activates the item editor.
5. In the **Icon** field, enter the icon to display in the tab that activates the item editor.
6. In the **URL** field, enter the URL of the web page created previously containing the item editor user interface.
7. Save the item editor definition item.
8. In the Sitecore desktop, select the *Master* database.

For information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

For more information about the query string parameters passed to the custom editor, see the manual *Client Configuration Cookbook*.

3.5 Insert Options

Insert options control the types of items users can insert beneath existing items.

Developers apply insert options:

- To help users create appropriate types of items under existing items.
- To restrict the types of items users can create under existing items.
- To allow different users to create different types of items under existing items.
- To help users create a number of items with a single action in the user interface.
- To create items programmatically, for example, by invoking a wizard.

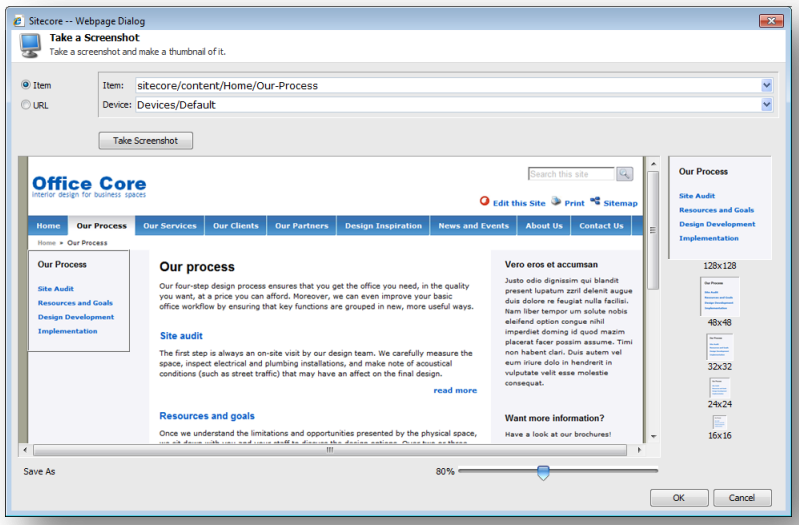
For more information about insert options, see the manuals *Data Definition Reference* and *Data Definition Cookbook*.

3.6 Item Thumbnails

You can use the item thumbnail tool to take screenshots of Sitecore components as they appear on your website. Creating thumbnail images gives developers a visual representation of the controls available to them. For example, create thumbnails to show a preview of available components in the Select a Rendering or Layout Presets dialog boxes.

To create an item thumbnail:

1. In the **Content Editor**, click the **View** tab and select the **Standard Fields** check box.
2. Select an appropriate item. For example, the **Side Menu** rendering — `/sitecore/layout/Renderings/Starter Kit/Side Menu`
3. In the **Side Menu** item, navigate to the **Appearance** section, **Thumbnail** field.
4. Click **Take Screenshot** to open the screenshot tool.



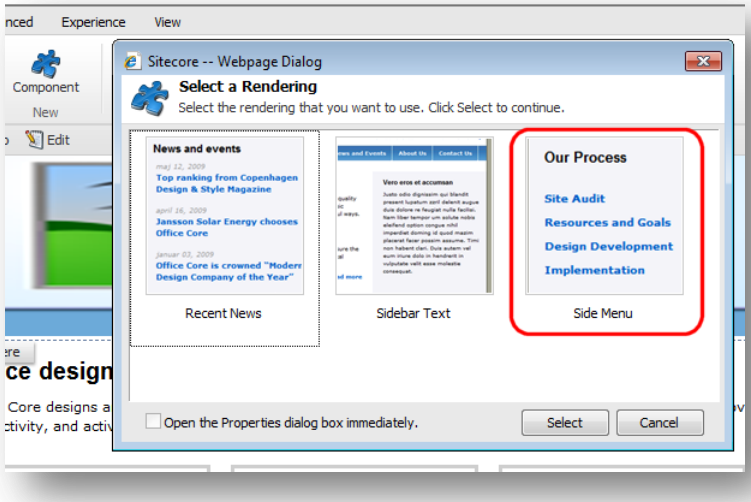
5. In the **Item** field click the drop-down menu and select an item in the content tree.

   For example to capture a screenshot of the **Side Menu** rendering on the **Our Process** page, select the **Our Process** item in the content tree.
6. In the **Device** drop-down menu, select **Default**.
7. Click **Take Screenshot**.
8. You can use the cropping square to select the side menu as it appears on this page and use the slider control to zoom in or out.
9. Click **Select** when you are satisfied with your selection.
10. Click **Save** to save the changes to the item.

Note
In the screenshot tool, you can either select an item from the content tree or enter a URL path to an item. The URL should be in the following format: http://*<your-site>*/*<page>*

To preview an item thumbnail in the Page Editor:

1. Open the **Page Editor** and on the **View** tab, select the **Designing** check box to activate design mode.
2. On the **Home** tab, click **Component** to display available placeholders on the page.
3. Select one of the available placeholders and click **Add to here** to open the **Select a Rendering** dialog box.



Note
The side menu rendering used in this example is already one of the allowed controls on this placeholder.

The Thumbnail field contains the following additional options:

- Browse – opens the image in the Media Browser
- Open Media Library – opens the image in the Media Library as part of the content tree
- Edit Image – opens the image in your default image editor
- Clear – removes the image
- Refresh – reloads the item in the Content Editor

Note
Screenshots created with the thumbnail tool are stored in the Sitecore Media Library as .png files.

Chapter 4

Data Validation

This chapter provides procedures to configure data validation. Developers use data validation to enforce rules for data entry. This chapter contains the following sections:

- Validation Overview
- Types of Validation
- Validation Error Levels
- Configuring Validation Rules
- Registering Validators
- Custom Validators
- Validation Actions

4.1 Validation Overview

You can configure data validation to control how Sitecore handles invalid data.

You can configure Sitecore to invoke validation rules in the Quick Action Bar, in the Validator Bar, when the user chooses the Validate command in the Proofing group on the Review tab, and when the user chooses a specific workflow command.

Note
In general, select the same validation rules for all four types of validation.

You can configure different validators to for each data template field, for all data template fields of a data template field type, for individual items, for all items based on a data template, and for all items.

Important
For each item, Sitecore invokes the item validation rules defined in the item or the standard values item associated with its data template, as well as global validation rules. For each data template field, Sitecore invokes the validation rules defined in the data template field definition item, as well as the validation rules defined in the data template field type validation rules definition item. The number of validators that the system must invoke for an item affects the server resources used and can affect client performance.

Important
Do not edit the default validator definition items under `/sitecore/system/Settings/Validation Rules`. Instead, register additional validators by creating additional validator definition items.

Note
After you stop editing a field value, Sitecore automatically invokes validation rules asynchronously, and then updates the user interface after validation completes. The `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name Validators.UpdateDelay` controls the length of time between the cessation of editing activity and the invocation of the validators. To disable this automatic revalidation feature, set the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name Validators.AutomaticUpdate` to `False`.

Note
The default Sitecore configuration disables validation in the Quick Action Bar. To enable Quick Action Bar validation, right-click the **Quick Action Bar**, and then select **Validation Rules**.

Tip
Avoid use of the **Validation** and **Validation Text** fields in the **Data** section of field definition items available in earlier versions of Sitecore in favor of the data validation features described in this document.

4.1.1 Types of Validation

You can validate various types of data.

Data Template Field Validation

You can configure Sitecore to validate the contents of each field in each data template.

Field Type Validation

You can configure Sitecore to validate the contents of each of the different field types.

Item Validation

You can configure validation for individual items.

Standard Validation

You can configure validation for all items based on a data template by configuring validation rules in the standard values of that data template.

Global Item Validation

You can configure validation that applies to all items.

4.1.2 Validation Error Levels

Validation error levels control the actions Sitecore takes on validation results. Each validator returns one of the following error levels defined in `Sitecore.Data.Validators.ValidatorResult`.

Error Level	UI Color	Function
`Unknown`	Gray	Validation incomplete or result otherwise unknown.
`Valid`	Green	Valid.
`Suggestion`	Green	Suggestions appear in the user interface.
`Warning`	Orange	Warnings appear in the user interface.
`Error`	Red	Errors prevent the user from completing workflow commands associated with the workflow validation action.
`CriticalError`	Red	Critical errors cause a modal warning when the user attempts to save an item and prevent the user from completing workflow commands associated with the workflow validation action.
`FatalError`	Red	Fatal errors cause a modal warning and prevent the user from saving the item.

4.2 Configuring Validation Rules

You can configure Sitecore to invoke validation rules in the Quick Action Bar, in the Validator Bar, when the user chooses the Validate command in the Proofing group on the Review tab, and when the user chooses a specific workflow command.

4.2.1 How to Configure Quick Action Bar Validation Rules

To configure validation rules to invoke in the **Quick Action Bar**:

1. In the **Content Editor**, select the appropriate validation rules definition item as described in the following sections.
2. In the **Validation** section, in the **Quick Action Bar** field, select validation rules.

4.2.2 How to Configure Validate Button Validation Rules

To configure validation rules to invoke when the user clicks the **Validate** command in the **Proofing** group on the **Review** tab:

1. In the **Content Editor**, select the appropriate validation rules definition item as described in the following sections.
2. In the **Validation** section, in the **Validate Button** field, select validation rules.

4.2.3 How to Configure Validation Bar Validation Rules

To configure validation rules to invoke in the **Validation Bar**:

1. In the **Content Editor**, select the appropriate validation rules definition item as described in the following sections.
2. In the **Validation** section, in the **Validation Bar** field, select validation rules.

Note
The validation bar does not appear in Content Editor if the **value** attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name ContentEditor.ShowValidatorBar` is `false`.

4.2.4 How to Configure Workflow Validation Rules

To configure validation rules to invoke when the user chooses a workflow command:

1. In the **Content Editor**, select the appropriate validation rules definition item as described in the following sections.
2. In the **Validation** section, in the **Workflow** field, select validation rules.

4.2.5 How to Create a Workflow Command or State Validation Action

To configure Sitecore to invoke the validation workflow action when the user chooses a workflow command or when the user chooses a workflow command:

1. In the **Content Editor**, select the workflow state or command.

2. Insert a validation workflow action definition item based on the `/System/Workflow/Validation Action` data template.
3. In the validation workflow action, in the **Data** section, in the **Type** field, enter:

```
Sitecore.Workflows.Simple.ValidatorsAction,Sitecore.Kernel.
```

4. In the **Max Result Allowed** field, enter the maximum `Sitecore.Data.Validators.ValidatorResult` value the workflow validation action can generate. If validation generates a higher validation error level, the workflow validation action will prevent the user from completing the workflow command. The default value for this field is `Warning`.
5. In the **Unkown Result** field, enter a message to display to the user if the validator returns `Unknown` as the validation result.
6. In the **Warning Result** field, enter a message to display to the user if the validator returns `Warning` as the validation result.
7. In the **Error Result** field, enter a message to display to the user if the validator returns `Error` as the validation result.
8. In the **Critical Error Result** field, enter a message to display to the user if the validator returns `CriticalError` as the validation result.
9. In the **Fatal Error Result** field, enter a message to display to the user if the validator returns `FatalError` as the validation result.

4.2.6 How to Configure Validation Rules for All Instances of a Data Template Field Type

To configure validation rules to invoke for all instances of a specific data template field type:

1. In the **Content Editor**, select the `/Sitecore/System/Settings/Validation Rules/Field Types` item.
2. Select the existing data template field type validation rules definition item, or insert a data template field type validation rules definition item using the `/System/Validation/Field Type Validation Rules` data template.

   For the name of the field type validation definition item, use the name of the field type as it appears in the **Type** field in the **Data** section of the field type definition item.
3. In the data template field type validation rules definition item, in the **Validation Rules** section, configure validation rules.

4.2.7 How to Configure Validation Rules for All Items:

To configure validation rules for all items:

- In the **Content Editor**, select the `/Sitecore/System/Settings/Validation Rules/Global Rules` item, and then configure validation rules.

4.2.8 How to Configure Validation Rules for an Individual Item or All Items Based on a Specific Data Template

To configure validation rules for an individual item or all items based on a specific data template:

1. In the **Template Manager** or the **Content Editor**, show the standard fields.
2. Select standard values item for the data template, or select the individual item.
3. Configure the validation rules.
4. Hide the standard fields.

For information about showing and hiding the standard fields, see section *How to Show or Hide the Standard Fields*.

4.2.9 How to Configure Validation Rules for a Specific Data Template Field

To configure validation rules for a specific data template field:

1. In the **Template Manager** or the **Content Editor**, select the data template field definition item.
2. In the **Validation Rules** section, configure field validation rules.

4.2.10 Default Item Validators

You can configure item validation rules using the following default validators.

Validator	Function
Broken Links	Identifies invalid references.
Duplicate Name	Identifies items with a name or display name that matches the name or display name of another item under the same parent (case-insensitive).
Full Page XHtml	Checks the validity of the output of requesting the item using the default device.
Media Size Too Big	Identifies media too large to serialize for database storage.
URL Characters	Checks item names for characters requiring escape sequences in URLs.

Note
XHTML validators validate the XML against the schema defined by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name XHtmlSchemaFile`.

4.2.11 Default Field Validators

You can configure field validation rules using the following default validators.

Validator	Function
Broken Links	Identifies invalid references.
Is Email	Identifies invalid email addresses.
Is Integer	Identifies invalid integers.
Is XHtml	Checks validity of field against local XHTML schema.
Max Length 40	Identifies field values containing more than 40 characters.
Rating 1 to 9	Identifies invalid integers, negative numbers, and values greater than nine.
Required	Identifies empty fields.
Spellcheck	Identifies fields containing spelling errors.
W3C XHtml Validation	Identifies invalidity against remote W3C validation service.
Alt Required	Identifies missing Alt text in media library image data templates.
Extension May Not Start with a Dot	Identifies media items with a dot character in the field named extension, which is invalid.
External Link Target	Identifies external links in Rich Text Editor fields that should open in a new browser window and provide a title.
Image Has Alt Text	Identifies image fields that do not contain alternate text and that reference media items that do not contain alternate text.
Image Has Alt Text from Media Library	Identifies image fields that do not specify alternate text and that reference media items that do contains alternate text.
Image Size	Identifies image field that references a media item that exceeds the size limit.
Rich Text Image Size	The width of images used in the Rich Text Editor must not exceed the `value` of the `/configuration/sitecore/settings/setting` element in `web.config` with `name` `Media.MaxImageWidth`.

Note
The `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name` `HtmlEditor.ValidatorServiceUrl` controls the URL used for W3C XHtml Validation.

4.3 Registering Validators

You can configure validation rules definition items to use the default validators, or register custom validators.

Important
To maximize the performance of the Sitecore user interfaces, validators that consume significant resources and especially processing time, such as to invoke external services or other potentially long-running operations, should run in separate threads. Using a separate thread allows the system to invoke additional validators before a validator completes. Sitecore will update the user interface as the different validators complete.

4.3.1 How to Register a Validator

To register a field validator:

1. In the **Content Editor**, select the appropriate project-specific folder under the `/Sitecore/System/Settings/Validation Rules/Field Rules` item or the `/Sitecore/System/Settings/Validation Rules/Item Rules` item.
2. Insert a validator definition item using the `/System/Validation/Validation Rule` data template.
3. In the validator definition item, in the **Data** section, in the **Type** field, enter the class signature.
4. In the **Parameters** field, enter URL-escaped key=value parameters, separated by ampersand characters (“`&`”).
5. If the validator should run in a separate thread, then in the **Content Editor**, in the validator definition item, in the **Data** section, select **Use Thread**.

4.3.2 How to Register a Regular Expression Field Validator

To register a regular expression field validator:

1. Register the field validator.
2. In the **Content Editor**, in the validator definition item, in the **Data** section, in the **Type** field, enter the following .NET type signature:

```
Sitecore.Data.Validators.FieldValidators.RegexValidator,Sitecore.Kernel
```

3. In the **Parameters** field, enter:

```
Pattern=RegularExpression&Text=Message
```

4. Replace `RegularExpression` with a regular expression.
5. Replace `Message` with the message to display if the field does not match the pattern.

   Sitecore will call `String.Format()` to substitute `{0}` in the message with the name of the field.

For more information about registering a validator, see the section *How to Register a Validator*.

4.3.3 How to Register an Integer Field Validator

To register an integer field validator:

1. Register the field validator.

2. In the **Content Editor**, in the validator definition item, in the **Data** section, in the **Type** field, enter the following .NET type signature:

```
Sitecore.Data.Validators.FieldValidators.IntegerFieldValidator, Sitecore.Kernel
```

3. In the **Parameters** field, enter the following:

```
AllowNegative=AllowNegative=Boolean&AllowZero=Boolean
```

4. Replace `Boolean` with `True` or `False` as appropriate.

For more information about registering a validator, see the section *How to Register a Validator*.

4.3.4 How to Register an Integer Range Field Validator

To register an integer field range validator:

1. Register the field validator.
2. In the **Content Editor**, select the validator definition item.
3. In the **Data** section, in the **Type** field, enter the following .NET type signature:

```
Sitecore.Data.Validators.FieldValidators.IntegerRangeValidator, Sitecore.Kernel
```

4. In the **Parameters** field, enter:

```
Min=Minumum&Max=Maximum
```

5. Replace `Minimum` with the minimum allowed value for the field.
6. Replace `Maximum` with the maximum allowed value for the field.

For more information about registering a validator, see the section *How to Register a Validator*.

4.3.5 How to Register a Maximum Length Field Validator

To register a maximum field length validator:

1. Register the field validator.
2. In the **Content Editor**, select the validator definition item.
3. In the **Data** section, in the **Type** field, enter the following:

```
Sitecore.Data.Validators.FieldValidators.MaxLengthFieldValidator, Sitecore.Kernel
```

4. In the **Parameters** field, enter the following:

```
MaxLength=MaximumLength
```

5. Replace `MaximumLength` with the maximum allowed length of the field value.

For more information about registering a validator, see the section *How to Register a Validator*.

4.3.6 How to Register Validators for Specific Items and Items Based on Specific Templates

To register validation rules for specific items and items based on specific templates:

1. In the **Content Editor**, select the `/Sitecore/System/Settings/Validation Rules/Item Rules` item.
2. Insert a validation rule definition item using the `/System/Validation/Validation Rule` data template.

3. Select the validation rule definition item.
4. In the **Data** section, in the **Type** field, enter the class signature.
5. In the validation rule definition item, in the **Data** section, in the **Parameters** field, enter URL-escaped key=value parameters, separated by ampersand characters ("`&`").

4.3.7 How to Disable Default Validation Rules

To disable default validation rules for all items:

1. In the **Content Editor**, select the `/Sitecore/System/Settings/Validation Rules/Global Rules` item.
2. In the **Validation Rules** section, configure validation rules.
3. Select the `/Sitecore/System/Settings/Validation Rules/Field Types` item.
4. Configure validation in field type validation rules definition items.
5. Select the `/Sitecore/Templates/System/Media` item.
6. Show the standard fields.
7. For each of the data templates used for media, in the **Template Manager** or the **Content Editor,** select the data template definition item, then click the **Content** tab in the editing pane, and then in the **Validation Rules** section, configure validation.
8. Hide the standard fields.

For more information about showing and hiding the standard fields, see the section *How to Show or Hide the Standard Fields*.

Important
No character in an item name can match the regular expression specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name InvalidItemNameChars`. An item name must match the regular expression specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name ItemNameValidation`. Do not change these two settings.

4.3.8 How to Override the Default Error Level for a Validator

To override the default error level for a validator:

1. In the **Content Editor**, select the validator definition item.
2. In the **Data** section, in the **Parameters** field, enter:

```
Result=ErrorLevel
```

3. Replace `ErrorLevel` with the name of the error level in `Sitecore.Data.Validators.ValidatorResult`.

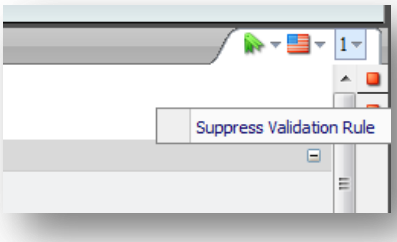
4.3.9 How to Suppress Validation Rules

You can suppress global item validation rules for individual content items.

In the Content Editor, entries in the validation bar, to the right of the editing pane indicate item validation violations. If you right-click on an entry, Sitecore presents available validation actions to resolve the violation.

To suppress a validation rule:

1. In the **Content Editor**, select a content item that has a validation rule violation.
2. In the validation bar, to the right of the editing pane, right click the validation icon and then click *Suppress Validation Rule*.



3. If you choose to click *Suppress Validation Rule*, Sitecore adds the validator to the __**Suppressed Validation Rules** field defined in the standard template.

The validation bar does not process the validators specified in this field.

4.4 Custom Validators

This section describes procedures to implement a custom validator.

4.4.1 How to Implement a Custom Validator

A custom validator involves two components: a .NET class and a validator definition item.

4.4.2 How to Implement a Custom Validator

To implement a custom validator:

1. In the Microsoft Visual Studio web application project, create a class based on the following prototype:

```
using System.Runtime.Serialization;
using Sitecore.Data.Validators;

namespace Namespace.Data.Validators.ItemValidators//TODO:namespace (FieldValidators)
{
  [Serializable]
  public class ClassName : Sitecore.Data.Validators.StandardValidator
  {
    public ClassName(){}//TODO:class name

    public ClassName(
      SerializationInfo info,StreamingContext context):base(info,context)
    {
    }

    public override string Name
    {
      get
      {
        return(GetType().ToString());//TODO:validator name
      }
    }

    protected override ValidatorResult GetMaxValidatorResult()
    {
      return(GetFailedResult(ValidatorResult.Error));//TODO:error level
    }

    protected override ValidatorResult Evaluate()
    {
      if(false)//TODO:validate ControlValidationValue
      {
        return(ValidatorResult.Valid);
      }
      else
      {
        Text = "error message";//TODO:error message
        return(GetFailedResult(ValidatorResult.Error));//TODO:error level
      }
    }
  }
}
```

2. In the class, replace `Namespace.Data.ItemValidators` with the appropriate namespace.
3. Replace all instances of `ClassName` with the class name.
4. Replace `GetType().ToString()` with the friendly name of the validator.

5. Replace all instances of with the appropriate validation error level.
6. Replace `false` with logic to validate the field value in `ControlValidationValue` or the item returned by `GetItem()`.
7. Replace `error message` with an error message.
8. Register the validator.

Important
If the result of the `GetMaxValidatorResult()` method is `FatalError` or `CriticalError`, then Sitecore will block operations such as save or workflow commands in the user interface as appropriate until validation completes. To avoid blocking user interface actions during validation, the `GetMaxValidatorResult()` method of expensive validators should not return `FatalError` or `CriticalError`. Validation Bar and Quick Action Bar validators never block user interface operations.

For more information about registering a validator, see the section *How to Register a Validator*.

4.5 Validation Actions

Validation actions represent clickable operations in the user interface to correct validation errors.

4.5.1 How to Create a Validation Action

To create a validation action for a Single-Line Text data template field:

1. In the Visual Studio web application project, create a class based on the following prototype:

```
using Sitecore.Shell.Framework.Commands.ContentEditor.Validators;

namespace Namespace.Shell.Framework.Commands.ContentEditor.Validators//TODO:namespace
{
  public class ClassName:ValidatorCommand//TODO:class name
  {
    public override void Execute(CommandContext context)
    {
      var validator=GetValidator(context);
      if(validator!=null)
      {
        var control = GetControlToValidate(validator);
        if (control!= null)
        {
          control = control as Sitecore.Web.UI.HtmlControls.Control;
          if(control!=null)
          {
            control.Value=Value;//TODO:logic
            Validate();
          }
        }
      }
    }
  }
}
```

2. In the class, replace `Namespace.Shell.Framework.Commands.ContentEditor.Validators` with the namespace of the class.
3. Replace `ClassName` with the name of the class.
4. Replace `Value` with the validated value for the field.
5. In the **Content Editor**, select the `/Sitecore/System/Settings/Validation rules/Field Rules` item.
6. In the **Content Editor**, insert a validation action definition item using the `/System/Menus/Menu Item` data template.
7. In the validation action definition item, in the **Data** section, in the **Display Name** field, enter the label the user should select in the user interface to invoke the validation action.
8. In the `/App_Config/commands.config` file, insert a new command based on the following prototype:

```
<command name="validator:ClassName" type="Namespace.Class,Assembly"/>
```

9. In the `/App_Config/commands.config` file, replace `ClassName` with the name of the class, and then replace `Namespace.Class,Assembly` with the namespace and the class signature.

Important
Because unsaved values are only available in the UI and not in the database, you cannot use `Sitecore.Data.Items.Item` and other APIs in validation actions. The API used to validate fields depends on the field type, and may involve JavaScript, such as for values of Rich Text Editor fields.

4.5.2 How to Use a Validation Action

To use a validation action:

1. In the **Content Editor**, edit an item that violates a validation rule that provides a validation action.
2. In the **Validation Bar**, right-click the validation indicator, and then select the validation action.

Chapter 5

The Page Editor

This chapter provides describes provides procedures to configure the Page Editor, which provides CMS features such as inline editing and the design mode.

This chapter contains the following sections:

- Page Editor Overview
- Placeholder Settings
- Page Editor Modes
- Edit Frames

5.1 Page Editor Overview

As a user navigates the web site, the Page Editor superimposes user interface elements that contain editing functions on the items the user selects.

5.1.1 Design Mode in the Page Editor

In the Page Editor, you can use design mode to configure layout details for content items. Design mode prevents users from binding presentation components other than those expressly allowed using Placeholder Settings.

Note
Design mode enforces component nesting order, prevents users from attempting to bind presentation components to placeholders that do not exist, and inserts fully-qualified placeholder keys.

5.2 Placeholder Settings

Placeholder settings control which presentation components users can bind to placeholders in Page Editor, design mode. For more information about design mode in the Page Editor, see the section Design Mode in the Page Editor.

Developers use placeholder settings:

- To help users choose appropriate presentation components to bind to placeholders.
- To prevent certain users from binding components to specific placeholders.

Layout details reference placeholders either by placeholder key, or by fully qualified placeholder key. For example, the fully qualified placeholder key for a placeholder with key `content` in a sublayout bound to a placeholder with key `main` in a layout would be `/main/content`.

For more information about layout details and placeholders, see the manual *Presentation Component Reference*.

Note
Design mode in the Page Editor uses fully qualified placeholder keys, but layout details do not require fully qualified placeholder keys.

Placeholder settings definition items with names matching placeholder keys automatically apply to those placeholders unless layout details specify placeholder settings. The placeholder settings definition item named `content` automatically applies to all placeholders with the key `content`, including nested placeholders, except where layout details specify placeholder settings. For the name of the default placeholder settings definition item for a placeholder, use the placeholder key. For example, for the placeholder with key `content`, edit the placeholder settings definition item named `content`. Alternatively, by convention, use the fully qualified placeholder key, replacing slash characters ("`/`") with dash characters ("`-`"). For example, to control the placeholder with key `content` in a sublayout intended bound to a placeholder with key `main` in a layout, insert a placeholder settings definition item named `content`, or insert a placeholder settings definition item named `main-content` and associate this placeholder settings definition item with the fully qualified placeholder key `/main/content` in layout details.

Developers configure placeholder settings using placeholder setting definition items. Developers can control the icon displayed for each placeholder by assigning an icon to each placeholder settings definition item. Developers can restrict which users can bind presentation components to a placeholder by denying write access to the placeholder settings definition item. Developers can enter HTML in the Description field of the placeholder settings definition item to control the tooltip that appears when a user moves the mouse over the placeholder in the Page Editor, design mode.

Important
Sitecore does not automatically apply placeholder settings using fully qualified placeholder keys to the corresponding nested placeholders. For example, for the placeholder with fully qualified key `/main/content`, if it exists, the placeholder settings definition item with name `content` applies, but the placeholder settings definition item with name `main-content` does not. To cause different uses of placeholders to apply different placeholder settings definition items, configure placeholder settings for placeholder keys or fully qualified placeholder keys using layout details.

5.2.1 How to Create a Placeholder Settings Definition Item

To create a placeholder settings definition item:

1. In the **Content Editor**, select the `/Sitecore/Layout/Placeholder Settings` item.
2. Insert a placeholder settings definition item using the `System/Layout/Placeholder` data template. Name the placeholder settings definition item after the key of the placeholder that it controls.
3. Select the placeholder settings definition item.
4. In the **Data** section, in the **Allowed Renderings** field, select the presentation components users can bind to the placeholder.
5. In the **Description** field, enter a summary of the placeholder. Consider including a graphic indicating the location of the placeholder within the layout or sublayout.
6. Set the icon for the placeholder settings definition item to control the image that appears next to the name of the placeholder in design mode.
7. To control which users can bind sublayout and renderings to the placeholder, configure write access rights for the placeholder settings definition item.

For more information about setting the icon for an item, see the section *How to Set the Icon for an Item*.

5.2.2 How to Configure Placeholder Settings for a Data Template or an Individual Item

To configure placeholder settings for a data template or an individual item:

1. In the **Template Manager** or the **Content Editor**, select the standard values item or the individual item.
2. Click the **Presentation** tab and in the **Layout** group, click **Details**.
3. In the **Layout Details** dialog box, under the appropriate device, click **Edit**.
4. In the **Device Editor**, click **Placeholder Settings**, and then click **Add**.
5. In the **Select the Placeholder Settings** dialog box, in the **Placeholder Key** field, enter the placeholder key or a fully qualified placeholder key.
6. For the **Settings Item**, click **Browse**, and then select the placeholder settings definition item.

5.2.3 How to Assign Components and Edit Placeholder Settings using the Page Editor

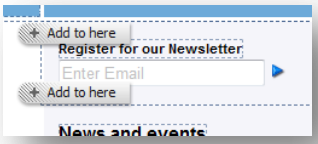
If you add a new component, using the Page Editor, there is no need to specify a placeholder first. Depending on where you want to add a component, the correct placeholder is assigned automatically.

Assigning Components

To add a component using the Page Editor:

1. Open the **Page Editor**. Ensure that you have design mode enabled.
2. On the **Home** tab, click **Component**.

You can now see all the available placeholders where it is possible to add components to the page.



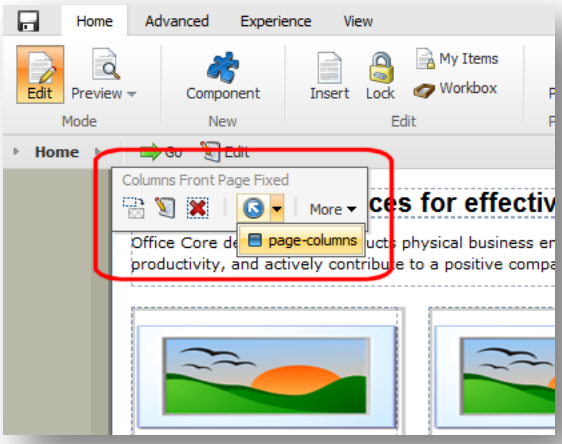
3. To add a control, click *Add to here*.
4. In the **Select a Rendering** dialog box, choose a component.
5. When you have chosen a component, click **Select** and then save your changes.

Note
Developers can specify which additional dialog boxes should appear after selecting a component to add. For example, it may be useful to open the Properties or Data Source dialog boxes automatically if additional configuration of a component is necessary.

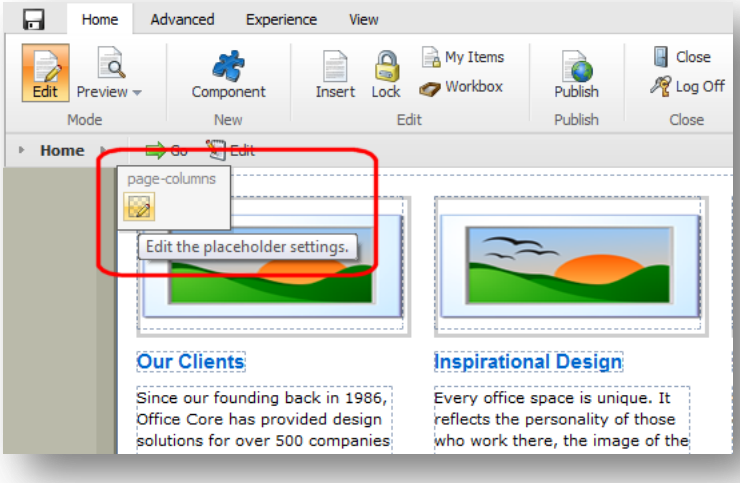
Editing Placeholder Allowed Controls

To edit placeholder allowed controls using the Page Editor:

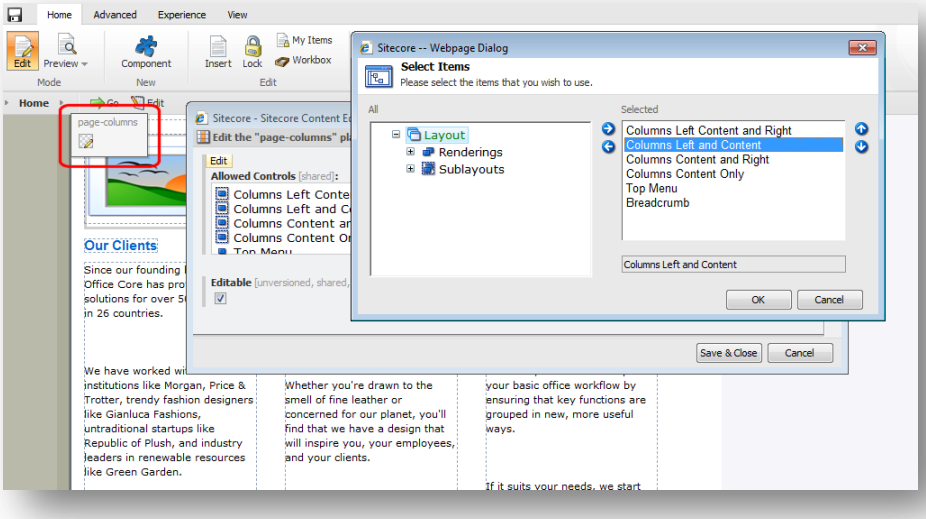
1. Select a placeholder or component.
2. If you select a component on the page, click *Show ancestors* to see which placeholder is available. Then click *Edit placeholder settings*. In this example, *page-columns* is the name of the placeholder.



3. Click *Edit the placeholder settings*.



4. In the **Edit Placeholder Settings** dialog box, click **Edit** to open the **Select Items** dialog box.



5. In the **Select Items** dialog box, select a rendering or sublayout control and click **OK**.

Note
You can add or remove placeholder allowed controls in this way.

5.3 Page Editor Modes

Each presentation component executes in a logical page mode depending on the user actions that caused Sitecore to invoke the component.

The modes in which a presentation component may execute include:

- The published web site.
- The browser-based debugger (with or without profiling and tracing active).
- Page Editor (with or without inline editing active).
- Preview.
- Design mode.

Presentation components can generate different output in different modes. For example, a presentation component may populate the HTML `<title>` element with the value of a field. The browser cannot provide inline editing facilities for the HTML `<title>` element. If the page does not use that field elsewhere, then the user cannot edit the field in the Page Editor. In this case, a presentation component used elsewhere on the page can output the value of the field elsewhere on the page if the user is inline editing. As another example, consider a rendering that should not output a `<div>` element if a field value is empty. To provide inline editing for the field, the rendering can output the `<div>` and the field editing control if the field has a value or if the user is in inline editing in the Page Editor.

5.3.1 How to Determine the Page Editor Mode

You can determine the Page Editor mode using the `sc:pageMode()` XSL extension function or properties of the `Sitecore.Context.PageMode` .NET object. For more information about the `sc:pageMode()` XSL extension function and the `Sitecore.Context.PageMode` .NET object, see the section Page Editor Modes.

For example, the following XSL code determines if the XSL rendering is running in any mode of the Page Editor, as opposed to the published site:

```
<xsl:if test="sc:pageMode()/pageEditor">
  <!-- the user is in the Page Editor. -->
</xsl:if>
```

The following C# code determines if the .NET component is running in the context of any mode of the Page Editor, as opposed to the published site:

```
if (Sitecore.Context.PageMode.IsPageEditor)
{
  // the user is in the page editor.
}
```

Note
The user can access multiple modes of the Page Editor simultaneously.

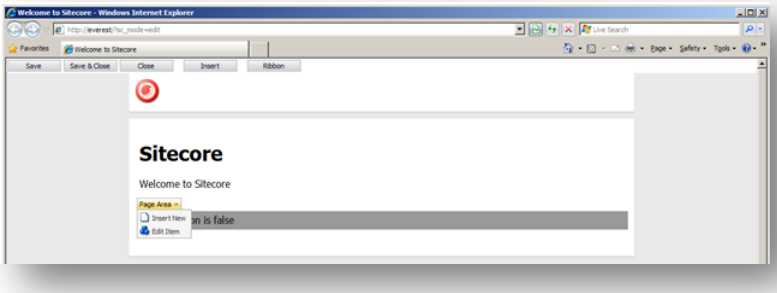
5.3.2 Page Editor Modes

The following table lists the XSL and .NET programming constructs to determine the page mode:

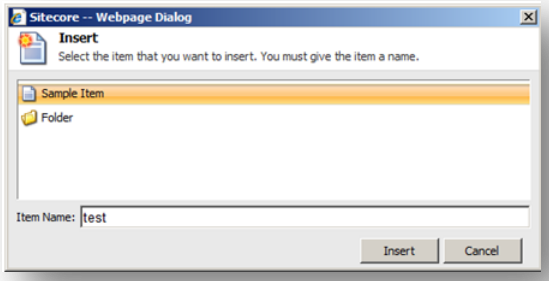
XSL `sc:pageMode()`	.NET `Sitecore.Context.PageMode`	**Description of Mode**
`/pageEditor`	`IsPageEditor`	Accessing the Page Editor.
`/pageEditor/edit`	`IsPageEditorEditing`	Inline editing and design in the Page Editor.
`/preview`	`IsPreview`	Previewing.
`/normal`	`IsNormal`	Published web site.
`/profile`	`IsProfiling`	Debugging and profiling.
`/debug`	`IsDebugging`	Debugging.

5.4 Edit Frames

Edit frames insert borders with context menus around markup structures when the user edits inline in the Page Editor. Edit frame commands activate Sitecore user interface commands. An edit frame generates a `<div>` element with an edit frame menu above it. Each edit frame command in the menu activates a Sitecore command.



The `Buttons` property of the edit frame control specifies an item in the *Core* database that contains children representing the edit frame menu items to expose in the edit frame menu. If you do not specify an edit frame menu, Sitecore uses the default edit frame menu defined by the `/Sitecore/Content/Applications/WebEdit/Edit Frame Buttons/Default` item in the *Core* database. This default edit frame menu contains the Insert edit frame menu command that allows the user to insert an item using the insert options defined for the data source of the edit frame.



For an example of implementing an edit frame, including a custom edit frame command, see the section How to Implement a Custom Field Editor.

For an example that uses an edit frame to open the field editor, see http://trac.sitecore.net/DefaultFieldEditor/.

Note
You can configure the default edit frame menu by setting the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with name `WebEdit.DefaultButtonPath`.

Note
Sitecore does not support nested edit frames. An edit frame can contain any other controls, including the FieldRenderer web control that generates inline editing controls, but cannot contain other edit frames.

For more information about the FieldRenderer web control, see the manual *Presentation Component Reference*.

5.4.1 How to Implement an Edit Frame Menu Command

To implement an edit frame menu command:

1. In the Visual Studio web application project, create an edit frame menu command class that inherits from `Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand`.
2. In the class, implement the `Execute()` method.
3. Add an edit frame menu command definition item to an edit frame menu.

For more information, see the section *How to Define an Edit Frame Menu*.

5.4.2 How to Define an Edit Frame Menu

To define an edit frame menu:

1. In the Sitecore desktop in the *Core* database, in the **Content Editor**, select the `/Sitecore/Content/Applications/WebEdit/Edit Frame Buttons` item.
2. In the **Content Editor**, insert an edit frame menu definition item using the `/System/WebEdit/Edit Frame Button Folder` data template.

   Tip
   Alternatively, you can duplicate the `/Sitecore/Content/Applications/WebEdit/Edit Frame Buttons/Default` edit frame menu definition.

3. Beneath the edit frame menu definition item, insert one or more edit frame menu command definition items using the `/System/WebEdit/Edit Frame Small Button` data template.

   Tips
   You can duplicate the `/Sitecore/Content/Applications/WebEdit/Edit Frame Buttons/Default/Insert` item to create your edit frame menu definition item.

   You can duplicate an existing edit frame menu command definition item and then update its properties.

   To insert the default field editor edit frame menu command, see the section How to Use the Default Field Editor.

4. In each edit frame menu command definition item, in the **Data** section, in the **Header** field, enter the menu label that should appear in the Page Editor for the menu command.
5. In the **Data** section, set the **Icon** field to an icon for the menu item.

   Tip
   You can also use this icon for the edit frame menu command definition item itself.

6. Set the **Click** field to the UI command the menu command invokes.
7. Set the **Tooltip** field to hover text for the menu item, and then save the edit frame menu command definition item.
8. In the Sitecore desktop, select the *Master* database.

For more information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

5.4.3 How to Insert an Edit Frame in a Layout or Sublayout

To insert an edit frame into a layout or sublayout, add code such as the following:

```
<sc:editframe runat="server">
...
</sc:editframe>
```

You can set properties of the edit frame as described in the section Edit Frame Properties.

5.4.4 How to Insert an Edit Frame in an XSL Rendering

To insert an edit frame into an XSL rendering, add code such as the following:

```
<sc:editFrame>
...
</sc:editFrame>
```

Set properties of the edit frame.

Edit Frame Properties

You can set the following properties of an edit frame:

Property (.NET)	Attribute (XSL)	Description
`Buttons`	`Buttons`	Edit frame menu items define available commands.
`CssClass`	`class`	Value for class attribute of `<div>` element generated by edit frame.
`DataSource`	`select`	The item to pass to edit frame menu commands.
`Debug`	`debug`	Controls whether Sitecore styles the `<div>` element generated by the edit frame.
`FramePadding`	`frame-padding`	Expand visible edit frame around contents by number of pixels indicated.
`HintOffset`	`hint-offset`	Pixels between edit frame hovering menu and edit frame region.
`Style`	`style`	Value for style attribute of `<div>` element generated by edit frame.
`Title`	`title`	Title of edit frame menu.
`Tooltip`	`Tooltip`	Tooltip for edit frame menu.

5.5 Creating Commands for a Field, Rendering or Placeholder

You can add commands to placeholders, renderings, fields, and markup elements so that they appear in the Page Editor interface. Use edit frames for markup elements, or the approach below for fields, renderings, and placeholders.

To create a new field command:

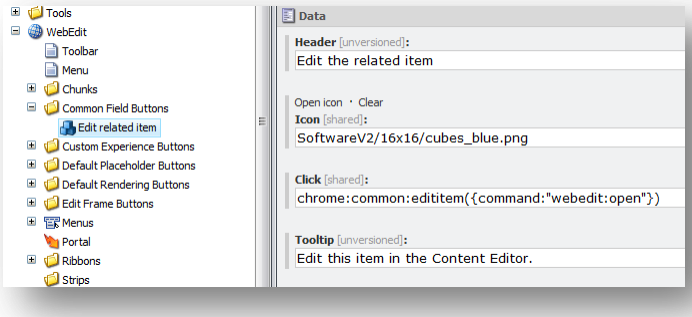
1. In the Visual Studio web application project, create a command class that inherits from: `Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand`
2. Compile your class.
3. Create a new entry in the `/App_Config/commands.config` file.
4. In the **Sitecore Desktop**, switch to the *Core* database. In the content tree, navigate to the **WebEdit** node:
   `/Sitecore/Content/Applications/WebEdit`

   There are several folders under the **WebEdit** node for each type of presentation component command:

   - Command Field Buttons
   - Custom Experience Buttons
   - Default Placeholder Buttons
   - Default Rendering Buttons

5. To define a new field command, select the **Common Field Button** folder:
   `/Sitecore/Content/Applications/WebEdit/Common Field Button`
6. Add a new button definition item using the `System/WebEdit/WebEdit Button` data template.
7. Give the button a name, such as *MyButton*.
8. In the **Data** section of the button definition item, complete the following fields:
   - **Icon** – Icon to display in the **Page Editor** when a user moves the mouse over this command.
   - **Click** – Point to the event command class you created in step 1.
   - **Tooltip** – Text to display when the user moves the mouse over this command.

   Example data section from the *Edit related item* command.



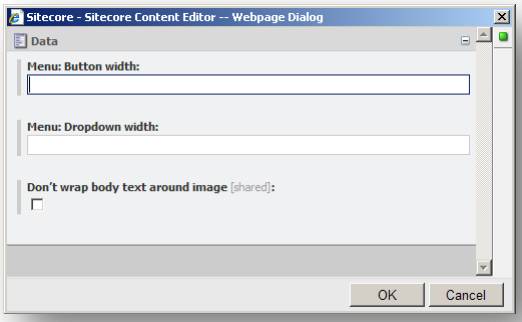
9. Save your changes and test your new command in the **Page Editor**.

Note
It is not possible to define a command for an individual placeholder. Placeholder commands apply to all placeholders.

Note
The `getChromeData` pipeline defined in the `web.config` file contains processors that determine which commands to expose for each edit frame, field, rendering, and placeholder displayed in the Page Editor.

5.6 The Field Editor

The field editor provides an interface within the Page Editor that allows the user to edit fields for which the Page Editor does not provide inline editing controls. The field editor user interface is highly streamlined in comparison to the Content Editor user interface.



You can configure edit frame commands to activate the field editor. Sitecore provides an edit frame command that you can use to activate the field editor to edit the fields that you specify. You can activate the field editor from within the Page Editor to edit fields that do not provide inline editing controls, or to edit fields from more than one item simultaneously.

For more information about edit frames, see section *Edit Frames*.

For an example that provides an edit frame menu command to open the field editor including all fields in the item that are not defined in the standard template and do not support inline editing, see http://trac.sitecore.net/DefaultFieldEditor/.

Warning
The field editor does not enforce item validation.

5.6.1 How to Use the Default Field Editor

The default field editor includes the fields that you specify.

To use the default field editor:

1. Define a field editor menu.
2. In the **Content Editor**, beneath the field editor menu definition item, insert the default field editor command definition item using the `System/WebEdit/Field Editor Button` data template.

   Note
   This field editor command definition item invokes the `webedit:fieldeditor` command.

3. In the default field editor command definition item, in the **Data** section, in the **Header** field, enter `Edit Item`.
4. In the **Data** section, set the **Icon** field to `people/16x16/cubes_blue.png`.

   Tip
   You can also use this icon for the edit frame menu command definition item itself.

5. In the **Content Editor**, in the default field editor command definition item, in the **Data** section, in the **Fields** field, enter the names or IDs of the fields to include in the field editor, separated by a pipe character ("|").

   Note
   Field names are not case-sensitive.

   Note
   If any of the specified fields do not exist, Sitecore does not generate an error.

6. In the Sitecore desktop, select the *Master* database.

For more information about defining a field editor menu, see the section *How to Define an Edit Frame Menu*.

For more information about setting the icon for an item, see the section *How to Set the Icon for an Item*.

For more information about selecting a database, see the section *How to Select a Database in the Sitecore Desktop*.

5.6.2 How to Implement a Custom Field Editor

To implement a field editor that includes fields dynamically:

1. In the Visual Studio web application project, create a class that inherits from the `Sitecore.Shell.Applications.WebEdit.Commands.FieldEditorCommand` class.
2. In the class, implement the `GetOptions()` method to create a list of fields to include in the field editor.
3. In the `GetOptions()` method, pass the list of fields to the constructor for the `Sitecore.Shell.Applications.WebEdit.PageEditFieldEditorOptions` class.
4. In the `GetOptions()` method, set additional properties of the `Sitecore.Shell.Applications.WebEdit.PageEditFieldEditorOptions` object.
5. In the `GetOptions()` method, return the `Sitecore.Shell.Applications.WebEdit.PageEditFieldEditorOptions` object.
6. Add the field editor command to the appropriate edit frame menu definition. For instructions to add a command to an edit frame menu definition item, see the section How to Define an Edit Frame Menu.
7. Configure an edit frame in a layout, sublayout, or XSL rendering to use the edit frame menu. For instructions to configure a layout, sublayout, or XSL rendering to use the edit frame menu, see the section How to Insert an Edit Frame in a Layout or Sublayout or How to Insert an Edit Frame in an XSL Rendering.

Note
The field editor merges all fields into a single visual section. You can set the `Title` and `Icon` properties of the `Sitecore.Shell.Applications.WebEdit.PageEditFieldEditorOptions` object to control the icon label used for this section. Alternatively, set the `PreserveSections` property to `true` to retain the original section structure.

Note
You can set the `DialogTitle` property of the `Sitecore.Shell.Applications.WebEdit.PageEditFieldEditorOptions` object to control the title of the field editor browser window.

Chapter 6

Security Based Configuration Features

This chapter describes the security features developers can use to optimize the user experience of content authors, including the Sitecore security roles.

This chapter contains the following sections:

- Sitecore Client Security Roles
- Security Presets
- Data Template Field Security

6.1 Sitecore Client Security Roles

Security administrators associate CMS users with one or more Sitecore client security roles to control the features available through the Sitecore user interfaces.

Sitecore applications do not expose every feature to every user. Limiting available features provides a number of benefits, including:

- Reducing the number of user interface elements improves performance and usability.
- Eliminating options prevents users from activating features inadvertently.
- Hiding unnecessary features minimizes training requirements.

A list of the specific features associated with each of the Sitecore roles is beyond the scope of this document.

The following table summarizes the options provided by membership in each of the default Sitecore client security roles:

Role	Function
Sitecore Account Managing	Access to user, role, and domain management features.
Sitecore Client Authoring	Access to content authoring features in the Content Editor, the Page Editor, and the other applications for content authors.
Sitecore Client Configuring	Access to item configuration features in the Content Editor.
Sitecore Client Designing	Access to design features in the Content Editor and the Page Editor
Sitecore Client Developing	Access to the Developer Center and other development features.
Sitecore Client Maintaining	Access to data template management features.
Sitecore Client Publishing	Access to site and item publishing features.
Sitecore Client Securing	Access to the Security Editor and the other features associated with assigning access rights.
Sitecore Client Translating	Access to the content translation features.
Sitecore Client Users	Allows users to log in to Sitecore clients. All of the other Sitecore client roles are members of the Sitecore Client Users role. Users in any Sitecore Client role are automatically members of the Sitecore Client Users role.

Security administrators use the following roles to restrict the features provided by membership of the Sitecore Client Authoring role. Administrators can make less sophisticated users members of these roles after assigning them the Sitecore Client Authoring role.

Role	Function
Sitecore Limited Content Editor	Limits features to provide a simplified Content Editor interface.

Role	Function
Sitecore Limited Page Editor	Limits features to provide a simplified Page Editor interface.
Sitecore Minimal Page Editor	Removes the ribbon from the Page Editor.

The following roles simplify security administration by providing default access rights to various branches in the database, as well as including membership of the relevant Sitecore client roles:

Role	Function
Author	Provides access rights to the `/sitecore/content` branch of the content tree. Member of Sitecore Client Authoring.
Designer	Provides access rights to the areas of the content tree required when making design changes. Member of Sitecore Client Designing.
Developer	Provides access rights to the areas of the content tree required during site development and maintenance. Member of Author, Designer, Sitecore Client Developing, Sitecore Client Maintaining, and Sitecore Client Configuring.

Sitecore also provides default roles to control access to features of the Online Marketing Suite (OMS):

Role	Function
Analytics Content Profiling	Provides access to maintain analytics profiles.
Analytics Maintaining	Provides access to the Marketing Center.
Analytics Reporting	Provides access to Sitecore Analytics reports.

If the default Sitecore roles do not meet your exact requirements for controlling the user interfaces, the access rights in the *Core* database give more precise control over which features Sitecore exposes to each user or role.

Important
To eliminate unnecessary features in the user interface and minimize the potential for users to invoke features by mistake, users should be members of the fewest possible roles. The only exception to this rule is the limited and minimal roles described previously; users should be members of these roles whenever possible. Most users should only be members of the Sitecore Client Authoring role. When users require additional functionality, security administrators can give them additional roles.

Important
Administrators have access to every feature, regardless of role membership. Minimize the number of administrators.

6.2 Security Presets

Security presets allow users to apply lists of access rights predefined by security administrators to items.

Security administrators use security presets:

- To simplify the process for end-users to apply access rights to content.
- To define common lists of access rights for users to apply to content.
- To provide single-click user interface components to apply lists of access rights that would otherwise require multiple actions.

You can access the security presets in the Content Editor, in the Presets group on the Security tab, and in the Security Editor, in the Presets group.

Tip
To minimize security administration, use security inheritance whenever possible.

6.2.1 How to Create a Security Preset

For the name of the security preset definition item, use a value appropriate for the command in the ribbon that will invoke this security preset.

To create a security preset by defining access rights through the user interface:

1. In the **Content Editor**, select the `/Sitecore/System/Settings/Security/Presets` item.
2. Use the `/System/Security/Security Preset` data template to insert a security preset definition item
3. In the **Data** section, in the **Security Preset** field, define access rights.

To create a security preset by copying access rights from an existing source item:

1. In the **Template Manager** or the **Content Editor**, select the source standard values item or the individual item.
2. In the **Template Manager** or the **Content Editor**, show the standard fields.
3. Show raw field values.
4. In the **Security** section, in the **Security** field, select the value, and then copy it to the Windows clipboard.
5. Hide the standard fields.
6. In the **Content Editor**, select the `/Sitecore/System/Settings/Security/Presets` item.
7. Insert a security preset definition item using the `/System/Security/Security Preset` data template.
8. In the **Security** section, in the **Security** field, paste the value from the Windows clipboard.
9. Hide raw field values.

For more information about showing and hiding the standard fields, see the section *How to Show or Hide the Standard Fields.*

For more information about showing and hiding raw field values, see the section *How to Show or Hide Raw Values*.

6.2.2 How to Apply a Security Preset

By default, to apply security presets, a user must be an administrator, or a member of the Sitecore Client Securing role.

Apply security presets using the **Content Editor** or the **Security Editor**.

Tip
Use Security Editor when applying access rights to multiple items.

To apply a security preset using **Content Editor**:

1. In the **Content Editor**, select the item.
2. Click the **Security** tab and in the **Presets** group, click a security preset command.

   Sitecore copies the access rights defined in the **Security Preset** field of the security preset definition item to the selected item.

To apply a security preset using the **Security Editor**:

1. In the Sitecore desktop, in the lower left corner, click the **Sitecore** button, and then click **Security Editor**. The **Security Editor** appears.
2. In the **Security Editor**, select the item.
3. In the **Presets** group, click the security preset command.

   Sitecore copies the access rights defined in the **Security Preset** field in the security preset definition item to the selected item.

6.3 Data Template Field Security

Data template field security separates the security rights that control access to an item from the security rights that control access to the individual field values in that item.

Developers use field security:

- To hide fields in the editing interfaces from some users.
- To make fields in the editing interfaces read-only for some users.
- To allow some accounts to update field values while preventing others from updating those fields.

For example, for search-engine optimization (SEO) purposes, an organization that uses the value of a data template field to populate the HTML `<title>` element may not want all users with write access to other fields to have write access to that field. Only certain users with write access to those items should also have access to update the fields used for SEO. You can meet this requirement with field security.

By default, access rights in the item define access rights to all fields in the item. If a field defines access rights, the user must have explicit field security access as well as access to the item containing the field.

How to Configure Data Template Field Security

To configure data template field security:

1. In the **Template Manager** or the **Content Editor**, select the data template field definition item.
2. Click the **Security** tab and in the **Security** group, click **Assign**. The **Security Settings** dialog box appears.
3. Grant and deny the field read and field write access rights for one or more accounts.

Chapter 7

Sitecore Client RSS Feeds

This chapter describe Sitecore client RSS feed features. RSS feeds syndicate information in a standard XML format. Sitecore client RSS feeds expose information and features to CMS users using the RSS XML format. CMS users can access Sitecore client RSS feeds using any RSS reader.

This chapter begins with an overview of Sitecore client RSS features, and then describes Sitecore client RSS configuration options.

This chapter contains the following sections:

- Sitecore Client RSS Feeds Overview
- Sitecore Client RSS Feeds Configuration

7.1 Sitecore Client RSS Feeds Overview

Sitecore client RSS feeds expose CMS features as Really Simple Syndication (RSS) feeds.

Sitecore manages the Sitecore client RSS feed definition items under the `/Sitecore/Content/Applications/Syndication/Feeds` item in the *Core* database.

7.1.1 Workflow Feed

The workflow feed syndicates information about items in a workflow.

The workflow feed includes an RSS entry for each state change for each item in the workflow.

The workflow feed appears as an RSS icon for each type of workflow shown in the Workbox.

7.1.2 Workflow State Feed

The workflow state feed syndicates information about items entering and exiting a workflow state.

The workflow state feed includes an RSS entry for each time that each item in the workflow enters or exits the workflow state.

The workflow state feed appears as an RSS icon for each state in each workflow shown in the Workbox.

7.1.3 Item Updated Feed

The item updated feed syndicates information about updates to an item, and optionally about updates to its descendants.

For items associated with a workflow, the item updated feed includes an RSS entry for each workflow state change. For items not associated with a workflow, the item updated feed includes an RSS entry for each new version in each language.

The item updated feed appears as the **Subscribe** command in the **Proofing** group on the **Review** tab in the **Content Editor**. You can choose to subscribe to changes to an item, or to changes to the item or any of its descendants.

For more information about Sitecore RSS features, see the manual *Presentation Component Cookbook* and the *Content Author's Cookbook*.

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

7.2 Sitecore Client RSS Feeds Configuration

This section describes configuration of Sitecore client RSS feeds.

7.2.1 The ClientFeeds.MaximumItemsInFeed Setting

The `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name` `ClientFeeds.MaximumItemsInFeed` specifies the maximum number of items to include in a Sitecore client RSS feed.

7.2.2 The ClientFeeds.ItemExpiration Setting

You can set the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with name `Clienteeds.ItemExpiration` to control the expiration of Sitecore client RSS syndication items. A value of zero ("`0`") for the `ClientFeeds.ItemExpiration` setting specifies that items do not expire. If you set this value to a positive integer, then RSS feed items on managed web sites expire from RSS feeds after the specified number of days.