



SITECORE BASICS



Sitecore Is an Experience Platform

Sitecore combines content management with a marketing platform

- *Design* and *populate* a website
- *Tailor* the *experience* to individual visitors

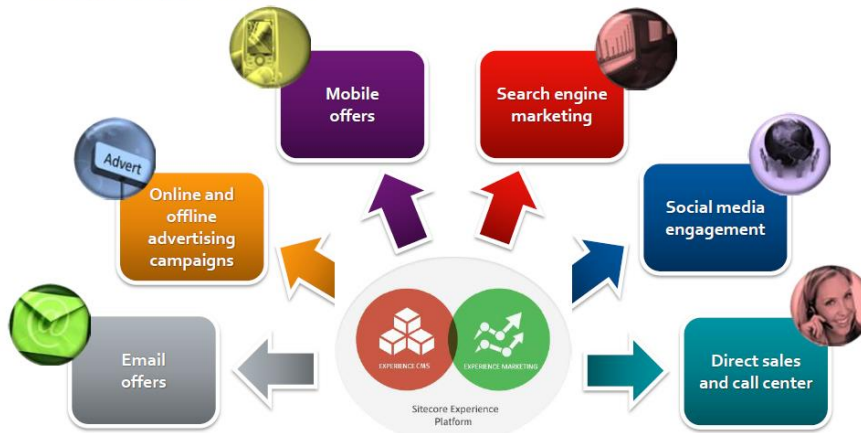


What does Sitecore offer marketers?

- Inline page *design* and *population*
- *Personalization* and *multivariate testing*
- *Cross-channel* communication
- Integration with *external tools* and *databases*
- Rich reporting on *individual visitors*

Cross-Channel Communication

Offering a consistent, individual experience regardless of how visitors interact with your content



The Developer's Role

Who are you?

- *Professional* ASP.NET developers
- Passionate about *supporting business goals*

What are your responsibilities?

- Build and scale according to *recommended practices*
- Support Sitecore's *marketing* and *analytics* features
- Build for the *Experience Editor*
- "Don't fight the framework!"

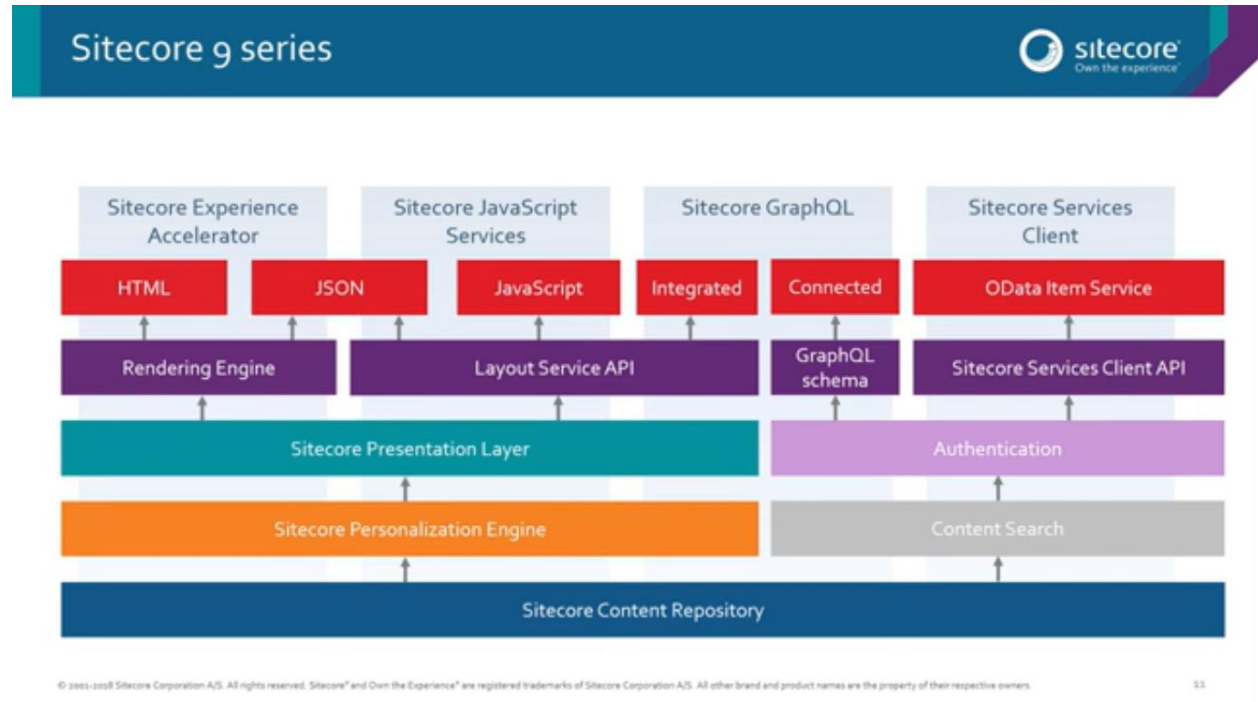
This course will get you there!



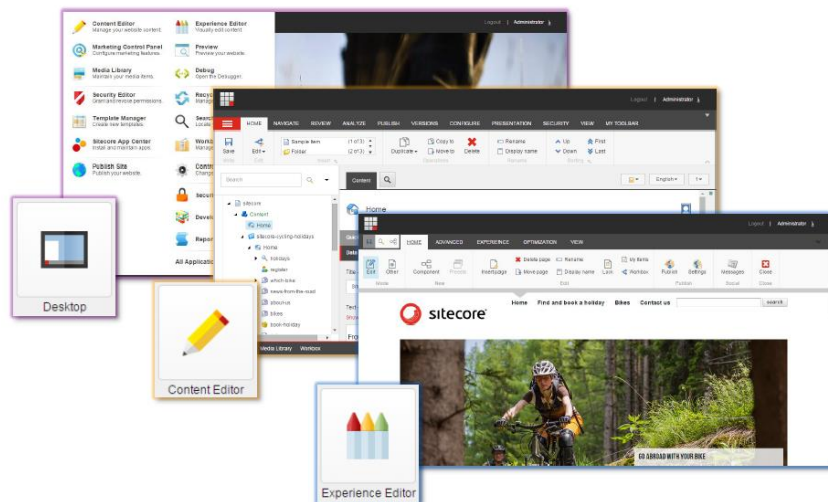
Where Can I Find More Information?

- SDN: <http://sdn.Sitecore.net>
- Knowledge Base: <https://kb.Sitecore.net>
- Marketplace: <https://marketplace.Sitecore.net>
- SitecoreXP: <https://dev.Sitecore.net>
- SitecoreXP Documents: <https://doc.Sitecore.net>

Sitecore Architecture



Sitecore as a Web Experience Manager (WXM)



Demo 1.3 – Sitecore Interfaces

In this demo:

- Use the *Experience Editor* – navigating, editing and designing
- Use the *Desktop*
- Use the *Content Editor*
- *Publish* content

<http://basicsitecore/>

To log in, append **/sitecore** to your URL:



Click Icon to select an editing interface after successful login



Items and the Content Tree

Items

- Everything in Sitecore is an *item*
- An item is a *unit of content*, not a page
- Items are *not* files
- Each item has a *GUID* (Global Unique Identifier)

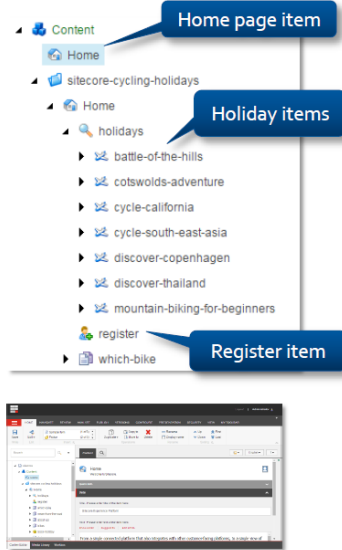
Item Name: Home
Item ID: {110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}

Content Tree

- Items are represented in a hierarchical structure called the *Content tree*
- Sitecore *interfaces* display all or part of the Content tree

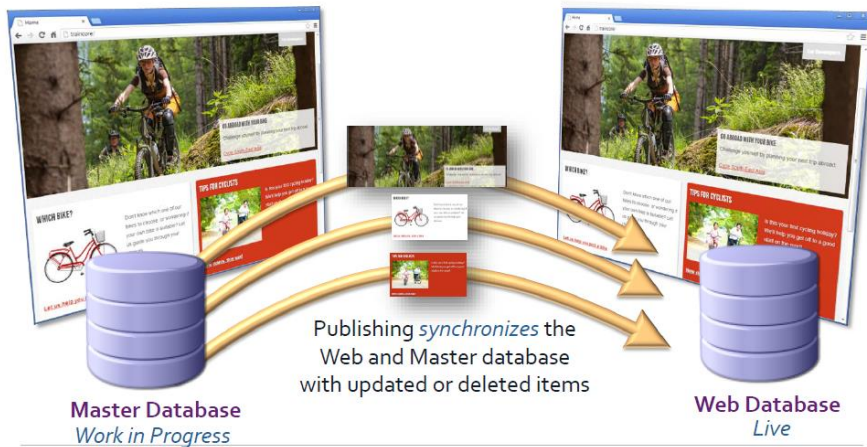
URLs

- Some items are addressable via a *URL*
- URL maps to *position in tree*



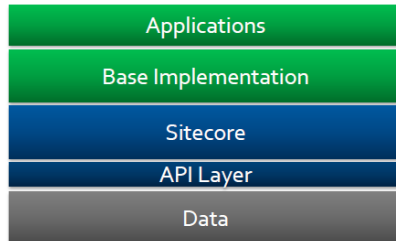
Publishing and the Sitecore Databases

- Sitecore *Publishes* items from the *Master* to the *Web* database



Sitecore Architecture

Sitecore is an ASP.NET application



Sitecore is a framework for:

Content versioning, media library, design/editing interfaces, language versioning, presentation versioning, search API, adaptive design via devices, rich API, hierarchical data storage and much more!

Separation of Concerns

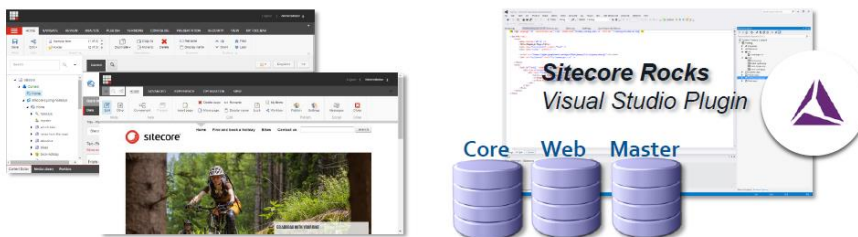
Authors and developers have separate responsibilities

Authors...

- Manage *content* and *presentation* using your *custom* implementation of Sitecore

Developers...

- Define *data structure* and *presentation components* for authors



Pages are *dynamically* assembled *on demand*

What Gets Installed?

Databases

- *Master* – authoring database, *work in progress*
- *Web* – published, *live content*
- *Core* – settings and membership
- *xDB* – MongoDB NoSQL collection. New starting with 7.5!
- *Reporting* – replaces Analytics DB previous to version 7.5



Installation Scenarios – Recommended Minimum



Content
Management
server



Database
server



Content
Delivery
server

Authoring environment

Previewing capabilities

- Access to master
- Access to web
- Access to core



Live environment

No access to Sitecore client (/sitecore)*

- Access to web
- Access to core



Sitecore 8 – Experience Database (xDB) and Experience Marketing

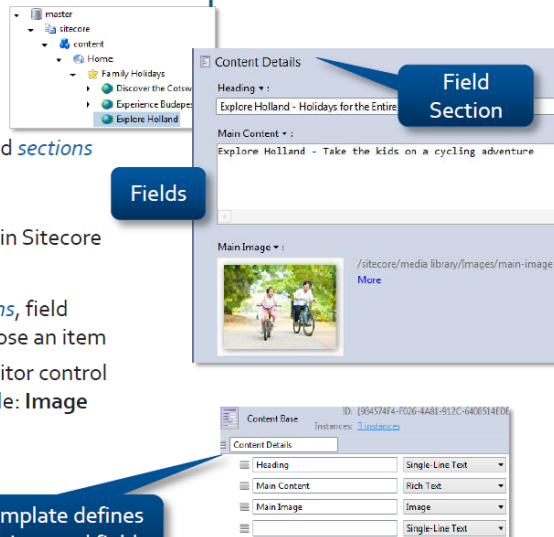
Items, Fields and Data Templates

An item

- Is a unit of *data* in Sitecore
- Has *fields* organized into field *sections*

Data templates

- *Fundamental* building block in Sitecore
- Define a *type* of item
- Determine what field *sections*, field *types* and field *names* compose an item
- Field types *determine* the editor control used in the tools, for example: Image fields, Rich Text fields

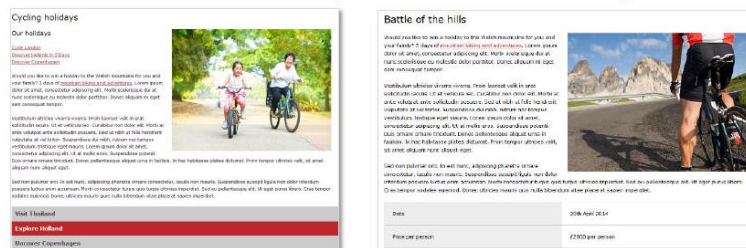


Deciding the Data Structure Architecture

Architecture inferred from designs, wireframes and static HTML

- Project should start with focus on *designing your data structure* (like SQL schema)
- Examine the content on the page and across all pages
- Is the structure of content *repeated* on multiple pages?
- *Recommended Practice*: Use a Sitecore Partner for your first project

What data architecture can you infer from the designs below?



Building Data Templates

Start by building data templates

- After your thorough data analysis, begin by building base templates
- Include Fields and Field Sections that can be reused
- In our project, Content Base will become the base template for our future templates
- The easiest way to create data templates is using Sitecore Rocks

Content Base

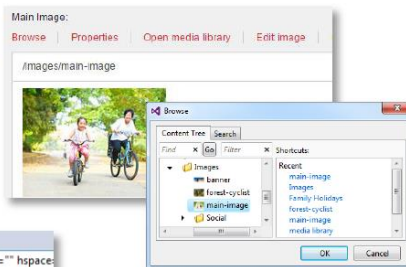
Heading
Main Content
Main Image

Field Type Determines...

- *Editing controls* used in the Sitecore Tools

- *Raw value stored* in the database

Main Image ▾ :
<image mediapath="/Images/main-image" alt="main-image" width="" height="" hspace="" image" mediaid="2CED451A-1C5B-4C76-97A3-6D4EE30A6851" />



Content Base	ID: 0B8374F4-F02B-47E1-912C-6408314ED6
Instances:	2 instances
Content Details	
Heading	Single-Line Text
Main Content	Rich Text
Main Image	Image
	Single-Line Text

Standard Values Allow You to Define Defaults



Standard Values item

- *Manually* create
- *Child* item of the data template
- Includes the data template fields and all *inherited* fields
- Always named `__Standard Values`

Standard Values allow you to...

- Specify default field values
- Set dynamic field values using tokens

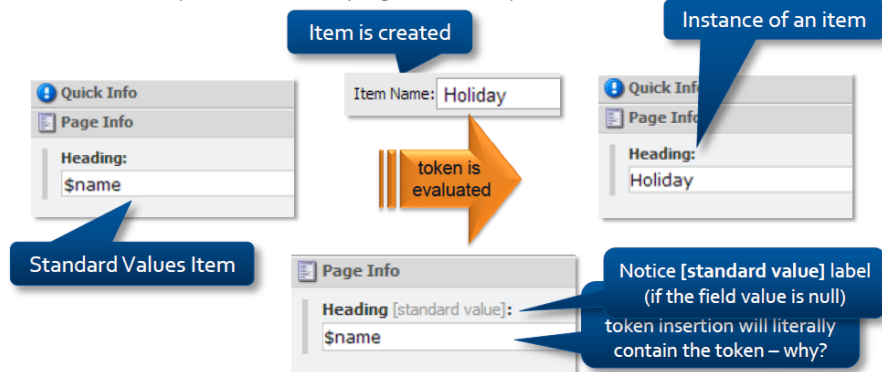
We will also cover assigning...

- Insert Options
- Default look and feel
- Default workflow state

Tokens

Tokens are replaced the moment an item is created

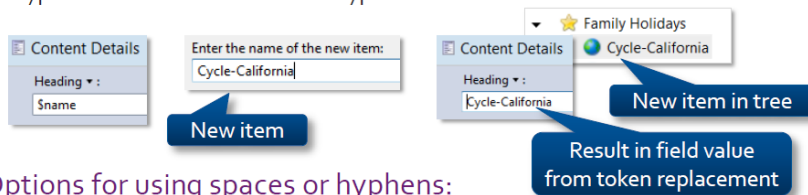
- `$name`, `$id`, `$time`, `$parentid` (further examples in Student Manual)
- You can add your own tokens programmatically



Item Naming Conventions

Business names preferred approach:

- Name items using spaces so tokens expand in fields correctly
- Hyphenated item names result in hyphenated field values where tokens were used



Options for using spaces or hyphens:

- *Create* item name *using spaces*, then *rename* item replacing spaces with hyphens
- This avoids item URL's being encoded, %20 will replace any spaces
- Add entry to <encodeNameReplacements> in web.config
- Character replacement is fragile and risky
- Can write custom solution (*pipeline*, *event*, *rules*)

Requires hyphen to be illegal character in item names

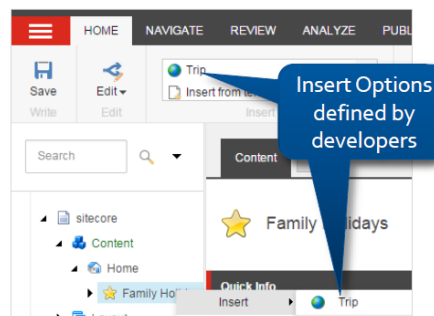
What Are Insert Options?

Why Insert Options?

- Authors *require* a list of *allowed* item types (data templates)
- Developers and administrators are less *restricted*

Insert Options...

- Define *allowed* child items
- Allow authors to *build* multiple levels of content in the tree
- Reduce *risk* of human error
- Usually *defined* on Standard Values
- Can be *overridden* on an item



Key Vocabulary

Data Template

Defines an item's type. Contains field sections and fields. Should always have a unique icon. Used to create items.

Field

Holds content. Can be different types: Single-Line Text, Rich Text, Image, General Link and more. Type determines the interface that the author sees for editing that field.

Data Template Inheritance

Data templates can inherit fields and field sections from other data templates. Reduces duplication.

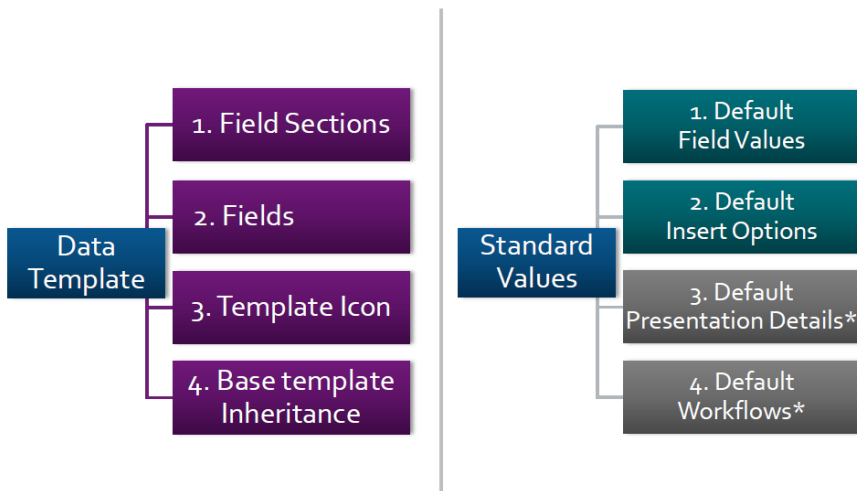
Standard Values

Standard Values is a child item of a data template and is the mechanism for defaults when items are created.

Insert Options

List of item types that can be inserted by authors as a child of a particular item.

Data Template vs. Standard Values



How Sitecore Resolves a Page

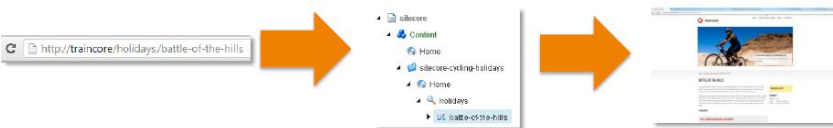
Static HTML

- URL points to an *HTML file* with the same name



Sitecore

- URL points to an *item* in Sitecore, which has a set of *rendering instructions*

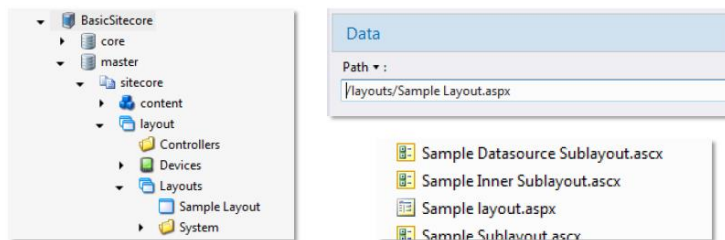


What Is a Layout?

A Layout is...

- An *.aspx* file on the file system
- A corresponding *Layout definition item* in the tree
- Linked by the *Path* field

See lab instructions for details on how to create a Layout in Sitecore Rocks!

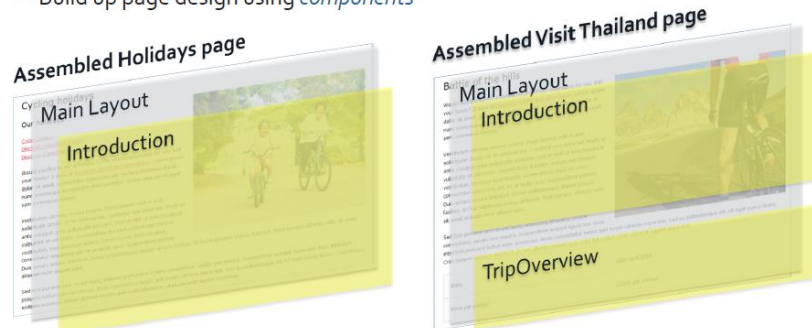


Note: The item *and* the file can be referred to as a "*layout*"

One Layout for All Items in Our Site

A Layout is a canvas or the 'scaffolding' for a site

- The training site **BasicSitecore** defines a *single* shared Layout for all items within the site (*per device*)
- Assign Layout on standard values (consider setting Layouts on *base* templates)
- Build up page design using *components*

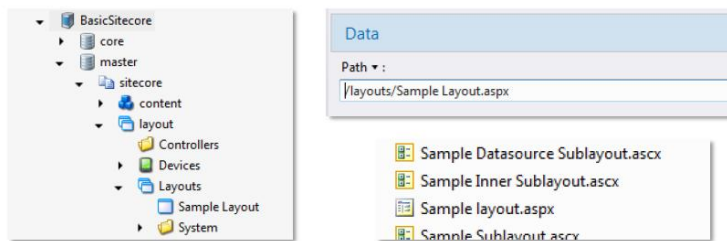


What Is a Layout?

A Layout is...

- An *.aspx* file on the file system
- A corresponding *Layout definition item* in the tree
- Linked by the *Path* field

See lab instructions for details on how to create a Layout in Sitecore Rocks!

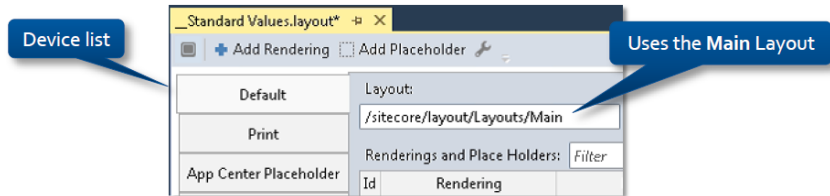


Note: The item *and* the file can be referred to as a "*layout*"

Assigning a Layout

Items must have a Layout to be viewed as web pages

- *Layouts* are part of an item's *presentation details*
- Recommended practice – *assign* Layouts to the data template's *standard values*
- An item can have *one* Layout *per device** (mobile, print, RSS)



Movable, Reusable Components

Pages are assembled from smaller units of functionality

- Various types – collectively known as *components*
- Components can be *re-used* and *moved* and they can be *nested*
- Arranged to create a unique *page design*



What Is a Component?

A component can be a...

- Sublayout (Sitecore's wrapper for a *Web User Control*)
- XSLT Rendering
- Sitecore Web Control



We will use the term *component*

There are *Sitecore MVC* alternatives – see videos!

All components consist of...

Definition item in Sitecore:

/sitecore/Layout/Sublayouts
/sitecore/Layout/Renderings

File on the file system:

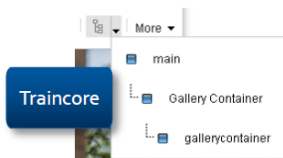
.ascx (Sublayout)
.xslt (XSLT rendering)

Linked by a Path field on definition item

What Are Placeholders?

- Placeholders are *Sitecore controls* used in Layouts and Sublayouts
 - Define *where* components can *dynamically* be bound to an area of a page
 - Can be added anywhere in the markup of *Layouts* or *Sublayouts*
- *Identified* by an attribute called *Key* (the ID attribute is not required)
- The Placeholder is *added directly* in the markup:

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



Lowercase Keys are recommended, easier to distinguish vs component names in Experience Editor

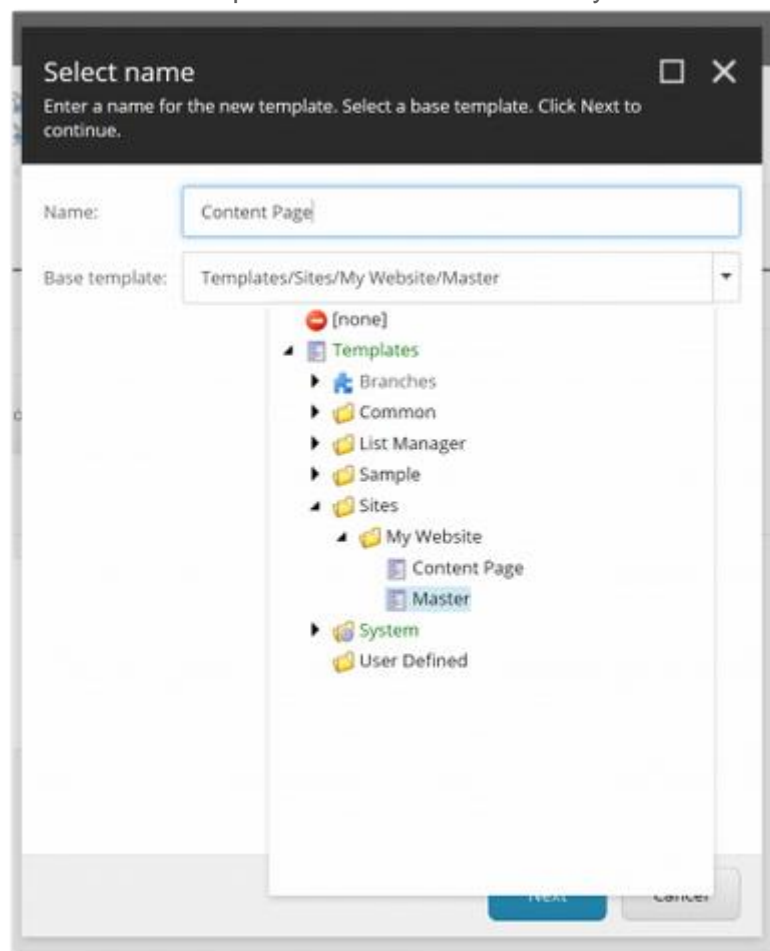
(Module 7 covers Placeholder Settings items, which are required by Experience Editor)

How To Create A Template For A Content Page In Sitecore

This is the third post in the 'Creating Your First Sitecore Page' series. In the last article, [How To Create A Master Data Template In Sitecore](#), we created a master template. The purpose of the master template is to define a schema for all the common properties that exists on all of our web pages. The page title is one such example. With our 'master' template created, we will now create a template for a standard content page.

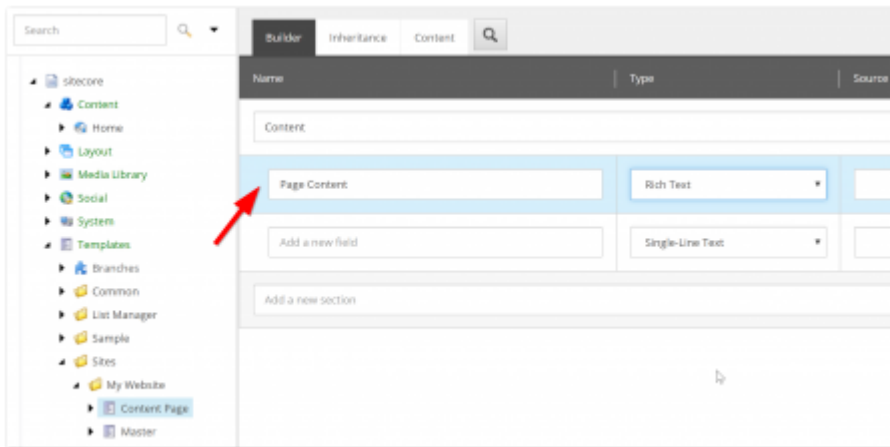
Creating Your First Sitecore Page Template

To create our content page, we need to create a new data template. Content page templates are pretty basic, a usual content page obviously needs all the standard page template settings, so it will need to inherit from the master template. It will also need a way for content editors to



add rich-text onto a page.

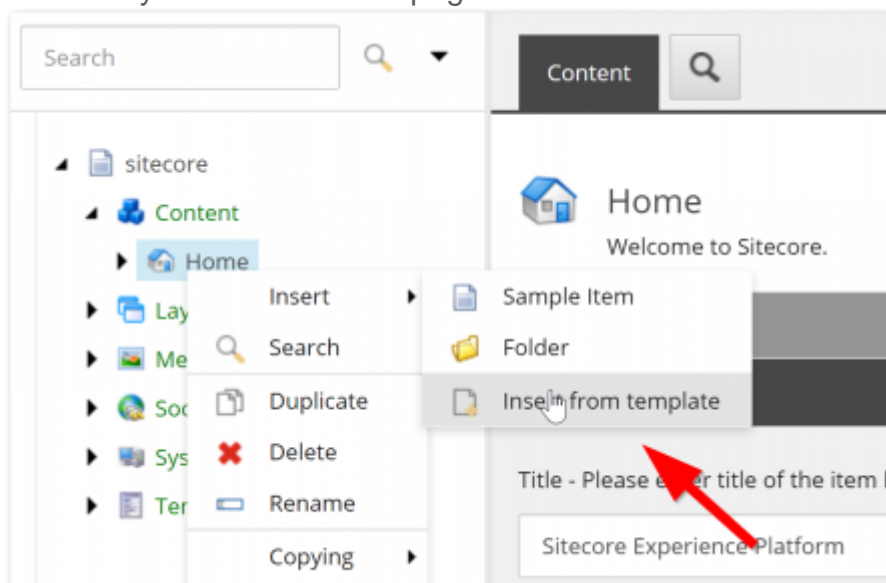
In the 'templates' area within the Sitecore content tree, create a new template within a folder (I recommend creating a folder under templates, called your host). When you create the new template, you will obvious need to give it a name. You will also need to set the 'base' template to the master template we created in the last



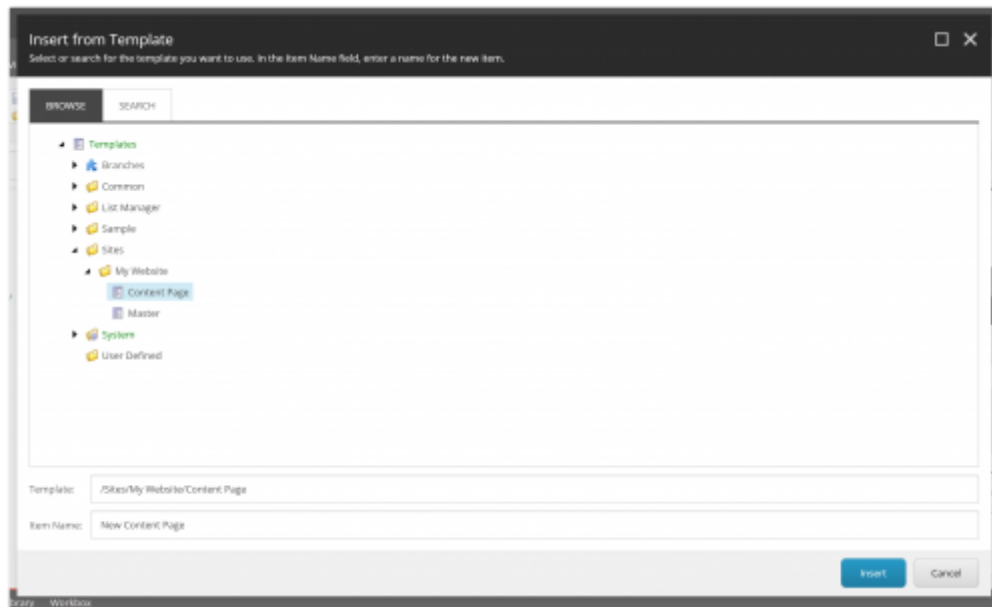
article. The only real thing you'll need in your content page template is a 'Content' section and a 'page content' property that allows 'rich-text' to be entered. Now we have a very basic master and page template set-up, we can create a content page.

How To Create A Page With A Data Template

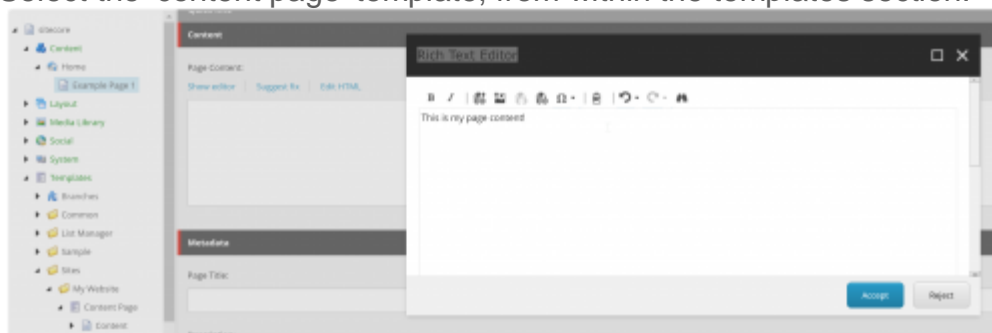
To finish off this article, I'll quickly run through how to create a page using our data template. In the Sitecore content tree, you should see a 'Content' item. This is the area where all your websites web pages will live.



If you right click on the 'home' item and select 'Insert' -> 'Insert from template'. You'll be presented with the template picker:



Select the 'content page' template, from within the templates section.



Now you can create your content page. As you can see above, the page content property loads the rich-text editor. In here, content editors can add and format text that we can later render out in a layout.

Metadata

Page Title:

Description:

Settings

☐ Display On Sitemap

In the content page, we also have our common global properties that are inherited from the master template. As I'm hoping you can see, this template inheritance is pretty simple to implement but gives you a lot of power. You might not split your templates up as efficiently as possible on your first project, but, after a few sites, you'll be a pro. We still have quite a lot to do before we can display our page on our website. In the next lesson, I'll cover layouts. The layout file is where we talk about HTML. The layout file is where we define how our site looks and what data should be displayed to the end user.

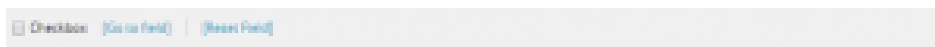
Mostly used fields Types in Sitecore

The Simple field types represent an individual value. An individual value may have multiple properties, such as the attributes in an Image field or the images and links in a Rich Text field. Sitecore provides the Simple field types described in the following sections.

Checkbox

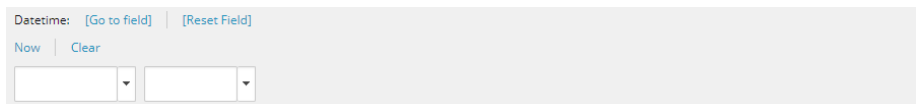
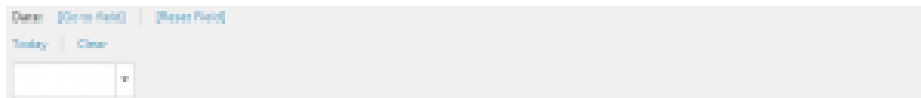
This field type displays a toggle button. If the user selects the checkbox, Sitecore stores the value 1 (the number one). If the user does not select the checkbox, Sitecore stores a blank value.

Tip Check for the literal value 1 in a checkbox field, or use a property such as `Sitecore.Data.Fields.CheckboxField.Checked`. Do not compare the value of a Checkbox field against `NULL`, `String.Empty`, 0 (the number zero), or any value other than 1.



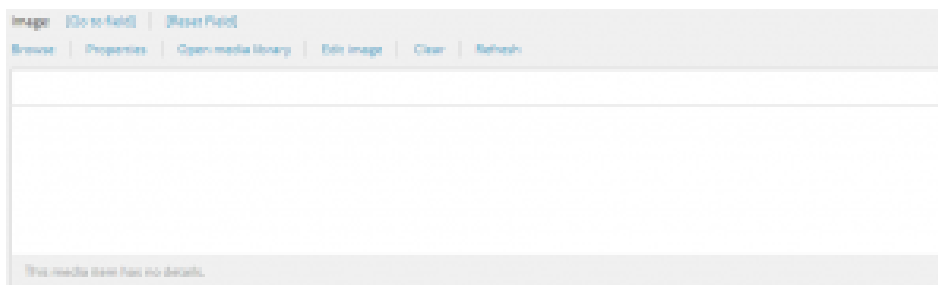
Date and Datetime

The Date field type provides a user interface allowing a user interface to select or enter a date using manually. The Datetime field type adds a user interface to select or enter a time manually. Sitecore stores the content as a text string with the format `yyyymmddThhmmss`. If the user does not enter a time, Sitecore stores the value `000000` (midnight). The value stored represents local time for the Web server.



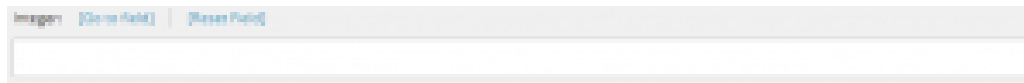
Image

The Image field type allows the user to select an image from the media library and specify image properties. Users can click Open file to open the Media Browser and select the desired file from the media library. The Source property of an Image field controls the selected item in the Media Browser. For more information about the Source property, see the section File Field Type.



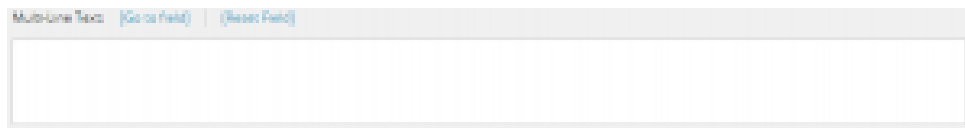
Integer

This field type store a single line of text. With a validation for integer values.

The image shows the user interface for an Integer field. At the top, there is a header bar with the label "Integer" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header is a single-line text input field.

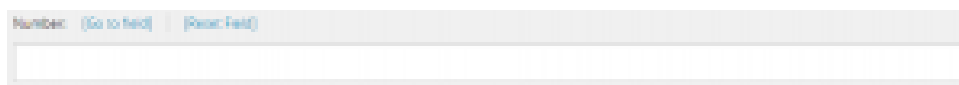
Multi-Line Text

Use the Multi-line field type for simple text values spanning multiple lines. This field has no validation and there is no Rich Text support for fields of this type.

The image shows the user interface for a Multi-Line Text field. At the top, there is a header bar with the label "Multi-Line Text" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header is a multi-line text input area.

Number

This field type store a single line of text. With a validation for number values.

The image shows the user interface for a Number field. At the top, there is a header bar with the label "Number" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header is a single-line text input field.

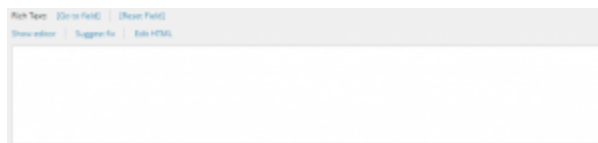
Password

This field type store a single line of text. With a mask for password values.

The image shows the user interface for a Password field. At the top, there is a header bar with the label "Password" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header is a single-line text input field with a password mask.


Rich Text

This field type stores HTML text. The field displays the contents as a browser would display the source HTML, although the field actually stores character encoded HTML. The Show Editor button provides access to a Rich Text Editor, while the Edit Html button provides access to the stored HTML. For further information about Rich Text fields, see the Content Reference manual.

The image shows the user interface for a Rich Text field. At the top, there is a header bar with the label "Rich Text" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header, there are three buttons: "Show editor", "Toggle fix", and "Edit HTML". Below these buttons is a multi-line text input area.

Single-Line Text

This field type store a single line of text.

The image shows the user interface for a Single-Line Text field. At the top, there is a header bar with the label "Single-Line Text" and two buttons: "(Go to Field)" and "(Reset Field)". Below this header is a single-line text input field.

Youtube videos links for Sitecore basics

- Raw Values in Sitecore - <https://www.youtube.com/watch?v=UE1LTrHrWOE>
- Insert Options in Sitecore - https://www.youtube.com/watch?v=358myfE4o_E
- Standard Values in Sitecore - https://www.youtube.com/watch?v=y_w34Pfkbsb4
- Standard Values Tokens in Sitecore - <https://www.youtube.com/watch?v=GR7MiEaozI0>
- Help Texts in Sitecore - <https://www.youtube.com/watch?v=dbow4buJa40>
- Layout in Sitecore - <https://www.youtube.com/watch?v=aCE9LeKGtwo>