Sitecore CMS 7 and later

Page Editor Recommended Practices for Developers

A Guide to Building for the Page Editor and Improving the Editor Experience

Table of Contents

Chapter 1

Introduction

This guide is designed to help you build for the Page Editor and improve the user experience for editors. It is primarily aimed at back-end developers. It is important for everyone involved in the implementation to consider building for the Page Editor as early on in the project lifecycle as possible.

Developers must design their solution to support both editing and designing capabilities (see Support Editing and Support Design). In addition, developers and project managers must budget for configuration and the extending of the Page Editor to improve the user experience. Finally, testing should ideally include a usability test of the Page Editor by the users of the solution.

The guide contains the following chapters:

- **Chapter 1 – Introduction**
  This chapter is a description of the content, aims, and the intended audience of this manual.

- **Chapter 2 – Support Editing**
  Instructions for supporting inline editing.

- **Chapter 3 – Support Design**
  Instructions for supporting the Page Editor's design mode.

- **Chapter 4 – Configuring the Page Editor**
  An overview of ways in which basic configuration can improve the Page Editor user experience.

- **Chapter 5 – Extending the Page Editor**
  An overview of ways in which the Page Editor can be customized and extended to improve the user experience.

- **Chapter 6 – Detecting the Page Editor**
  Instructions for detecting Page Editor mode in Javascript and C#, and ways to modify site behavior based on the page mode.

- **Chapter 7 – Tips and Tricks**
  Miscellaneous tips and tricks, and a summary of the Page Editor's role in Sitecore modules.

Chapter 2

Support Editing

The following chapter covers the fundamental requirements for supporting the ability to edit content inline using the Page Editor's Editing mode.

The chapter contains the following sections:

- Always Go Through the FieldRenderer Pipeline

2.1 Always Go Through the FieldRenderer Pipeline

If you render a field via the `FieldRenderer` pipeline, it is automatically editable in the Page Editor. All Sitecore controls and helpers eventually go through this pipeline:

- Sitecore controls for use in XSLT renderings and sublayouts, such as:

```
<sc:FieldRenderer FieldName="Text" runat="server" />
```

- Sitecore provides specialized controls for Date, Text, Link, and Image fields:

```
<sc:Text Field="Text" runat="server" />
```

- `sc:field` in XSLT renderings:

```
<xsl:value-of select="sc:field('Text')" />
```

- The Sitecore MVC helper:

```
@Html.Sitecore().Field("Text")
```

- Using the FieldRenderer.Render() method in code-behind:

```
FieldRenderer.Render("Text")
```

Sitecore MVC Tip
The `FieldRenderer.Render()` method returns a string. In Page Editor mode, this string includes the additional HTML to support inline editing. If you have a property in an MVC model that is being populated by `FieldRenderer.Render()`, ensure that you return a `HtmlString` – otherwise the Page Editor HTML is escaped and rendered as plain text when you try to use the property in your view, making it impossible to edit the value inline

If you render a field's raw value to the browser, that value is not editable in the Page Editor. This includes the following methods:

- `Sitecore.Context.Item.Fields["Text"].Value`
- `Sitecore.Context.Item["Text"]`
- `sc:fld("Text")`

2.1.1 How to Disable Editing

Sometimes, you may not want an editor to be able to edit a particular field inline. Good examples include the page title or meta description, neither of which are rendered to the page and are therefore not accessible. Rather than displaying the raw value of a field, you can disable inline editing for that particular instance of a Sitecore control or helper:

- When using a Sitecore control:

```
<sc:Text Field="Text" DisableWebEditing="true" runat="server" />
```

- When using the Sitecore MVC helper:

```
@Html.Sitecore().Field("Text", new { disable-web-editing=true })
```

- When using the `FieldRenderer.Render()` method directly:

```
FieldRenderer.Render("Text", "disable-web-editing=true")
```

The raw values for field types such as General Link, Rich Text, and Image contain custom XML. These cannot be displayed directly to the browser.

Chapter 3

Support Design

This chapter covers the fundamental requirements for supporting the ability to assemble a page from a library of components using the Page Editor's Designing mode.

The chapter contains the following sections:

3.1 Reasons to Support Design in the Page Editor

A Sitecore page is assembled from a variety of components. Some of these components display field values from the item that is currently being viewed – the context item – but many output field values from elsewhere in the content tree. For example, a homepage banner may be represented as a separate *Banner* item in a global location. Although it is displayed on the homepage, the actual content does not belong to the *Home* item.

Modular content and presentation details present editors with certain challenges when using the **Content Editor**:

- An item representing a page – such as the *Home* item – does not contain all of the content that a visitor sees on the homepage.
- An editor must look at the item's presentation details to find out where content is coming from and locate it in the content tree.
- If an editor wants to make changes to the appearance of an item, they must edit that item's presentation details directly.
- An item's presentation details gives no indication of how components are arranged.

By contrast, the **Page Editor** is a much more visual tool:

- Using the Page Editor to edit inline gives editors an instant preview of what their page will look like to visitors.
- The Page Editor makes it easier to configure component personalization and multivariate tests

By building to support the Page Editor, you are also building to support Sitecore's marketing and analytics features, as they depend on the same modular foundation.

Important
Support for the Page Editor must be taken into account at the start of your project, as it affects the way you architect and build your solution.

3.2 Break Up Your Pages

You should break page designs into smaller, re-usable components rather than designing individual page types. This lets editors create a variety of pages from smaller parts.

Note
You may want to specify a default set of components on the data template’s standard values. Later changes to standard values will affect all items based on that data template, and editors can reset to standard values if they want to restore the default appearance of a page.

The following example demonstrates how to break up a simple page design into components.

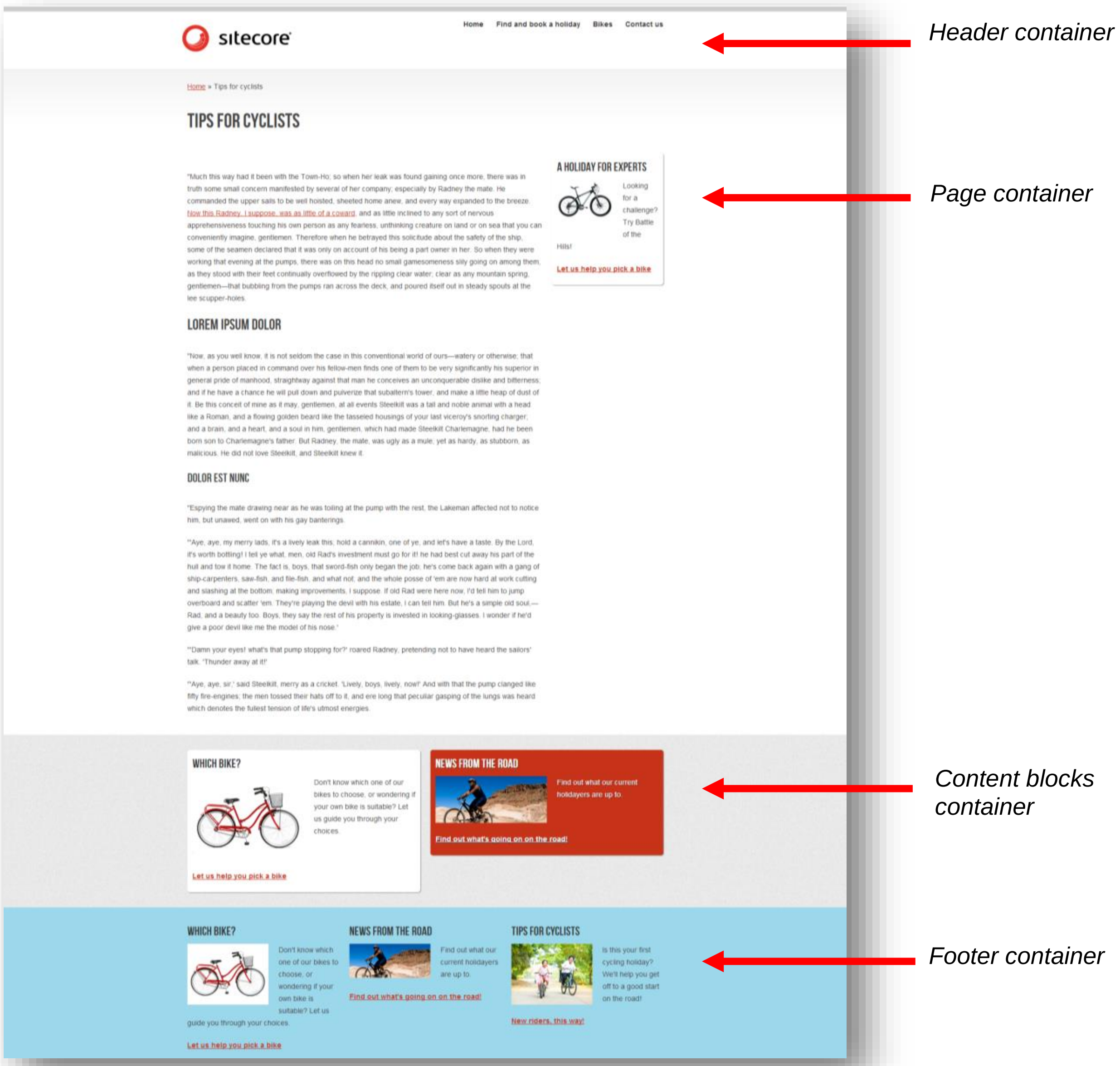
Note
You may want to limit flexibility in order to maintain design integrity and simplify the Page Editor experience for your editors. This can be done by having fewer page components or setting up restrictions within the Page Editor.

Breaking a page up into components lets you personalize and test individual parts of the page by varying the content or design based on specific visitor attributes.

3.2.1 Horizontal Containers

Our sample design consists of four main horizontal containers – the header, footer, page, and content block containers. These containers contain little more than styled `<div>` elements and a nested placeholder. Additional components are nested within these placeholders. It is likely that the header and footer will appear on every single page. Therefore, you might consider using static binding to include them in the main site layout.

Important
Statically bound components cannot be personalized or tested via the Page Editor. They are also not appropriate if future work requires a page to have a slightly different header or footer – for example, a checkout page might not have a standard breadcrumb.



Home » Tips for cyclists

TIPS FOR CYCLISTS

"Much this way had it been with the Town-Ho; so when her leak was found gaining once more, there was in truth some small concern manifested by several of her company; especially by Radney the mate. He commanded the upper sails to be well hoisted, sheeted home anew, and every way expanded to the breeze. Now this Radney, I suppose, was as little of a coward, and as little inclined to any sort of nervous apprehensiveness touching his own person as any fearless, unthinking creature on land or on sea that you can conveniently imagine, gentlemen. Therefore when he betrayed this solicitude about the safety of the ship, some of the seamen declared that it was only on account of his being a part owner in her. So when they were working that evening at the pumps, there was on this head no small gamesomeness slily going on among them, as they stood with their feet continually overflowed by the rippling clear water; clear as any mountain spring, gentlemen—that bubbling from the pumps ran across the deck, and poured itself out in steady spouts at the lee scupper-holes.

A HOLIDAY FOR EXPERTS

Looking for a challenge? Try Battle of the Hills!

Let us help you pick a bike

LOREM IPSUM DOLOR

"Now, as you well know, it is not seldom the case in this conventional world of ours—watery or otherwise; that when a person placed in command over his fellow-men finds one of them to be very significantly his superior in general pride of manhood, straightway against that man he conceives an unconquerable dislike and bitterness; and if he have a chance he will pull down and pulverize that subaltern's tower, and make a little heap of dust of it. Be this conceit of mine as it may, gentlemen, at all events Steelkilt was a tall and noble animal with a head like a Roman, and a flowing golden beard like the tasseled housings of your last viceroy's snorting charger; and a brain, and a heart, and a soul in him, gentlemen, which had made Steelkilt Charlemagne, had he been born son to Charlemagne's father. But Radney, the mate, was ugly as a mule; yet as hardy, as stubborn, as malicious. He did not love Steelkilt, and Steelkilt knew it.

DOLOR EST NUNC

"Espying the mate drawing near as he was toiling at the pump with the rest, the Lakeman affected not to notice him, but unawed, went on with his gay banterings.

"'Aye, aye, my merry lads, it's a lively leak this; hold a cannikin, one of ye, and let's have a taste. By the Lord, it's worth bottling! I tell ye what, men, old Rad's investment must go for it! he had best cut away his part of the hull and tow it home. The fact is, boys, that sword-fish only began the job; he's come back again with a gang of ship-carpenters, saw-fish, and file-fish, and what not; and the whole posse of 'em are now hard at work cutting and slashing at the bottom; making improvements, I suppose. If old Rad were here now, I'd tell him to jump overboard and scatter 'em. They're playing the devil with his estate, I can tell him. But he's a simple old soul,—Rad, and a beauty too. Boys, they say the rest of his property is invested in looking-glasses. I wonder if he'd give a poor devil like me the model of his nose.'

"'Damn your eyes! what's that pump stopping for?' roared Radney, pretending not to have heard the sailors' talk. 'Thunder away at it!'

"'Aye, aye, sir,' said Steelkilt, merry as a cricket. 'Lively, boys, lively, now!' And with that the pump clanged like fifty fire-engines; the men tossed their hats off to it, and ere long that peculiar gasping of the lungs was heard which denotes the fullest tension of life's utmost energies.

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

Let us help you pick a bike

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

Find out what's going on on the road!

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

Let us help you pick a bike

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

Find out what's going on on the road!

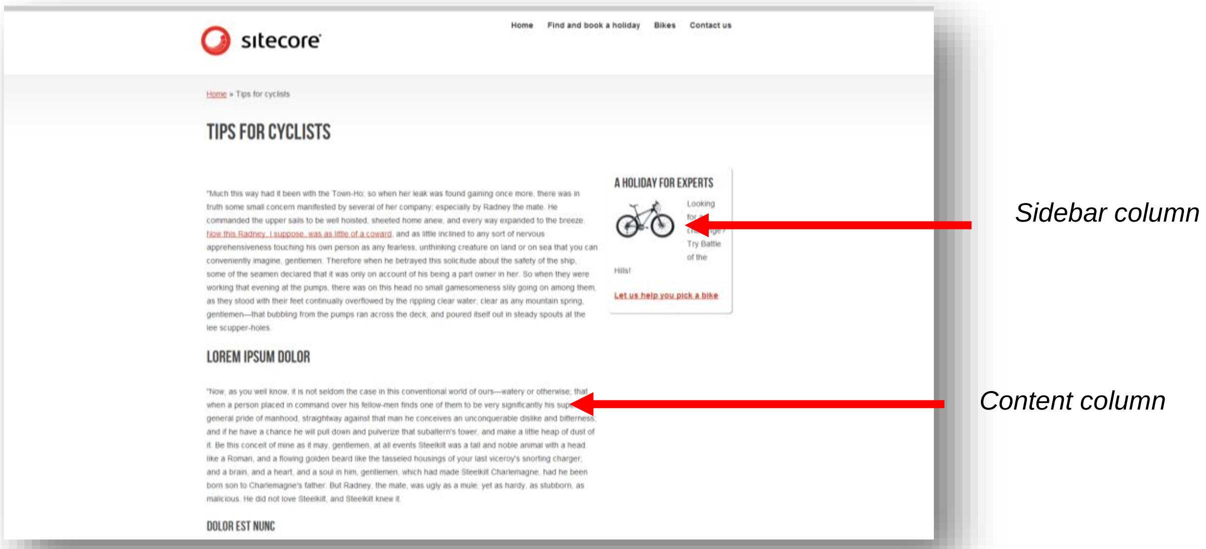
TIPS FOR CYCLISTS

Is this your first cycling holiday? We'll help you get off to a good start on the road!

New riders, this way!

3.2.2 Columns

Within containers, a page can be further subdivided into vertical columns. In the following example, a two-column grid is nested within the page container. Because future designs may introduce one and three column alternatives, the page container and the two-column grid are separate components.



TIPS FOR CYCLISTS

"Much this way had it been with the Town-Ho; so when her leak was found gaining once more, there was in truth some small concern manifested by several of her company; especially by Radney the mate. He commanded the upper sails to be well hoisted, sheeted home anew, and every way expanded to the breeze. Now this Radney, I suppose, was as little of a coward, and as little inclined to any sort of nervous apprehensiveness touching his own person as any fearless, unthinking creature on land or on sea that you can conveniently imagine, gentlemen. Therefore when he betrayed this solicitude about the safety of the ship, some of the seamen declared that it was only on account of his being a part owner in her. So when they were working that evening at the pumps, there was on this head no small gamesomeness slily going on among them, as they stood with their feet continually overflowed by the rippling clear water; clear as any mountain spring, gentlemen—that bubbling from the pumps ran across the deck, and poured itself out in steady spouts at the lee scupper-holes.

A HOLIDAY FOR EXPERTS

Looking for [illegible] Try Battle of the Hills!

LOREM IPSUM DOLOR

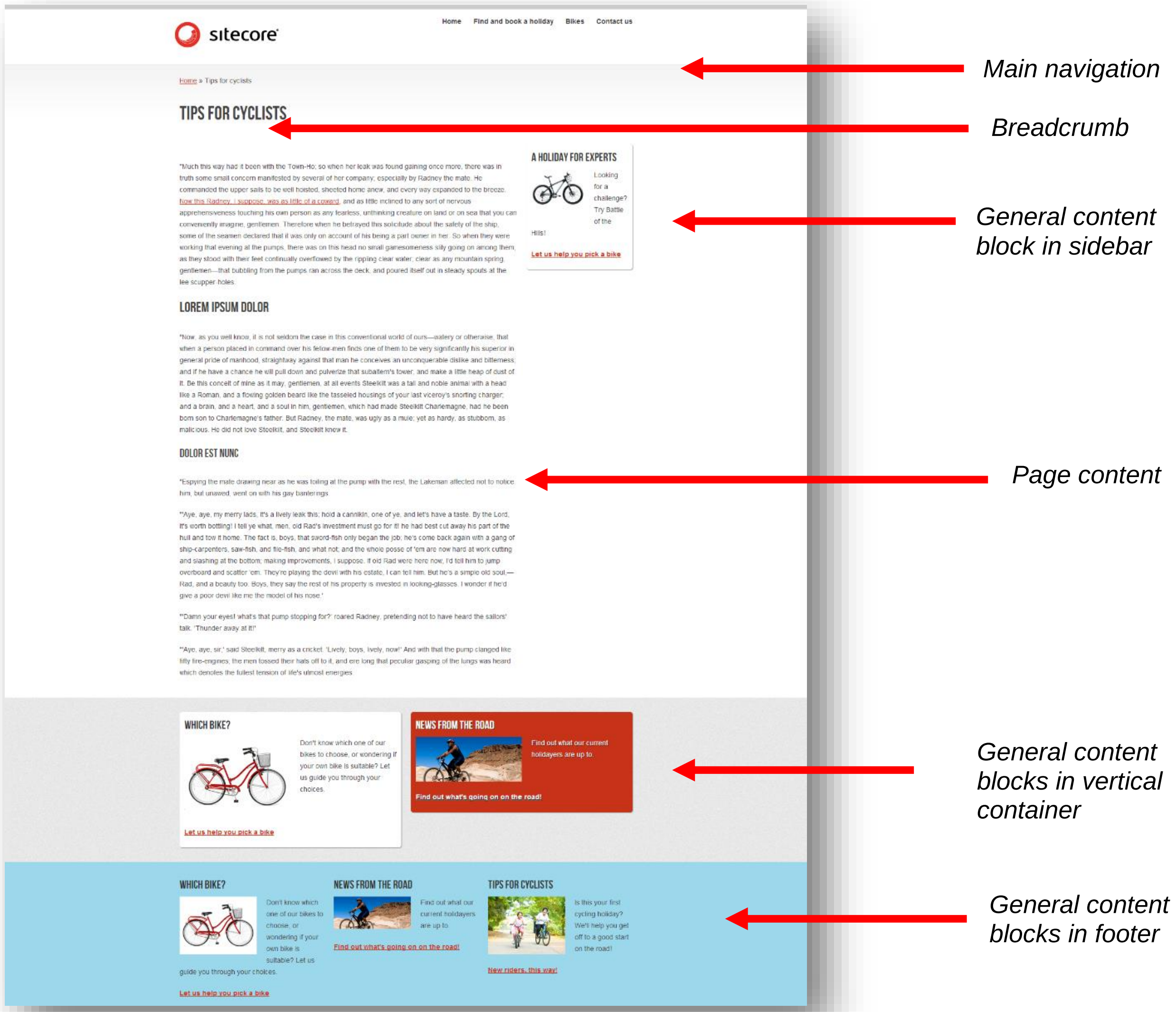
"Now, as you well know, it is not seldom the case in this conventional world of ours—watery or otherwise; that when a person placed in command over his fellow-men finds one of them to be very significantly his sup[illegible] general pride of manhood, straightway against that man he conceives an unconquerable dislike and bitterness; and if he have a chance he will pull down and pulverize that subaltern's tower, and make a little heap of dust of it. Be this conceit of mine as it may, gentlemen, at all events Steelkilt was a tall and noble animal with a head like a Roman, and a flowing golden beard like the tasseled housings of your last viceroy's snorting charger; and a brain, and a heart, and a soul in him, gentlemen, which had made Steelkilt Charlemagne, had he been born son to Charlemagne's father. But Radney, the mate, was ugly as a mule; yet as hardy, as stubborn, as malicious. He did not love Steelkilt, and Steelkilt knew it.

DOLOR EST NUNC

3.2.3 Functional Blocks

Containers and vertical grids contain functional blocks that generate Sitecore content. The following example uses four different types:

- Main navigation – this can be statically bound to the header container.
- Breadcrumb – this can be statically bound to the page container, unless certain pages do not require it.
- Page content – although content can be generated directly in the left-hand column, creating a separate page content component lets you reuse the two-column grid for other types of content, such as a form or a gallery.
- General content blocks – small chunks of content consisting of a title, text, image, and link. Although they have different designs and appear in different locations, functionally they are identical.


sitecore
Home
Find and book a holiday
Bikes
Contact us
Home » Tips for cyclists
Main navigation
Breadcrumb
General content block in sidebar
Page content
General content blocks in vertical container
General content blocks in footer
Let us help you pick a bike
Find out what's going on on the road!
New riders, this way!

TIPS FOR CYCLISTS

"Much this way had it been with the Town-Ho; so when her leak was found gaining once more, there was in truth some small concern manifested by several of her company; especially by Radney the mate. He commanded the upper sails to be well hoisted, sheeted home anew, and every way expanded to the breeze. Now this Radney, I suppose, was as little of a coward, and as little inclined to any sort of nervous apprehensiveness touching his own person as any fearless, unthinking creature on land or on sea that you can conveniently imagine, gentlemen. Therefore when he betrayed this solicitude about the safety of the ship, some of the seamen declared that it was only on account of his being a part owner in her. So when they were working that evening at the pumps, there was on this head no small gamesomeness slily going on among them, as they stood with their feet continually overflowed by the rippling clear water; clear as any mountain spring, gentlemen—that bubbling from the pumps ran across the deck, and poured itself out in steady spouts at the lee scupper-holes.

A HOLIDAY FOR EXPERTS

Looking for a challenge? Try Battle of the Hills!

LOREM IPSUM DOLOR

"Now, as you well know, it is not seldom the case in this conventional world of ours—watery or otherwise, that when a person placed in command over his fellow-men finds one of them to be very significantly his superior in general pride of manhood, straightway against that man he conceives an unconquerable dislike and bitterness; and if he have a chance he will pull down and pulverize that subaltern's tower, and make a little heap of dust of it. Be this conceit of mine as it may, gentlemen, at all events Steelkilt was a tall and noble animal with a head like a Roman, and a flowing golden beard like the tasseled housings of your last viceroy's snorting charger; and a brain, and a heart, and a soul in him, gentlemen, which had made Steelkilt Charlemagne, had he been born son to Charlemagne's father. But Radney, the mate, was ugly as a mule; yet as hardy, as stubborn, as malicious. He did not love Steelkilt, and Steelkilt knew it.

DOLOR EST NUNC

"Espying the mate drawing near as he was toiling at the pump with the rest, the Lakeman affected not to notice him, but unawed, went on with his gay banterings.

"'Aye, aye, my merry lads, it's a lively leak this; hold a cannikin, one of ye, and let's have a taste. By the Lord, it's worth bottling! I tell ye what, men, old Rad's investment must go for it! he had best cut away his part of the hull and tow it home. The fact is, boys, that sword-fish only began the job; he's come back again with a gang of ship-carpenters, saw-fish, and file-fish, and what not; and the whole posse of 'em are now hard at work cutting and slashing at the bottom; making improvements, I suppose. If old Rad were here now, I'd tell him to jump overboard and scatter 'em. They're playing the devil with his estate, I can tell him. But he's a simple old soul,—Rad, and a beauty too. Boys, they say the rest of his property is invested in looking-glasses. I wonder if he'd give a poor devil like me the model of his nose.'

"'Damn your eyes! what's that pump stopping for?' roared Radney, pretending not to have heard the sailors' talk. 'Thunder away at it!'

"'Aye, aye, sir,' said Steelkilt, merry as a cricket. 'Lively, boys, lively, now!' And with that the pump clanged like fifty fire-engines; the men tossed their hats off to it, and ere long that peculiar gasping of the lungs was heard which denotes the fullest tension of life's utmost energies.

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

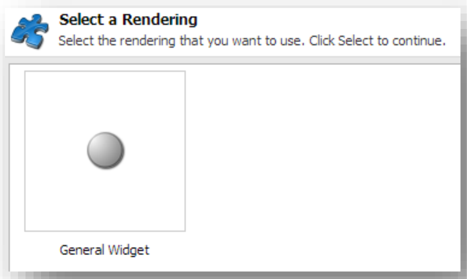
TIPS FOR CYCLISTS

Is this your first cycling holiday? We'll help you get off to a good start on the road!

3.3 Naming Conventions

3.3.1 Naming Components

When editors insert new components, they use the name of the component definition item as a way of identifying the components:


Select a Rendering
Select the rendering that you want to use. Click Select to continue.

Therefore, you should give your components easily identifiable and readable names.

Note
By default, Sitecore displays a large version of whatever icon you choose for the component definition item. It is recommended practice to choose a component thumbnail that gives editors a preview of the component they are inserting (see Component Previews)

3.3.2 Naming Data Templates and Fields

Data templates representing a page should not be named after the default appearance of that page. For example, it would be inappropriate to name a data template *Two Column Page* or *Three Column Page*, as you cannot assume that the data template will always be used for two or three column layouts respectively.

You should name fields after the data they contain rather than where that data might be displayed. For example, imagine that you have created a field called *Sidebar*. There are two problems with this name:

- You cannot guarantee that the data contained in this field will always appear in a sidebar
- *Sidebar* says nothing about the type of data this field contains

Consider what kind of information you want to display in a sidebar. For example, holiday price information belongs to a holiday item. In this case, you should create a field called *Price Information*. If the information is related to a holiday but not part of it – such as a related holiday promotion – that information should be pulled in from elsewhere in the content tree using data sources.

3.4 Create Placeholder Settings Items

A page is assembled by binding components to placeholders. Placeholders are Sitecore controls that specify a unique key. In web forms, they are represented by a control:

```
        <sc:Placeholder Key="main" runat="server" />
```

In Sitecore MVC, placeholders are defined using the Sitecore HTML helper:

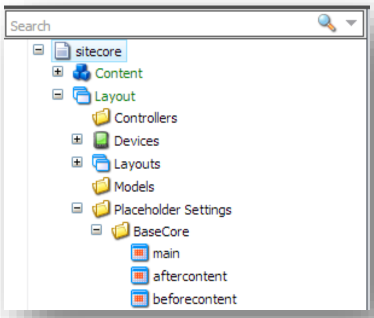
```
        @Html.Sitecore().Placeholder("main")
```

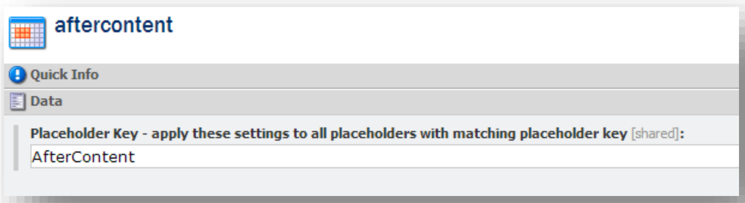
In order to make a placeholder visible in the Page Editor, you must create a placeholder settings item. If you want placeholders to be visible irrespective of whether a placeholder settings item has been created, change the following setting to `true` in the `web.config`:

```
        <setting name="WebEdit.PlaceholdersEditableWithoutSettings" value="false" />
```

Note
If you do not create a placeholder settings item, you will still be able to bind components to placeholders using the Presentation Details dialog.

Placeholder settings items are located under `/sitecore/Layout/Placeholder Settings`




aftercontent

Quick Info

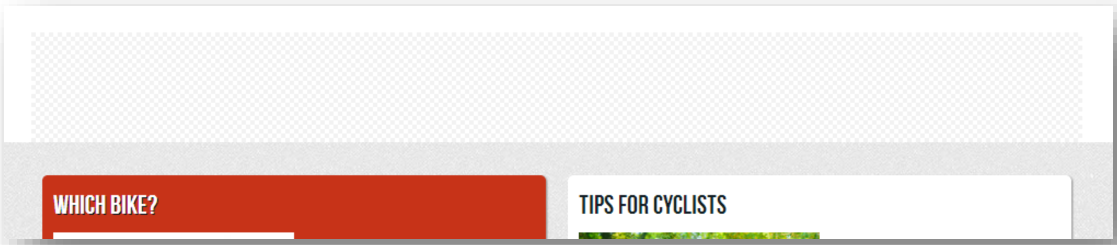
Data

Placeholder Key - apply these settings to all placeholders with matching placeholder key [shared]:

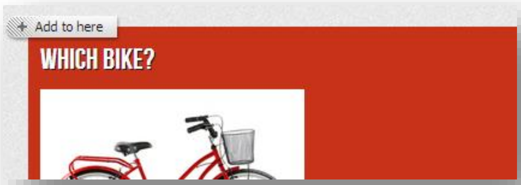
AfterContent

Note
The **Placeholder Key** refers to the Key attribute on the placeholder control. This field is not case sensitive, and it does not matter if your placeholder setting item’s name and the key it represents do not use the same case.

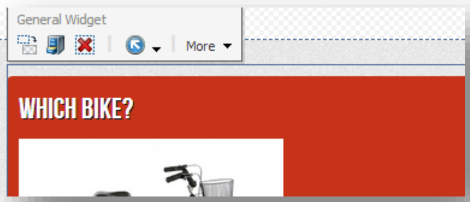
Once the placeholder settings item has been created, placeholders will appear as boxes with a checkered background in the Page Editor:



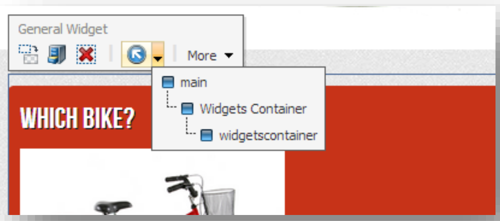
If a placeholder already contains components, **Add to here** controls will appear when you click the **Component** button in the ribbon:



You can move components between placeholders:



It is good practice to give your placeholder settings items the same name as the key they represent. Using lowercase and removing spaces makes it easier for editors to distinguish placeholders from components in the presentation hierarchy. In the following example, **main** and **widgetscontainer** are placeholders:

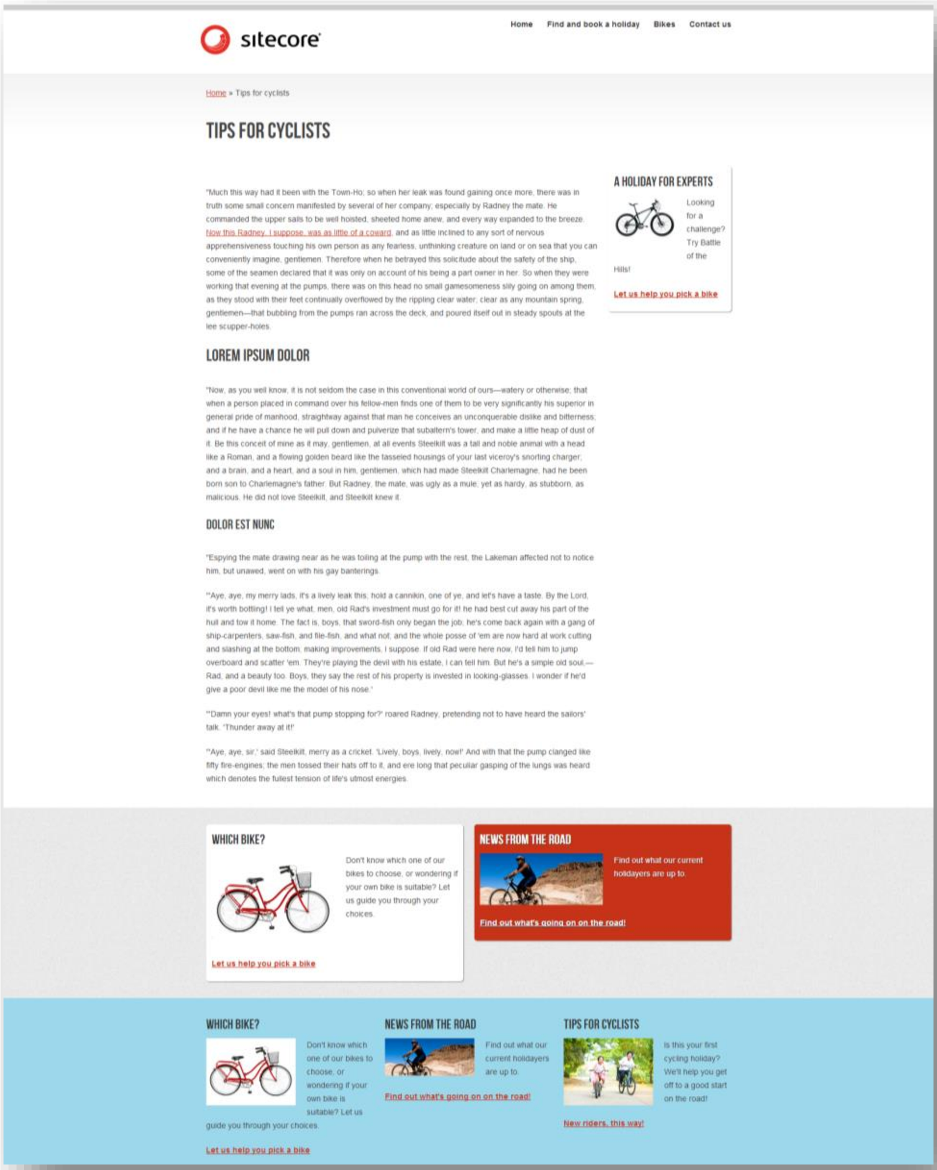


Note
By default, placeholders will not show up in the Page Editor without a corresponding placeholder settings item. However, you can override this behavior by changing the following setting to `true` in the `web.config`:

```
<setting name="WebEdit.PlaceholdersEditableWithoutSettings" value="false"
/>
```

3.5 Use Data Sources

Not all data displayed on a page belongs exclusively to that page. Content, like components, can be reused across multiple pages. In the following example, only the title – Tips for cyclists – and the main content belongs to the context item. Each of the General content blocks is displaying data that is pulled from somewhere else in the content tree. You can do this by specifying a different data source each time the component is used.



Home Find and book a holiday Bikes Contact us

Home » Tips for cyclists

TIPS FOR CYCLISTS

"Much this way had it been with the Town-Ho; so when her leak was found gaining once more, there was in truth some small concern manifested by several of her company; especially by Radney the mate. He commanded the upper sails to be well hoisted, sheeted home anew, and every way expanded to the breeze. Now this Radney, I suppose, was as little of a coward, and as little inclined to any sort of nervous apprehensiveness touching his own person as any fearless, unthinking creature on land or on sea that you can conveniently imagine, gentlemen. Therefore when he betrayed this solicitude about the safety of the ship, some of the seamen declared that it was only on account of his being a part owner in her. So when they were working that evening at the pumps, there was on this head no small gamesomeness slily going on among them, as they stood with their feet continually overflowed by the rippling clear water; clear as any mountain spring, gentlemen—that bubbling from the pumps ran across the deck, and poured itself out in steady spouts at the lee scupper-holes.

A HOLIDAY FOR EXPERTS

Looking for a challenge? Try Battle of the Hills!

Let us help you pick a bike

LOREM IPSUM DOLOR

"Now, as you well know, it is not seldom the case in this conventional world of ours—watery or otherwise; that when a person placed in command over his fellow-men finds one of them to be very significantly his superior in general pride of manhood, straightway against that man he conceives an unconquerable dislike and bitterness; and if he have a chance he will pull down and pulverize that subaltern's tower, and make a little heap of dust of it. Be this conceit of mine as it may, gentlemen, at all events Steelkilt was a tall and noble animal with a head like a Roman, and a flowing golden beard like the tasseled housings of your last viceroy's snorting charger; and a brain, and a heart, and a soul in him, gentlemen, which had made Steelkilt Charlemagne, had he been born son to Charlemagne's father. But Radney, the mate, was ugly as a mule; yet as hardy, as stubborn, as malicious. He did not love Steelkilt, and Steelkilt knew it.

DOLOR EST NUNC

"Espying the mate drawing near as he was toiling at the pump with the rest, the Lakeman affected not to notice him, but unawed, went on with his gay banterings.

"'Aye, aye, my merry lads, it's a lively leak this; hold a cannikin, one of ye, and let's have a taste. By the Lord, it's worth bottling! I tell ye what, men, old Rad's investment must go for it! he had best cut away his part of the hull and tow it home. The fact is, boys, that sword-fish only began the job; he's come back again with a gang of ship-carpenters, saw-fish, and file-fish, and what not; and the whole posse of 'em are now hard at work cutting and slashing at the bottom; making improvements, I suppose. If old Rad were here now, I'd tell him to jump overboard and scatter 'em. They're playing the devil with his estate, I can tell him. But he's a simple old soul,—Rad, and a beauty too. Boys, they say the rest of his property is invested in looking-glasses. I wonder if he'd give a poor devil like me the model of his nose.'

"'Damn your eyes! what's that pump stopping for?' roared Radney, pretending not to have heard the sailors' talk. 'Thunder away at it!'

"'Aye, aye, sir,' said Steelkilt, merry as a cricket. 'Lively, boys, lively, now!' And with that the pump clanged like fifty fire-engines; the men tossed their hats off to it, and ere long that peculiar gasping of the lungs was heard which denotes the fullest tension of life's utmost energies.

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

Let us help you pick a bike

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

Find out what's going on on the road!

WHICH BIKE?

Don't know which one of our bikes to choose, or wondering if your own bike is suitable? Let us guide you through your choices.

Let us help you pick a bike

NEWS FROM THE ROAD

Find out what our current holidayers are up to.

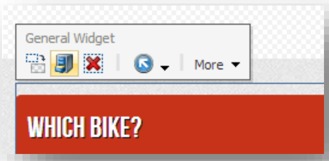
Find out what's going on on the road!

TIPS FOR CYCLISTS

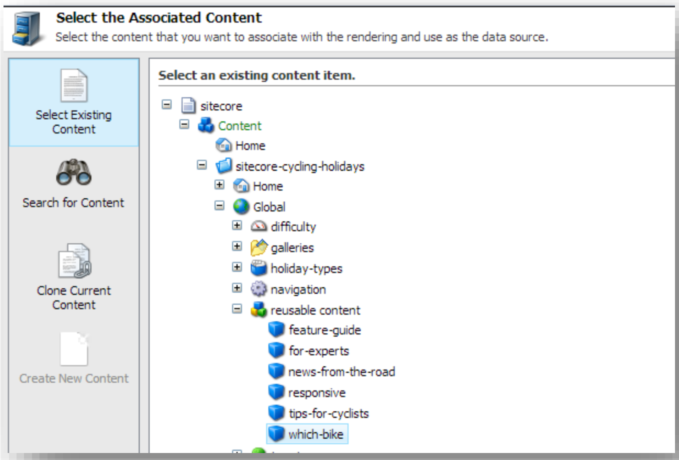
Is this your first cycling holiday? We'll help you get off to a good start on the road!

New riders, this way!

Editors can vary the data source in the Page Editor by selecting the component and clicking the **Set associated content** button:

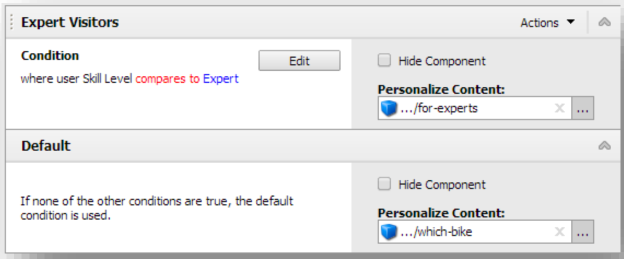


They can then choose an item to act as a data source:



Important
Any item in the Sitecore tree can act as a data source. It can be an item that represents a page, or an item that only represents a small chunk of reusable content. In the example above, the chosen data source is an item that lives outside the main site tree, and does not have presentation details of its own.

It is important to note that component personalization and testing rely in part on being able to vary a component's data source. In the following example, the component is set up to vary its data source from **which-bike** to **for-experts** if a certain condition is met.


Expert Visitors

Condition: where user Skill Level compares to Expert

Hide Component

Personalize Content: .../for-experts

Default

If none of the other conditions are true, the default condition is used.

Hide Component

Personalize Content: .../which-bike

Similarly, testing lets you present different data sources to different users and measure which content resulted in the most effective visits.

3.5.1 Coding for Datasources

By default, controls and helpers that output content target the context item. Each of the following examples look to `Sitecore.Context.Item` to find the **Text** field rather than the item specified on their parent rendering.

Sublayouts and XSLT:

```
<sc:Text Field="Text" runat="server" />
```

MVC renderings:

```
@Html.Sitecore().Field("Text")
```

In both cases, the request ultimately goes through the `FieldRenderer` pipeline:

```
FieldRenderer.Render("Text")
```

In order to target the data source item, you must pass the item through to the `FieldRenderer` pipeline from your control or helper.

3.5.2 Retrieving the Datasource in a Sublayout

In a sublayout's code-behind, retrieve the data source item from the `Sublayout` class by casting `this.Parent` as a `Sublayout`:

```
string datasource = String.Empty;

if (this.Parent is Sublayout)
{
    datasource = ((Sublayout)Parent).DataSource;
}
```

The `datasource` object is the item GUID in string form. Be advised that if no data source has been set, this property will return an empty string. Use this string to set the control's `DataSource` property:

```
<sc:Text Field="Text" ID="TextField" runat="server" />
```

Target the control by its ID as you would with any other ASP.NET control:

```
string datasource = String.Empty;

if (this.Parent is Sublayout)
{
    datasource = ((Sublayout)Parent).DataSource;

    if (!String.IsNullOrEmpty(datasource))
    {
       TextField.DataSource = datasource;
    }
}
```

Alternatively, convert it into a Sitecore `ID,` retrieve the actual Sitecore `Item` and set the control's `Item` property:

```
string datasource = String.Empty;

if (this.Parent is Sublayout)
{
    datasource = ((Sublayout)Parent).DataSource;

    if (!String.IsNullOrEmpty(datasource))
    {
       Item item = Sitecore.Context.Database.GetItem(new ID(datasource));
       TextField.Item = item;
    }
}
```

You can also retrieve the data source ID as a string using the `Attributes` collection:

```
        string datasourceID = Attributes["sc_datasource"];

        if (!String.IsNullOrEmpty(datasourceID))
        {
           Item data sourceItem = Sitecore.Context.Database.GetItem(new ID(datasourceID));
        }
```

Tip
You can return a collection of items based on a query rather than a selection. For more information, see the development team's blog post:
http://www.sitecore.net/Community/Technical-Blogs/Sitecore-7-Development-Team/Posts/2013/04/Sitecore-7-Datasources.aspx

3.5.3 Retrieving the Datasource in XSLT

The `$sc_item` variable returns the data source item if one has been set. Otherwise, it returns the context item:

```
        <sc:Text Field="Text" Select="$sc_item" />
```

3.5.4 Retrieving the Datasource in a View Rendering

When you use Sitecore's default `RenderingModel`, the data source is available as a property.

```
        @Model.Item
```

The following will also work:

```
        @Model.Rendering.Item
```

Pass this object into the field helper:

```
        @Html.Sitecore().Field("Text", Model.Item)
```

3.5.5 Retrieving the Datasource in a Controller Rendering

To access the data source item within a controller rendering action, use the `RenderingContext`:

```
        RenderingContext.Current.Rendering.Item
```

Note
Do not use RenderingContext directly in a view – ASP.NET MVC recommended practice dictates that all view data is passed to the view via a model or view model object.

3.6 Page Editor Security

3.6.1 Essential Roles

In order to design pages using the Page Editor, editors must belong to the group **sitecore\Sitecore Client Designing**. The **sitecore\Designer** group combines this role with **sitecore\Sitecore Client Users**. Editors with this permission can use Editing and Designing modes.

Standard read/write rules apply. Editors can modify the same content in the Page Editor as they can in the Content Editor. Editors who are not in the **sitecore\Sitecore Client Designing** group can still use the Page Editor to enter content, but cannot add or remove components or change their properties.

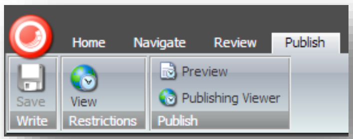
3.6.2 Disabling Page Editor Access

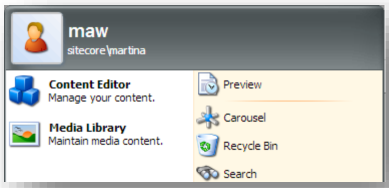
If you do not want editors to be able to access the Page Editor at all, you can restrict access to the Page Editor buttons in the **core** database as you would with any other item.

By default, there are Page Editor buttons in the following locations:

- `/sitecore/content/Documents` and `settings/All users/Start menu/Right/Page Editor`
- `/sitecore/content/Applications/Content Editor/Ribbons/Chunks/Publish/Page Editor`

If Read access is denied on these items, they do not appear in the ribbon or Desktop menu:



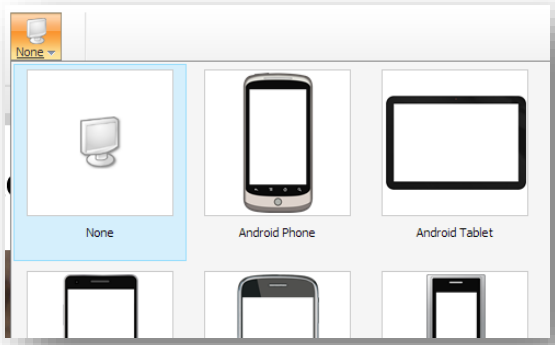


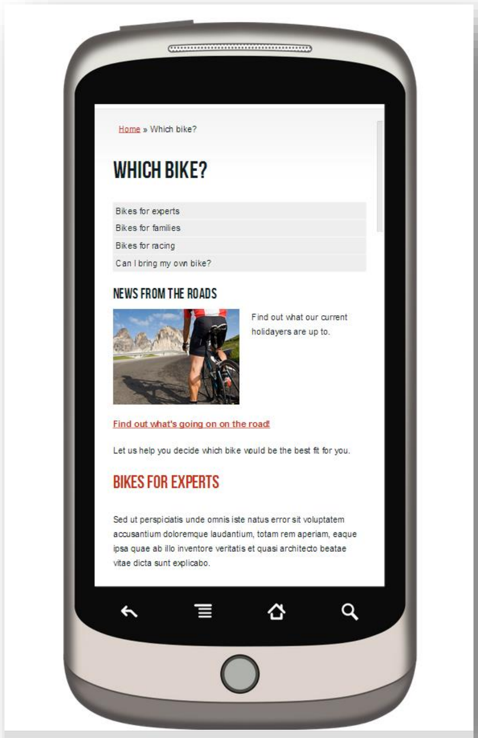
To remove the Page Editor button from the Sitecore login page, edit the link out of `<root>\Website\sitecore\login.aspx`

3.7 Responsive Design

If your site is responsive, whatever combination of components an editor uses to build a page on a Desktop machine must also work on a smaller device.

- Ensure that your placeholder restrictions account for smaller devices. Will a particular widget still work in a particular location on a smaller screen?
- Suggest that editors use the **Device Simulator** in the **Experience** tab in **Preview** mode. This will give them an idea of what their page will look like on a smaller screen:




Home » Which bike?
WHICH BIKE?
Bikes for experts
Bikes for families
Bikes for racing
Can I bring my own bike?
NEWS FROM THE ROADS
Find out what our current holidayers are up to.
Find out what's going on on the road!
Let us help you decide which bike would be the best fit for you.
BIKES FOR EXPERTS
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Note
The Page Editor interface is not currently optimized for mobile or touch screens.

Chapter 4

Configuring the Page Editor

The following chapter covers ways in which the Page Editor can be configured to improve the user experience for Sitecore editors.

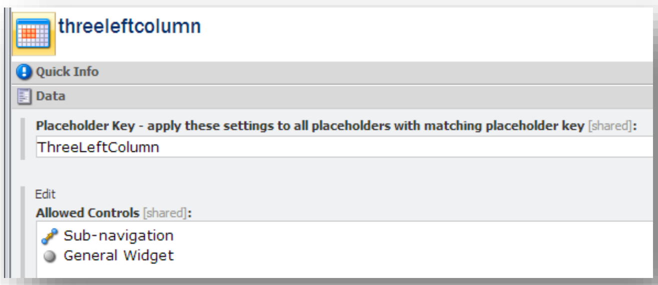
The chapter contains the following sections:

- Allowed Controls
- Locking Placeholders
- Placeholder Settings Overrides
- Editable Components
- Datasource Location and Template
- Compatible Renderings
- Rich Text Editor Field Buttons
- Component Previews

4.1 Allowed Controls

Placeholder settings items allow you to specify a list of components that can be added to a particular placeholder.

Note
These restrictions only apply to the Page Editor – anyone that can access the item's presentation details can still add any component to any placeholder.



threeleftcolumn

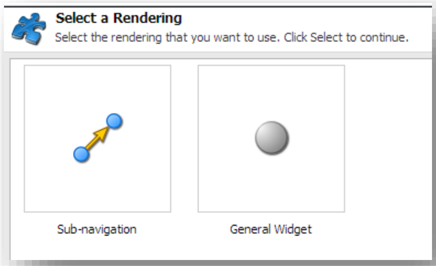
Quick Info

Data

Placeholder Key - apply these settings to all placeholders with matching placeholder key [shared]:
ThreeLeftColumn

Edit
Allowed Controls [shared]:
Sub-navigation
General Widget

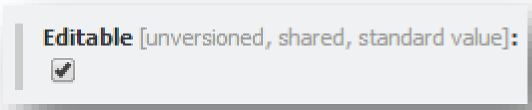
When you specify a list of allowed components, editors only see the components that are guaranteed to work in that particular placeholder:



Select a Rendering
Select the rendering that you want to use. Click Select to continue.

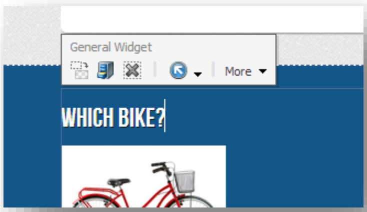
4.2 Locking Placeholders

In some cases, you may want to allow an editor to change a component's data source without allowing them to delete the component or move it out of the particular placeholder. The **Editable** checkbox on the placeholder settings item prevents editors from being able to add or remove components:



Editable [unversioned, shared, standard value]:
☑

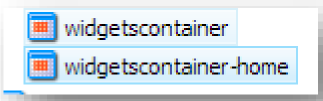
Add to here labels no longer appear, and the **Move** and **Delete** icons are greyed out:



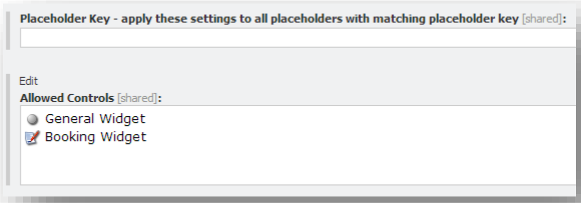
4.3 Placeholder Settings Overrides

A placeholder is likely to be used in more than one context. For example, the **WidgetContainer** placeholder may appear on the homepage as well as the holidays page. However, you may want to specify different settings for the same placeholder depending on where it appears. You can do this by creating a placeholder settings override for that placeholder. You can apply a placeholder settings override to a particular item, or a data template's standard values using the following procedure:

1. Create a regular placeholder settings item with a similar name to the one you want to override:

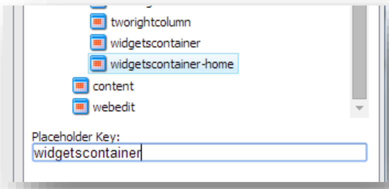


2. Make sure that the **Placeholder Key** field is blank, and set up any restrictions that you require:


Placeholder Key - apply these settings to all placeholders with matching placeholder key [shared]:

Edit
Allowed Controls [shared]:
General Widget
Booking Widget

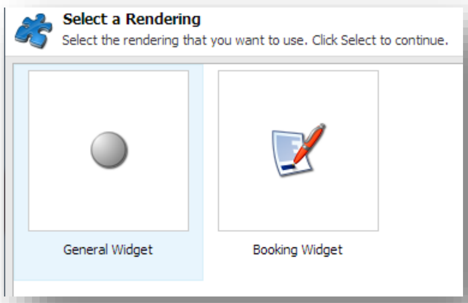
3. Select the item or standard values that you want to apply the override to.
4. Open the presentation details dialog and click **Edit**.
5. In the left menu, click **Placeholder Settings**.
6. Click **Add** to create a new override
7. Choose the placeholder settings item you just created, and map it to an existing placeholder key:



Placeholder Key: widgetscontainer

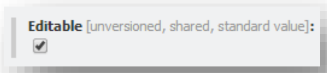
8. Save the item

Wherever the placeholder key appears in the presentation details of the particular item (or item type, if the override was configured on the data template standard values), its settings are determined by the placeholder settings item override. In the following example, the override allows any instance of the **WidgetsContainer** placeholder key on the homepage to accept the **Booking Widget** component as well as the **General Widget** component.


Select a Rendering
Select the rendering that you want to use. Click Select to continue.
General Widget
Booking Widget

4.4 Editable Components

Like placeholder settings items, components also have an **Editable** checkbox:


Editable [unversioned, shared, standard value]:
☑

If you clear this checkbox:

- Editors cannot add this component to **any** placeholder, even if the component has been specified as an allowed control.
- Editors cannot select the component in the Page Editor.

The fields displayed by the component remain editable unless configured not to be. This setting is particularly useful if you want the placement of a particular component, such as a booking form, only to be controlled by developers.

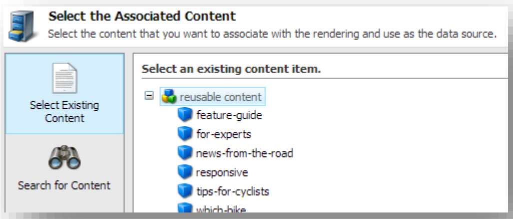
4.5 Datasource Location and Template

Many components require a data source to be set in order to function. By default, Sitecore requires editors to click the **Set associated content** button in the component toolbar and choose an item. This has a number of disadvantages:

- It requires several clicks.
- Editors must locate an item of the correct type in a large content tree.
- Editors are not prompted to select a data source, and may forget to do so.

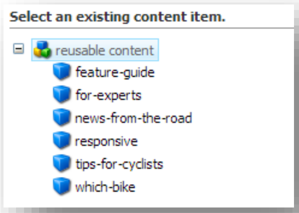
Specifying a data source template and the location for the component has the following advantages:

- Setting a data source location restricts the editor to a particular area of the content tree, even if it is just `/sitecore/Content item`.

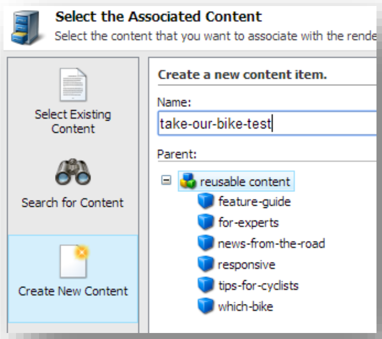

Select the Associated Content
Select the content that you want to associate with the rendering and use as the data source.

Select an existing content item.
reusable content
feature-guide
for-experts
news-from-the-road
responsive
tips-for-cyclists
which-bike

- Setting a data source location forces the editor to pick a data source when inserting a new component. The **Select the Associated Content** dialog pops up immediately.
- Setting a data source template restricts the type of item an editor can pick as a data source. Item types that are unavailable are greyed out.

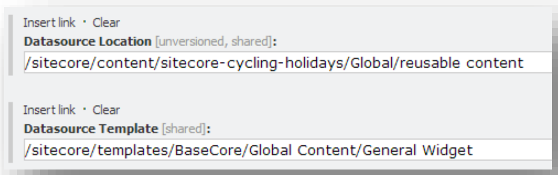

Select an existing content item.
reusable content
feature-guide
for-experts
news-from-the-road
responsive
tips-for-cyclists
which-bike

- Setting both the data source location and data source template on a component enables the **Create New Content** button. This enables authors to create a new data source item of the correct type and in the correct location without having to open the Content Editor.


Select the Associated Content
Select the content that you want to associate with the rende

Create a new content item.
Name:
take-our-bike-test
Parent:
reusable content
feature-guide
for-experts
news-from-the-road
responsive
tips-for-cyclists
which-bike

The **Datasource Location** and **Datasource Template** fields are located on the component definition item:



Datasource Location [unversioned, shared]:

/sitecore/content/sitecore-cycling-holidays/Global/reusable content

Datasource Template [shared]:

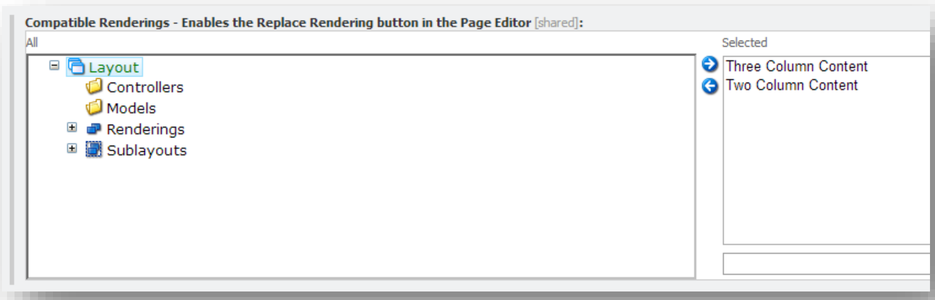
/sitecore/templates/BaseCore/Global Content/General Widget

4.6 Compatible Renderings

Certain components, such as two column and three column containers, are interchangeable. This means that they:

- Perform a very similar function, but may have a slightly different design.
- Accept the same data source. This is not a requirement, but compatibility suggests that content does not need to change when the component data source changes.
- Accept the same rendering parameters. This is not a requirement, but compatibility suggests that the rendering parameter selections are also compatible.

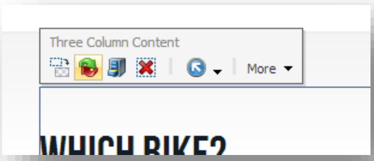
Sitecore enables you to configure a list of components that are compatible with the currently selected component. You can do this by using the **Compatible Renderings** multilist on the component definition item:



Compatible Renderings - Enables the Replace Rendering button in the Page Editor [shared]:

Note
In order to be able to switch back and forth between compatible renderings, each component in the list must list the others in their own Compatible Renderings field. If they do not, you may not be able to switch back to the component that was initially selected.

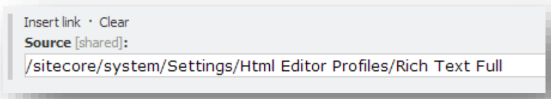
Components with compatible renderings have a button marked **Replace with another component** included in the toolbar:



Important
Components listed as compatible must also be in the **Allowed Controls** list for the same placeholders – otherwise they will not appear in the selection list.

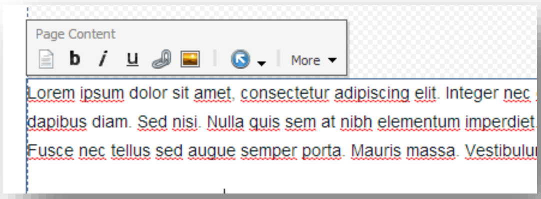
4.7 Rich Text Editor Field Buttons

Rich text editor fields can be customized using **Rich text editor profiles** (or RTE profiles). RTE profiles are folders of buttons and toolbars that are set up on the core database. To associate a particular field with a profile, set its source to the profile path:



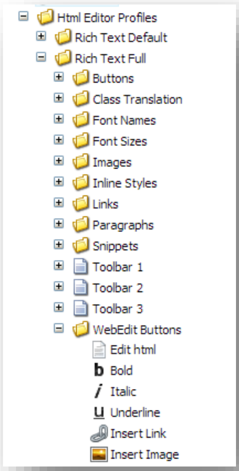
Insert link · Clear
Source [shared]:
/sitecore/system/Settings/Html Editor Profiles/Rich Text Full

Generally, developers duplicate an existing RTE profile, rename it – for example, Sample Rich Text Default – and add/remove buttons and toolbars as required. All RTE profiles are stored under `/sitecore/system/Settings/Html Editor Profiles` in the core database. When viewed in the Page Editor, Rich Text Editor fields have a number of type-specific buttons in the toolbar:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec
dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet
Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulu

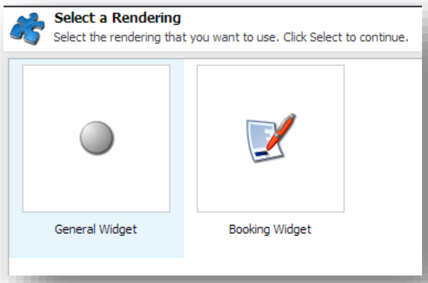
These buttons are also controlled by the RTE profile associated with the field, and come from a folder called *Webedit Buttons*:



To restrict editor creativity in this particular instance, consider removing the **Edit html** button. This button opens up a separate rich text editor dialog with a full set of options that you may not want to make available in the Page Editor.

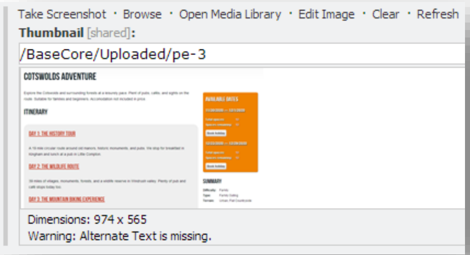
4.8 Component Previews

The **Select a Rendering** dialog, which appears when you insert a new component, displays a larger version of the component item's icon by default:


Select a Rendering
Select the rendering that you want to use. Click Select to continue.
General Widget
Booking Widget

This does not tell the editor much about what the component is going to look like. To give them a useful preview, take a screenshot of the component and upload it to the Thumbnail field in the component.

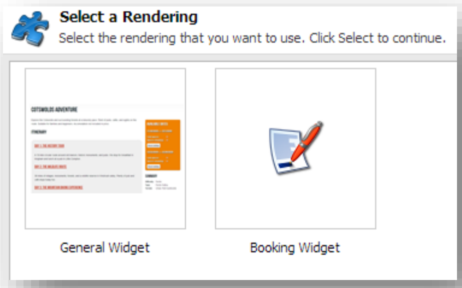
Note
To access the **Thumbnail** field, you need to enable **Standard Fields** in the Content Editor.


Take Screenshot · Browse · Open Media Library · Edit Image · Clear · Refresh
Thumbnail [shared]:

/BaseCore/Uploaded/pe-3

Dimensions: 974 x 565
Warning: Alternate Text is missing.

The **Select a Rendering** dialog now displays a preview of the component:


Select a Rendering
Select the rendering that you want to use. Click Select to continue.
General Widget
Booking Widget

Chapter 5

Extending the Page Editor

The following chapter covers ways in which the Page Editor can be extended using custom buttons and toolbars.

The chapter contains the following sections:

- Custom Experience Buttons
- Edit Frames
- Customizing the Ribbon

5.1 Custom Experience Buttons

The floating toolbars that surround fields and components can be modified to include additional custom experience buttons. These buttons can be used to extend and improve the Page Editor experience by reducing trips to the Content Editor and Desktop, and adding functionality.

- All custom experience buttons are defined in the core database, under `/sitecore/content/Applications/WebEdit/Custom Experience Buttons`.
- Once created and configured, custom experience buttons can be assigned to field definition items or component definition items in the master database.
- The assigned custom experience button appears in the floating toolbar wherever the field or component is displayed in the Page Editor.

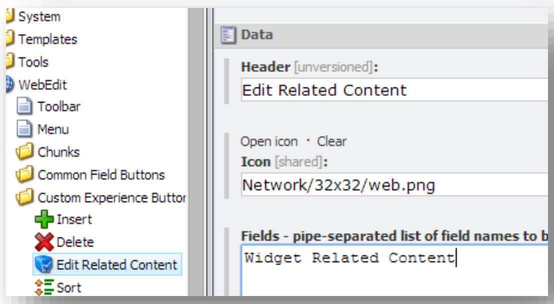
Note
Custom experience buttons assigned to a component are only visible in **Design mode**.

5.1.1 Field Editor Buttons

Field Editor Buttons are the most commonly used type of custom experience button. They are used to expose additional fields in a dialog. This is particularly useful when:

- Fields are not visible on the page, such as metadata or styles.
- The type of the field is impossible to edit inline, such as multilists or treelists.

In the following example, the **Edit Related Content** field editor button has been configured to open the **Widget Related Content** field in a dialog:



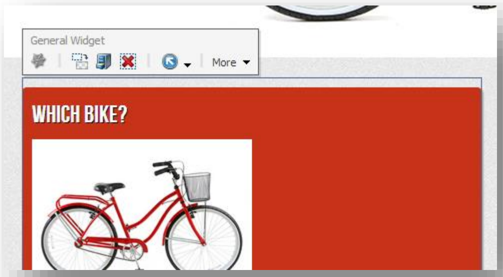
Header [unversioned]:
Edit Related Content

Icon [shared]:
Network/32x32/web.png

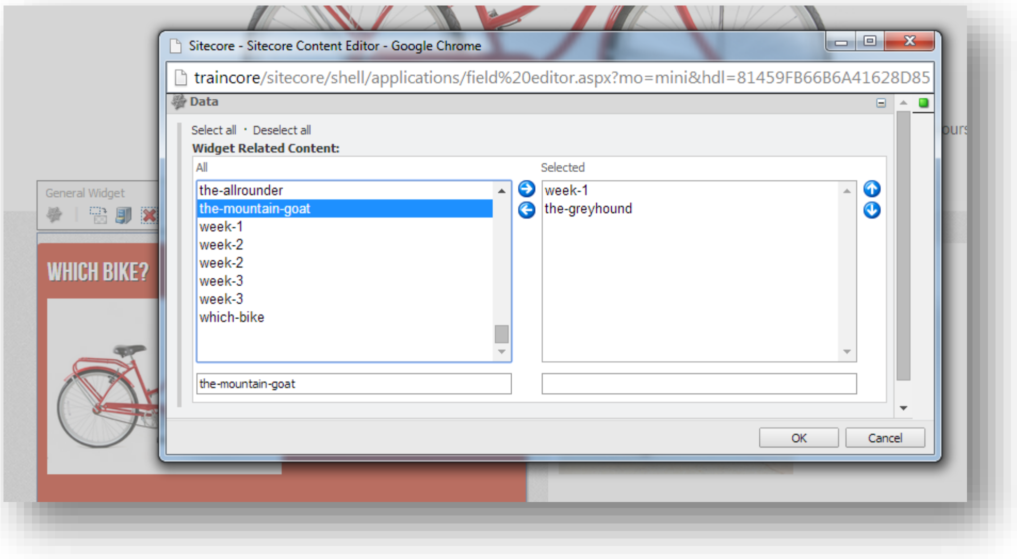
Fields - pipe-separated list of field names to b
Widget Related Content

Note
You can include any number of fields in this list, separated by a | (pipe) character.

As a result, the new button appears in the General Widget floating toolbar:



Clicking the button opens a dialog with the specified field.



Sitecore - Sitecore Content Editor - Google Chrome

traincore/sitecore/shell/applications/field%20editor.aspx?mo=mini&hdl=81459FB66B6A41628D85

Data

Select all · Deselect all

Widget Related Content:

All: the-allrounder, the-mountain-goat, week-1, week-2, week-2, week-3, week-3, which-bike

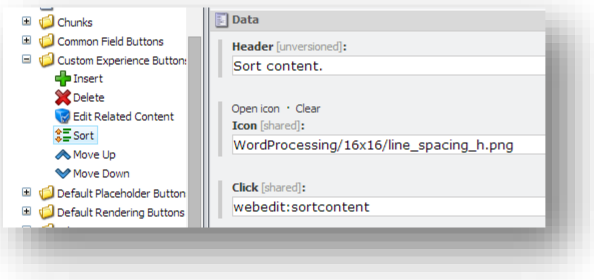
Selected: week-1, the-greyhound

the-mountain-goat

Important
Field editor buttons are sensitive to the context data source. In the previous example, the **Widget Related Content** field comes from the data source selected on the General Widget component in the background.

5.1.2 Webedit Buttons

WebEdit Buttons require programming and can be used to trigger custom behavior. Unlike field editor buttons, which specifically open a set of fields, webedit buttons trigger a **command**. In the following example, the **webedit:sortcontent** command is being triggered



Header [unversioned]: Sort content.

Icon [shared]: WordProcessing/16x16/line_spacing_h.png

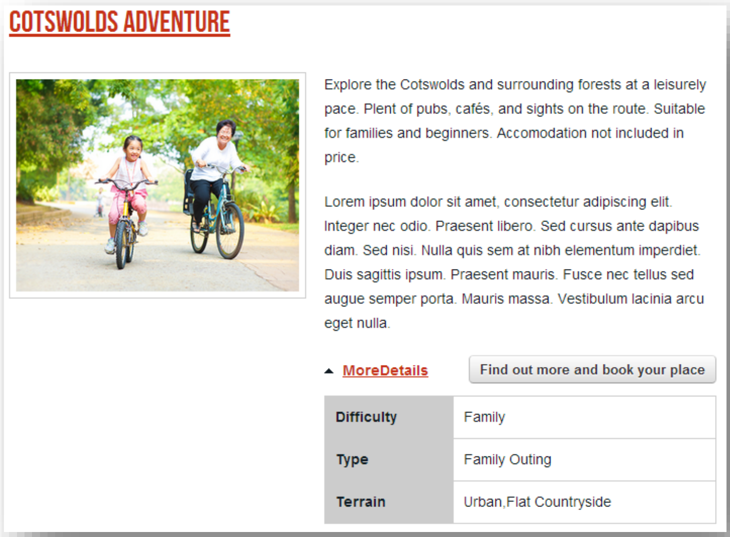
Click [shared]: webedit:sortcontent

Examples of how you might use webedit buttons include:

- Adding a spellcheck button to rich text fields
- Move components up or down
- Publishing the component's data source

5.2 Edit Frames

In some instances, you may want to add custom experience to a particular part of a component or page. The following component contains a table that outputs a number of lists. These are stored in Sitecore as multilist fields, and the best way to expose this field is to use a field editor button (see Field Editor Buttons).

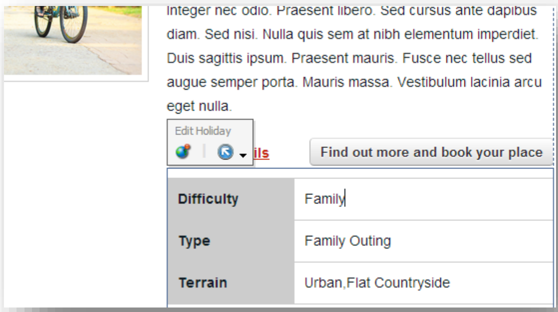

COTSWOLDS ADVENTURE

Explore the Cotswolds and surrounding forests at a leisurely pace. Plent of pubs, cafés, and sights on the route. Suitable for families and beginners. Accomodation not included in price.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

Difficulty	Family
Type	Family Outing
Terrain	Urban,Flat Countryside

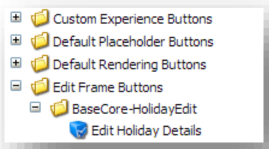
However, because the component is large, you should place the field editor button close to the content it modifies. Edit frames enable you to surround particular areas of your markup with a floating toolbar. This floating toolbar looks and functions exactly like the ones that surround Sitecore components and fields, and it can be used to display a list of custom experience buttons exactly like the ones that you can add to components and fields. The following example shows how to surround a table with an edit frame that makes an ‘Edit Holiday’ field editor button available:


Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

Difficulty	Family
Type	Family Outing
Terrain	Urban,Flat Countryside

To create an edit frame around each search result, surround the search result markup with an `<sc:EditFrame />` control and specify two things:

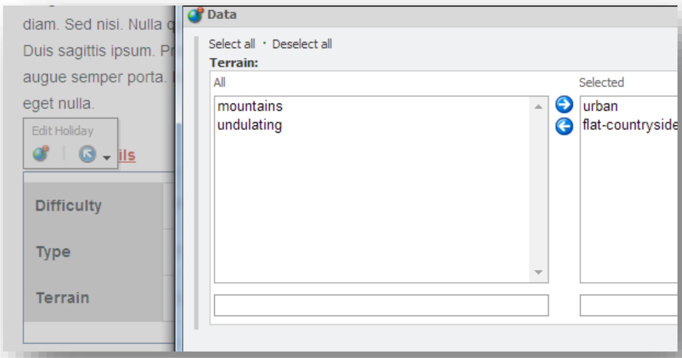
- The path to the **folder** of the custom experience buttons in the core database. Each edit frame must have a sub-folder under **Edit Frame Buttons** that contains one or more custom experience buttons. In this example, there is only one: **Edit Holiday Details**.


Custom Experience Buttons
Default Placeholder Buttons
Default Rendering Buttons
Edit Frame Buttons
BaseCore-HolidayEdit
Edit Holiday Details

- The data source item to target with the edit frame

```
        <div class="sectionItem searchResult">
             <sc:EditFrame Buttons="/sitecore/content/Applications/WebEdit/Edit Frame
Buttons/BaseCore-HolidayEdit" DataSource="<%# Item.ItemId.ToString() %>" runat="server">
                <!-- My search result -->
             </sc:EditFrame>
        </div>
```

Clicking the button opens a field editor dialog for the holiday item:



Note
If an editor does not have permission to modify the item that the edit frame is referencing, they will not see the button at all. Similarly, if an editor is limited to certain fields, the fields that they are not allowed to edit are greyed out.

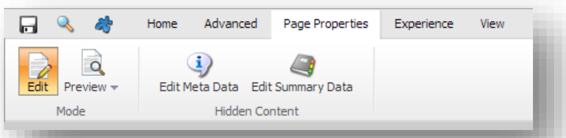
You can also use an edit frame to:

- Surround content that is brought in via AJAX or from an index rather than through the `FieldRenderer` pipeline on page load, and is therefore not editable by default (see Inline Editing Dynamic or Indexed Content for methods of editing content that has not come through the `FieldRenderer` pipeline).
- Surround entire groups of components. You may have created a custom webedit button that enables you to make bulk changes to the contained components.

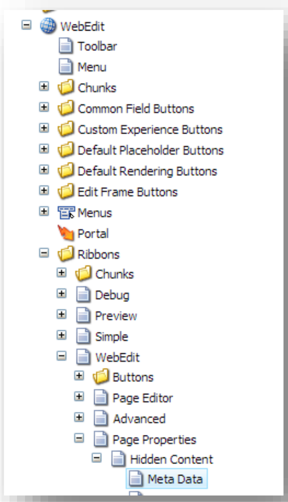
Important
Edit frames are not currently supported in Sitecore MVC.

5.3 Customizing the Ribbon

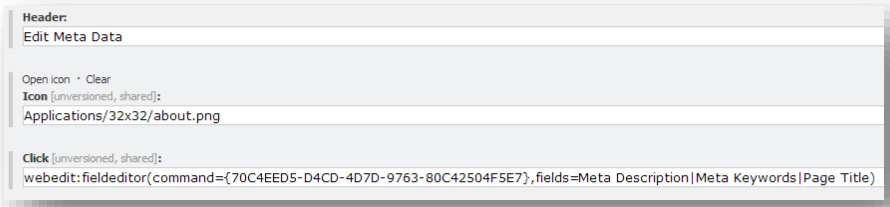
You can also add additional buttons to the ribbon. This is particularly useful if there are page fields that are never displayed, but do not belong to a particular component, either – such as the page title or meta description. The following is an example of a custom Page Editor tab:



Like custom experience buttons and edit frames, these additional buttons are defined in the core database in `/sitecore/content/Applications/WebEdit`:



The buttons (of type `/sitecore/templates/System/Ribbon/Large Button`) are very similar to custom experience buttons, and take a parameterized command in the **Click** field:

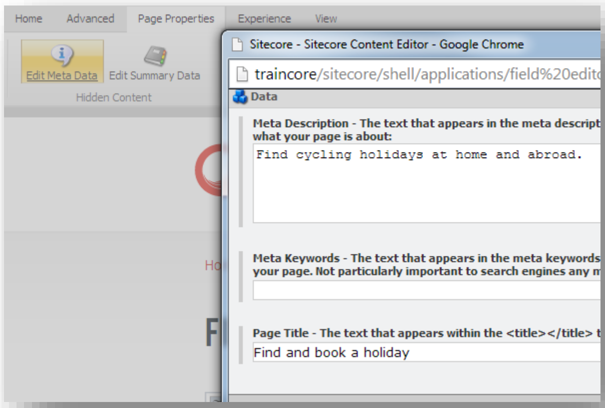


Header:
Edit Meta Data

Open icon · Clear
Icon [unversioned, shared]:
Applications/32x32/about.png

Click [unversioned, shared]:
webedit:fieldeditor(command={70C4EED5-D4CD-4D7D-9763-80C42504F5E7},fields=Meta Description|Meta Keywords|Page Title)

In the previous example, clicking the **Edit Meta Data** button opens a field editor with the fields



Sitecore - Sitecore Content Editor - Google Chrome

traincore/sitecore/shell/applications/field%20edit

Data

Meta Description - The text that appears in the meta descript
what your page is about:

Find cycling holidays at home and abroad.

Meta Keywords - The text that appears in the meta keywords
your page. Not particularly important to search engines any m

Page Title - The text that appears within the <title></title> t

Find and book a holiday

displayed:

With these additions to the ribbon, the editor no longer has to open the Content Editor to modify page-level fields that are not visible.

Chapter 6

Detecting the Page Editor

The following chapter covers how to detect Page Editor mode in C# and Javascript, and how detection can be used to improve the Page Editor user experience for editors.

The chapter contains the following sections:

- Detecting Page Mode
- Modifying Component Behavior Depending on Page Mode

6.1 Detecting Page Mode

The view presented to site visitors is not always appropriate for inline editing. This is especially true for animated elements. Developers can detect whether or not a site is being viewed in Page Editor mode and modify behavior accordingly.

Note
The same class that enables you to detect Page Editor mode can also be used to detect Debug, Preview, and Normal mode.

6.1.1 Detecting Page Mode Using C#

The `Sitecore.Context.PageMode` class enables developers to target the Page Editor in code. In addition to a generic `IsPageEditor` Boolean, you can also target edit and designing mode with `IsPageEditorEditingMode`.

```
        If (Sitecore.Context.PageMode.IsPageEditor)
        {
              // Page Editor-specific behavior
        }
```

The available options are:

- `IsDebugging`
- `IsNormal`
- `IsPageEditor`
- `IsPageEditorEditing` (encompasses Editing and Designing mode)
- `IsPreview`
- `IsProfiling`
- `IsSimulatedDevicePreviewing`

6.1.2 Detecting Page Mode in Sitecore MVC

Page Editor mode can be detected using exactly the same class in Sitecore MVC.

```
        @Sitecore.Context.PageMode.IsPageEditor
```

However, recommended practice within ASP.NET MVC is for all data required by the view to come from the view model. Therefore, you may want to incorporate an `IsPageEditor` property into your custom view model rather than calling `Sitecore.Context.PageMode.IsPageEditor` directly:

```
        public class CustomModel
              public bool IsPageEditor { get; set; }
        }
```

6.1.3 Detecting Page Mode Using Javascript

Sitecore makes a Javascript object available in Preview and Page Editor. In Preview mode, the following object is available:

```
Sitecore
```

In Page Editor mode, the following object is available:

```
Sitecore.PageModes.PageEditor
```

These objects can be used to change the behavior of your own Javascript:

```
if (Sitecore) {
    if (Sitecore.PageModes.PageEditor) {
        return 'pageeditor';
    }
    return 'preview';
}

return 'visitor';
};
```

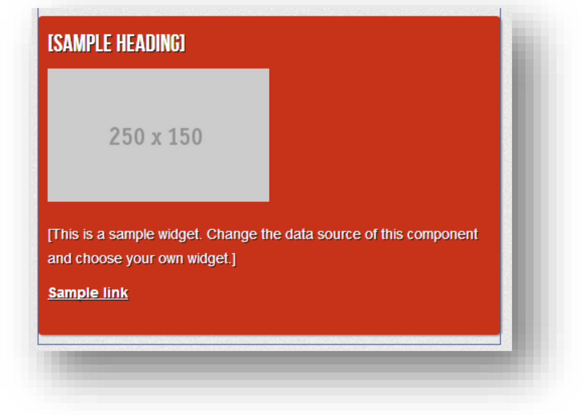
6.2 Modifying Component Behavior Depending on Page Mode

The following is a list of ways that you can utilize page mode detection to improve the Page Editor experience. This is by no means an exhaustive list.

Note
Debug and Preview mode should represent how a visitor will see your site. Therefore, it is recommend that you change site behavior specifically for the Page Editor (by checking for `Sitecore.Context.PageModes.IsPageEditor`). For example, if you hide or show components when `Sitecore.Context.PageModes.IsNormal` is false, this will affect Debug and Preview mode as well.

6.2.1 Sample Data

You can force an editor to choose a data source by setting a component's **Datasource location** and **Datasource template**. However, an editor might still accidentally clear the data source. In the following example, sample content is displayed in Page Editor mode if no data source has been specified. The content prompts editors to choose a data source:


[SAMPLE HEADING]
250 x 150
[This is a sample widget. Change the data source of this component and choose your own widget.]
Sample link

To do this, check if a data source has been specified. If a data source does not exist and you are in Page Editor mode, set the `Item` or `DataSource` property of each control (such as `<sc:Text />`) to a sample item somewhere else in the content tree. In Preview and Normal mode, hide the component from visitors.

```
        string datasource = (Sublayout)this.Parent.Datasource;

        if (datasource == null)
        {
             if (Sitecore.Context.PageMode.IsPageEditor)
             {
                Item sampleItem = MyReferences.SampleWidgetItem;

                HeadingTextControl.Item = sampleItem;
                HeadingTextControl.DisableWebEditing = true;
             }
             else
             {
               this.Visible = false;
             }
        }
```

There are a number of things to note about the above code:

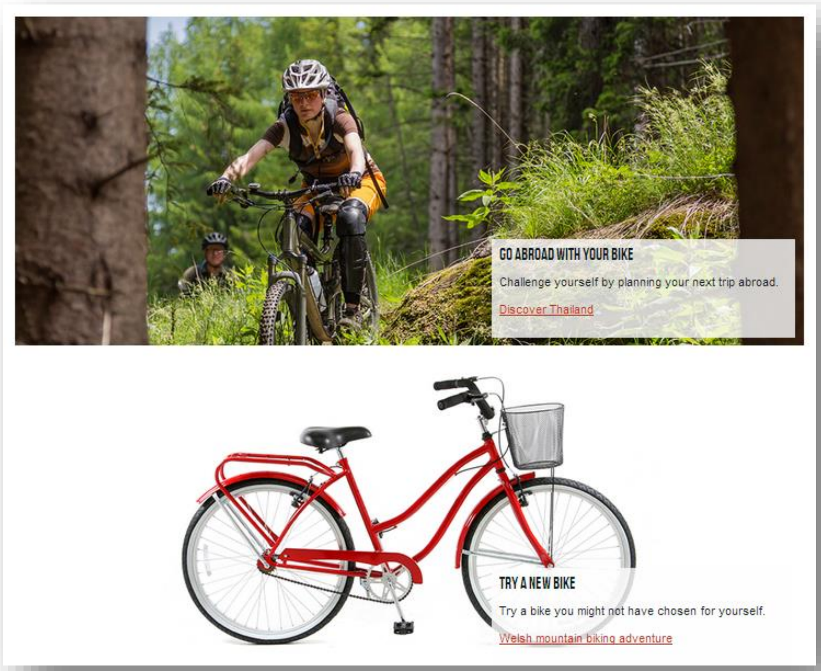
- `MyReferences.SampleWidget` is a reference to sample item somewhere else in the tree. The sample item is created from exactly the same data template as a real data source. The reason for using a sample item is that it can be translated, and any data template changes will be propagated. The content is also translatable.

- We are setting the `Item` property of each individual control rather than setting the component's `Datasource` property, which will not work.
- We are setting each Sitecore control's `DisableWebEditing` to true if the sample data source is being displayed. If we do not, editors may try to edit the sample text rather than choosing a new data source.

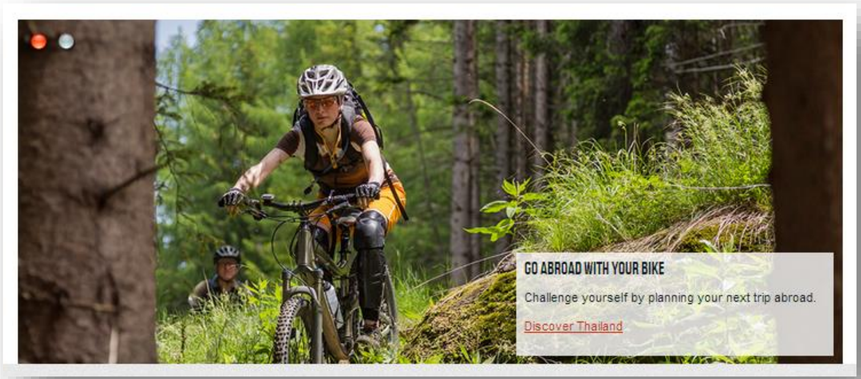
Alternatively, you can replace the entire component with a message prompting editors to choose a data source.

6.2.2 Dynamic Components

Animated components, such as scrolling galleries, can be difficult to edit. Front-end developers or back-end developers can suspend animation or display the content in a different way to improve the editing experience. In the following example, the gallery is not initialized in Page Editor mode:


GO ABROAD WITH YOUR BIKE
Challenge yourself by planning your next trip abroad.
Discover Thailand
TRY A NEW BIKE
Try a bike you might not have chosen for yourself.
Welsh mountain biking adventure

In Preview or Normal mode, the gallery is initialized as normal:

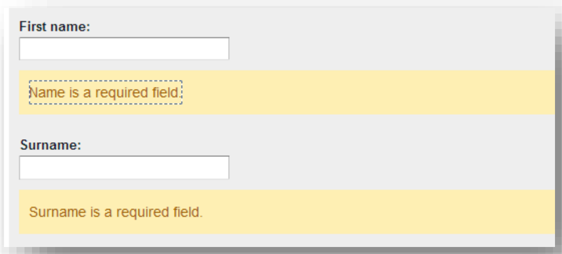


6.2.3 Hidden Content

Certain content may only be visible when the visitor interacts with the page. This might make it difficult – or impossible – for the editor to modify content inline, and may require opening the Content Editor. Examples of this type of content include:

- Form error and confirmation messages
- Modal windows
- Content that only appears on hover

Developers can help by displaying this content in context when the page is being viewed in Page Editor mode. In the following example, the form's error messages are displayed when the page loads.


First name:

Name is a required field.

Surname:

Surname is a required field.

Displaying hidden content reduces the risk of that content is missed, in particular, during translation.

6.2.4 External Content

External content, such as Twitter feeds or news streams, cannot be modified by the editor. External content can also increase page load time, and any included Javascript can conflict with the Page Editor.

Therefore, consider replacing external content with a static message that instructs the editor to switch to Preview mode if they want to see the fully rendered page.

Chapter 7

Tips and Tricks

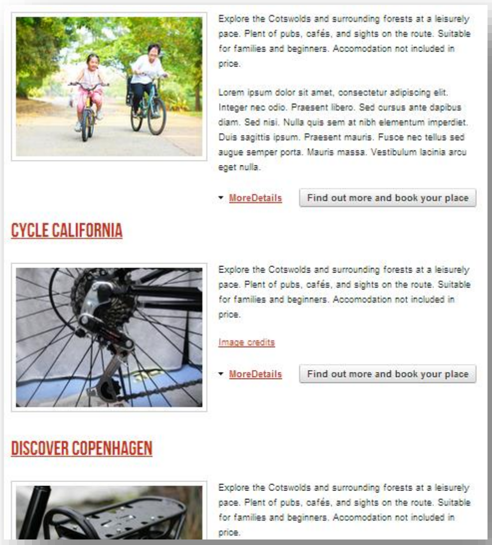
The following chapter lists miscellaneous tips and tricks for improving the Page Editor experience.

The chapter contains the following sections:

- Inline Editing Dynamic or Indexed Content
- The Page Editor and Sitecore Modules

7.1 Inline Editing Dynamic or Indexed Content

Content that is rendered directly from a search index (Sitecore supports Lucene and Solr out of the box) is not editable in the Page Editor. The following example shows three search results being output from an external index:



Explore the Cotswolds and surrounding forests at a leisurely pace. Plent of pubs, cafés, and sights on the route. Suitable for families and beginners. Accomodation not included in price.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

MoreDetails | Find out more and book your place

CYCLE CALIFORNIA

Explore the Cotswolds and surrounding forests at a leisurely pace. Plent of pubs, cafés, and sights on the route. Suitable for families and beginners. Accomodation not included in price.

Image credits

MoreDetails | Find out more and book your place

DISCOVER COPENHAGEN

Explore the Cotswolds and surrounding forests at a leisurely pace. Plent of pubs, cafés, and sights on the route. Suitable for families and beginners. Accomodation not included in price.

There are a number of ways to enable editors to manage this content:

- Surround each search result with an **Edit Frame**. This edit frame should point to a folder of field editor buttons that expose the fields you want to make editable (such as the search result title or description). Set the edit frame data source to the item ID represented by the search result (Sitecore's `SearchResultItem` has an `ID` property). If the editor has access to the item, they will be able to click the field editor button and update the content.
- Render search result content from the database rather than from the index when in Page Editor mode. This requires using the `ID` property from the `SearchResultItem` (or equivalent, if using a custom class) to retrieve the item, and outputting its field values using the `FieldRenderer` instead of the indexed values.

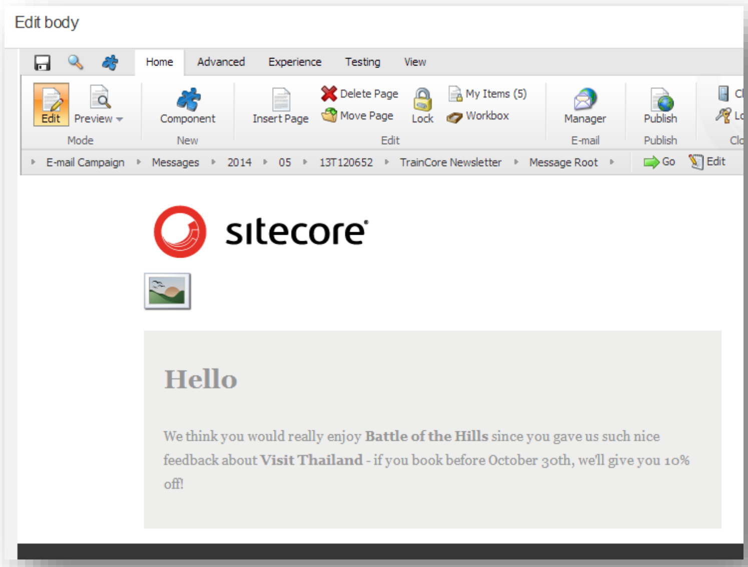
The same principle applies to any content that has been brought onto the page via **AJAX**. Even if this content comes from Sitecore originally (via a custom web API, for example), it is not rendered by the `FieldRenderer` pipeline and will therefore not be editable in the Page Editor. Consider displaying the data on page load in Page Editor mode only, to allow for editing.

7.2 The Page Editor and Sitecore Modules

7.2.1 Email Campaign Manager (ECM)

Email Campaign Manager emails are built from components using the Page Editor interface. If you want to build your own ECM email, emails should be built with exactly the same principles as a standard Sitecore page:

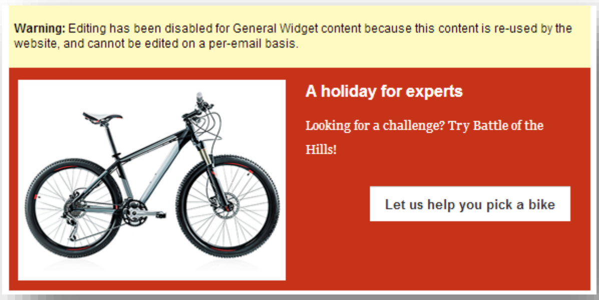
- Where appropriate, ensure that your HTML email design supports componentization
- Ensure that you are supporting both designing and editing
- Configure your component definition items and placeholder settings items as required



Hello

We think you would really enjoy **Battle of the Hills** since you gave us such nice feedback about **Visit Thailand** - if you book before October 30th, we'll give you 10% off!

Email components can use the same data sources as the main website. However, editors might not understand that this content is shared. If they change content in the context of an email, these changes will appear on the main site as well. This could present a problem if the changes are specific to a particular email. To avoid this problem, you can disable editing for a component that uses shared content or display a warning when the editor is in Page Editor mode:



Warning: Editing has been disabled for General Widget content because this content is re-used by the website, and cannot be edited on a per-email basis.

A holiday for experts

Looking for a challenge? Try Battle of the Hills!

7.2.2 Web Forms for Marketers (WFFM)

Web Forms for Marketers prompt you to add the **Form** component to a particular placeholder when you install the module. If you want to make the component available in other placeholders, add it to the **Allowed controls** list of the placeholder settings item.

7.2.3 JQuery Mobile Component Library for Sitecore

There is a Page Editor-compatible JQuery Mobile Component Library available on the Sitecore Marketplace:
https://marketplace.sitecore.net/en/Modules/Sitecore_jQuery_Mobile_Component_Library.aspx