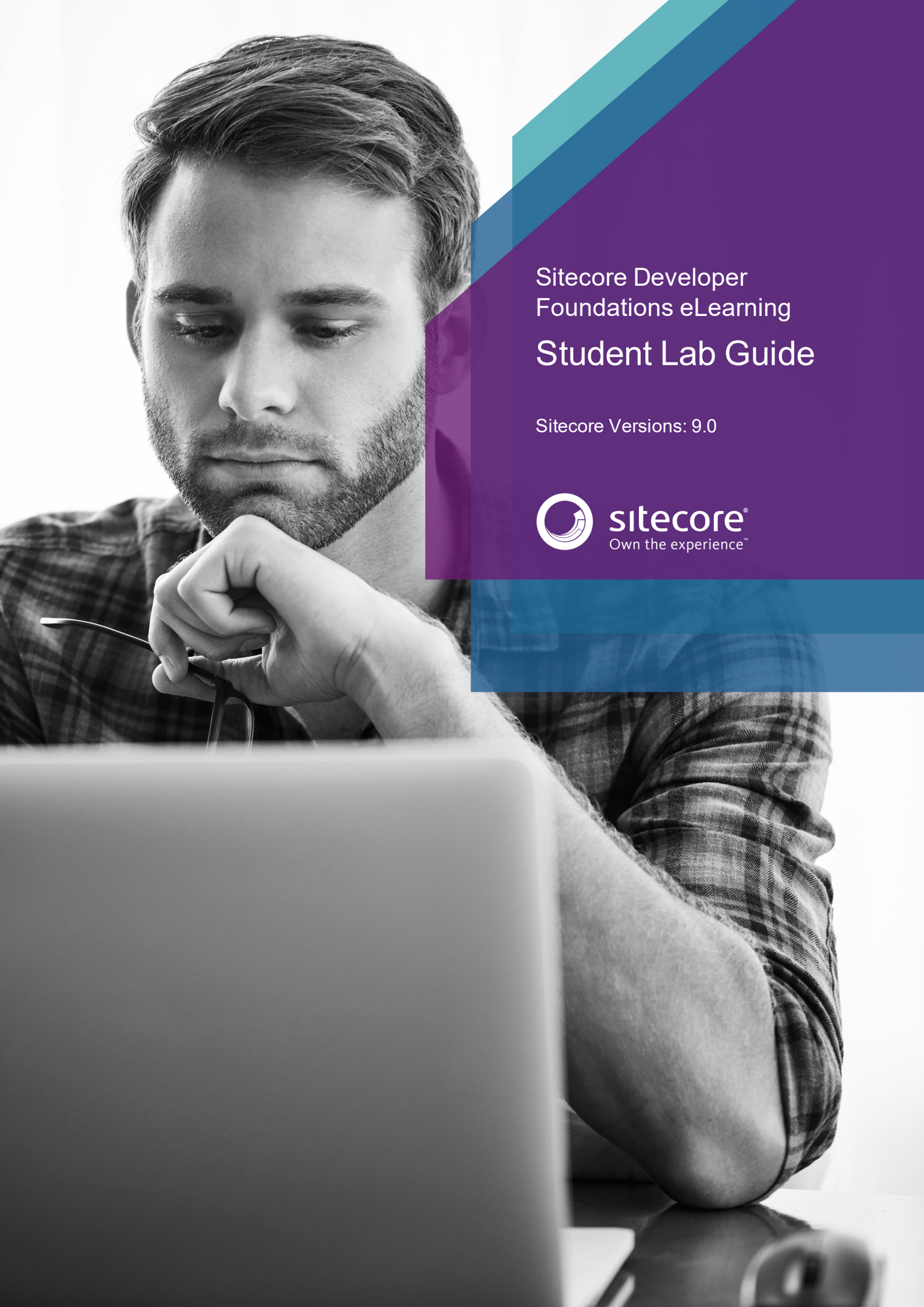


Sitecore Developer Foundations eLearning

Student Lab Guide

Sitecore Versions: 9.0

sitecore®
Own the experience™

CONTENTS

MODULE 1: Defining Sitecore

In this module, you will learn about the basic concepts of the Sitecore Experience Manager, which is Sitecore's Web Content Management System. This includes the foundation pillars and features of a Web Content Management System (WCMS). You will explore Sitecore users and applications, and learn about Sitecore's basic architecture, and key terminology.

You will also get introduced to The Adventure Company, a fictitious company created for Sitecore training purposes.

By the end of this module, you will be able to:

- Define a Experience Content Management System (xManagement).
- Describe the features of a WCMS.
- State the three foundation pillars of a WCMS.
- Describe Sitecore's architecture.
- Use basic Sitecore terminology.
- Identify the Sitecore applications and their users.

Lab 1.1: Exploring the Bootstrap-based HTML Templates

In this lab, you will learn how to create the templates by using the Twitter bootstrap templates. To get familiar with the HTML structure used, open the HTML-templates in a text editor. You will need a text editor for the later labs too.

While it is possible to use something as basic as Notepad, you will find it easier to use a more feature-rich tool such as Visual Studio Code, Notepad ++ or Sublime. Visual Studio Code is free to download and open source.

Detailed Steps

1. Open the **Student Resources** folder that you downloaded, that contains the HTML files.
2. Open **"Careers.html"** in a browser.
3. Make a note of the various page elements. Next, you will locate the code behind them.
    a. The header containing the site logo and navigation bar
    b. The banner and banner text
    c. The job and qualifications lists
    d. The spotlight job
    e. The footer
4. Open the **Careers.html** file in a text editor and find the markup that:
    a. Displays the Header (logo and navigation).
    b. Displays the banner.
    c. Shows the job and qualification lists.
    d. Shows the spotlight job.
    e. Belongs in the footer area
5. Open also the **Content-Page.html** file and notice that a lot of the page has the same markup as Careers.
6. Open the **WebDeveloper.html** file and find the part of the markup that differs from Content-Page.
7. Make a **wireframe diagram** of each of the three pages and mark basic elements in each one. Identify the common parts.

MODULE 2: Building the Site's Data Infrastructure

By the end of this module, you will be able to:

- Describe the templates' building blocks.
- Name the field types and their purpose.
- Give examples of how you use source fields.
- Identify when you need to use template inheritance.
- Apply template inheritance to new and existing templates.
- List what you should consider when you apply inheritance to templates.
- Describe the purpose of the Standard template.
- Use the Template Manager to build and configure a site infrastructure.
- State the importance of setting up icons on templates.

Lab 2.1: Creating Data Templates

In this lab, you will create some of the necessary templates to store the content for the site. You will start by creating a new folder where you store all your templates. You will then create a few templates.

2.1.1: Create a Template Folder

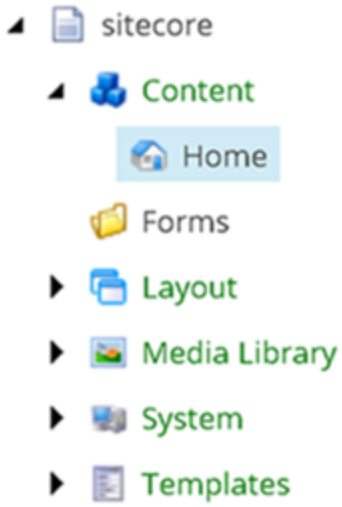
Detailed Steps

1. In a browser navigate to http://jobs.tac.local/sitecore.
2. Log in to Sitecore as admin using the default credentials:

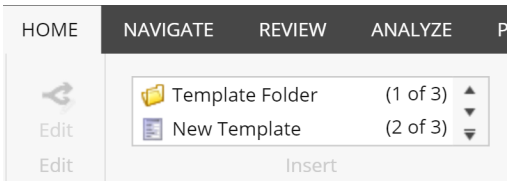
   User name: admin

   Password: b
3. On the Launchpad, select **Content Editor** .

   .
4. Select **/sitecore/Templates** in the content tree.



5. On the **Home** tab in the **Insert** group, click the **Template Folder.**



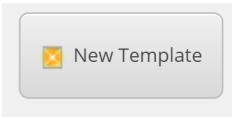
6. In the dialog box, enter the text *Jobs* and click **OK.**

> **NOTE:** It is best practice to put all your templates in a custom folder because this allows you to separate them by project or function.

2.1.2: Creating templates

Detailed Steps

1. Click the **New Template** button.

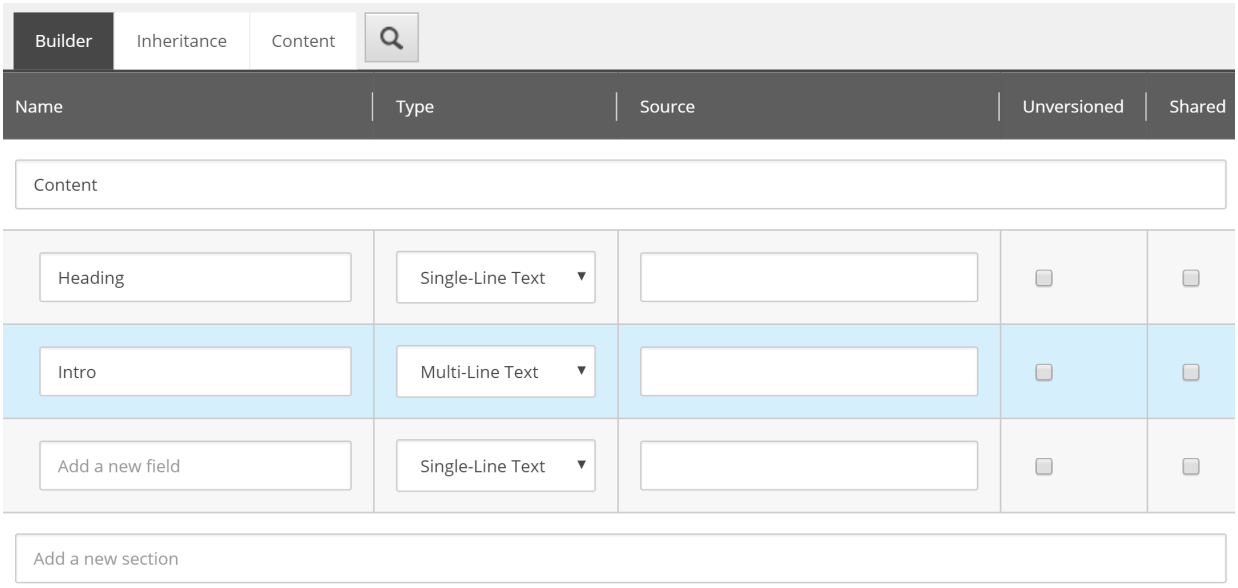


2. Name the template *Site* and click **Next.**
3. Confirm the location of the template inside the **Jobs** folder. Click **Next** and then **Close**.
4. Click the *Add new section* textbox and type *Content.*
5. Click the *Add new field* textbox and type *Heading.*
6. Click the textbox *Add new field* and then type *Intro.*
7. On the **Type** drop-down menu, select **Multi-Line Text.**

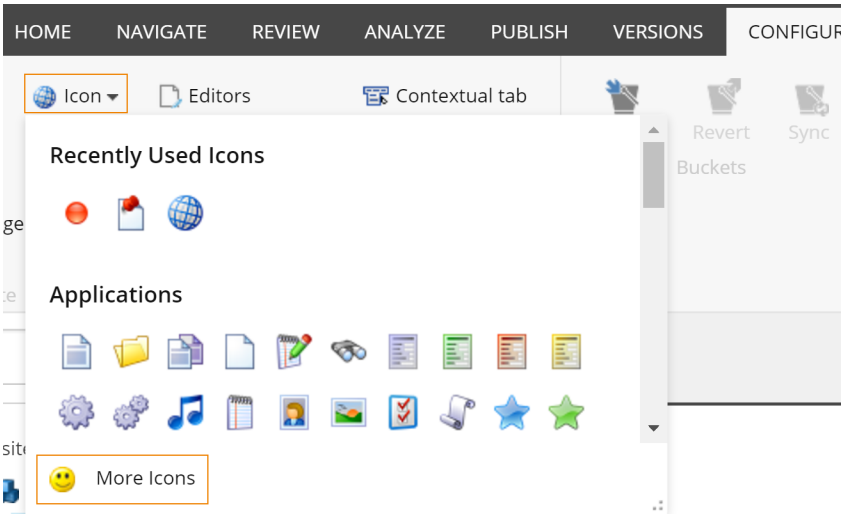
8. Click **Save** on the **Home** tab of the toolbar.

> You can also press Ctrl+S to save your changes in the Content Editor

You have now created a Sitecore template with two fields. Your screen should look like this:



9. You now need to set an icon for the *Site* template. On the **Configure** tab, **Appearance** group, use the Icon drop-down menu and select **More Icons.**



10. In the **Icon** textbox at the bottom of the dialog, type: *Network/32x32/environment.png.* Click **OK** to close the dialog.

2.1.3: Create the Other templates

Now you will repeat the steps you just performed to create the following templates:

Template Name	Field Section and Fields	Icon
GenericPage	Content • Heading - Single-Line Text • Intro - Multi-Line Text • Description - Rich Text	Applications/16x16/document_plain.png
JobDetails	Content • Heading - Single-Line Text • Intro - Multi-Line Text • Description - Rich Text • Image - Image Job information • Requirements - Rich Text • Location - Droplist • Hours - Droplist • Job Type - Droplist	Applications/32x32/document_pinned.png
Footer	Data • Main item - Droplink • Copyright - Multi-Line Text • Other links - Multilist	Applications/32x32/selection_down.png

Lab 2.2: Cleaning Our Templates

Now that you know about template inheritance, you need to clean up your template design to avoid redundant fields.

2.2.1: Create base templates

Detailed Steps

1. Using the Content Editor, select the */sitecore/Templates/Jobs* folder you created in the earlier lab.
2. Create a new template named *_Base Page*.

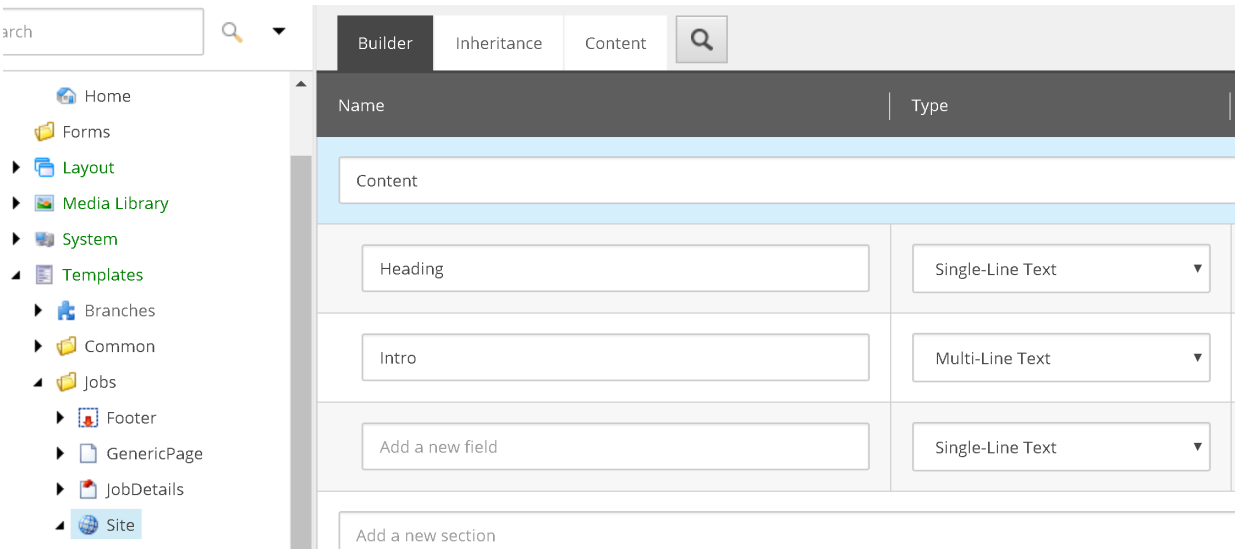
3. Add a field section called *Content* on the Builder tab.
4. Add the following fields:
    - **Heading** - Single-Line Text
    - **Intro** - Multi-Line Text
5. **Save** your work by pressing **Ctrl+S.**
6. Create another template named *_Base Image.*
7. Create a field section called *Content* and add a field named *Image* of type **Image**.
8. **Save** your work.
9. Create a new template named _Base Description.
10. Create a field section *Content* and add a field named *Description* of type **Rich Text.**
11. **Save** your work.

2.2.2: Add inheritance to the existing templates

Remove the duplicate fields you created earlier and instead make the templates inherit from the base templates that you just created.

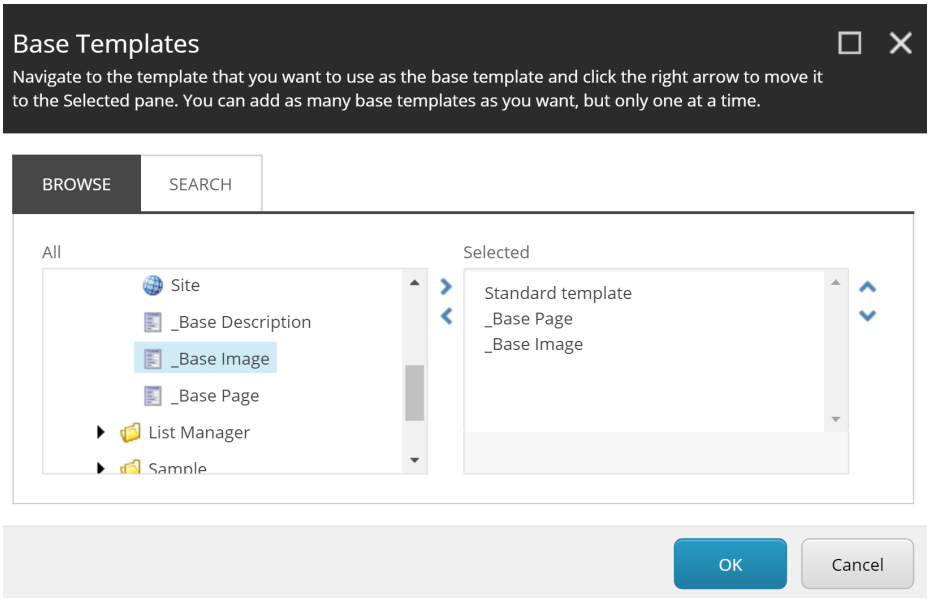
Detailed Steps

1. Select the template **Site.**
2. Select the **Content** field section by clicking in its textbox - it should be highlighted in blue.



3. On the **Builder Options** tab, click **Remove.** You should see that the **Content** field section and its **Heading** and **Intro** fields disappear.

4. Click the **Base templates** in the **Builder Options** tab.
5. Expand the tree and double-click the **/Templates/Jobs/_Base Page** template.
6. Double-click the **_Base Image** template.



7. Click **OK** to close the dialog.
8. Press **Ctrl+S** to save the changes.

You will now have to do the same for the **GenericPage** and **JobDetails** template. Remove the **Content** field section from both of them and set the inheritance as follows:

- **GenericPage** - inherits from _Base Page, _Base Description
- **JobDetails** - inherits from _Base Page, _Base Description, _Base Image

MODULE 3: Creating the Site's Content Structure

In this module, you will learn how to create content and how to support end users when they create content.

By the end of this module, you will be able to:

- Define a Sitecore item.
- Create content items based on templates.
- Use the Standard Values to set default values and settings.
- Explain the importance of insert options.
- Describe different content versioning options in Sitecore.

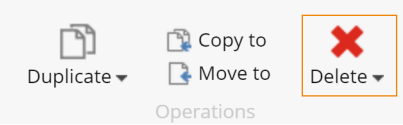
Lab 3.1: Creating Content as an Admin User

In this lab, you will create the basic scaffolding of the site so the users can create their own content. You will create the Home item and the basic structure of the site.

3.1.1: Create the scaffold of the site

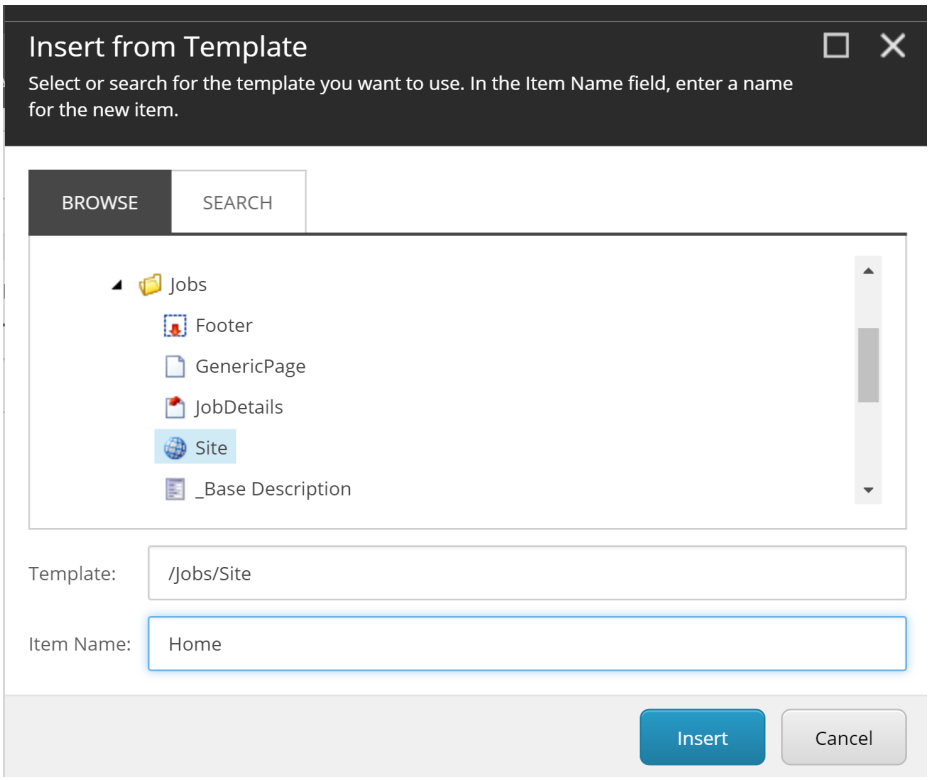
Detailed Steps

1. Log in to the **Content Editor** and select the **/sitecore/content/home** item.
2. Click **Delete** on the **Home** tab.

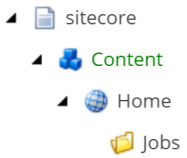


3. Click **OK** in the confirmation dialog.
4. The **Breaking Links** dialog will appear. Select **Remove links** and click **Continue** twice.
5. Click the **Insert from template** option in the **Home** tab.

6. Expand the tree to select the **Site** template you created earlier. In the Item name textbox, add *Home.*



7. Click **Insert** from template again to create another item under the **Home** item you just created.
8. Select the template **Templates/Common/Folder** and name the item *Jobs.*



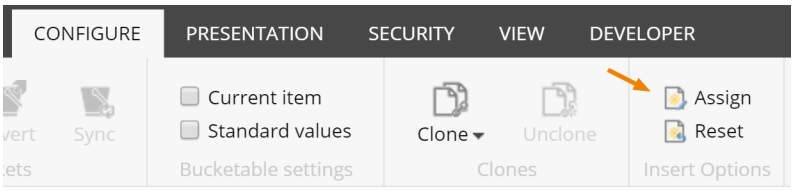
3.1.2: Setting the insert option

You now have the scaffold of the site, however, your editors will not be able to create content. Remember, they do not have the *Insert from template* option, so you will have to set insert options for them to be able to create new items.

Detailed Steps

1. Select the */sitecore/Content/Home* item in the **Content Editor**.

2. Click **Assign**on the Configure tab.



3. A dialog will appear. Find the **GenericPage** template, double-click to select it and then click **OK**.
4. Select the **Home** tab and notice that in the Insert group it now says, *GenericPage*. Users will now be able to create content
5. Select the **Jobs** folder under the **Home** item.
6. On the **Home tab**, click **Insert** (the title of the Insert group). Notice that it also opens the **Insert Options** dialog.

7. Double-click on the **Folder** selection on the right pane to remove it.
8. Search for **JobDetails** on the left pane and double-click to add it as the insert option.
9. Close the dialog by clicking **OK**.

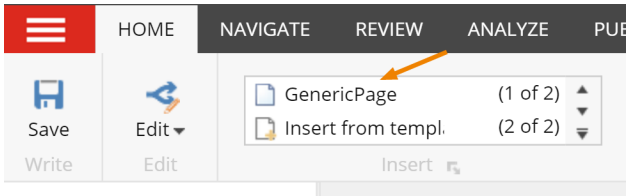
Lab 3.2: Creating Content as a User

In this lab, you will create content using the same steps that a regular user would follow.

3.2.1: Creating content using insert options

Detailed Steps

1. Select the**/sitecore/Content/Home** item.
2. Click the **GenericPage** option on the Home tab.



3. Give the item the name *About Us* and click **OK**.

4. Fill the **Heading** field with the text *About Us*.
5. Click the **Show Editor** link in the **Description** field.
6. Enter two paragraphs of filler text (use a generator like https://www.lipsum.com/).

> Notice that the Rich Text editor permits modifying the formatting of the text. The default configuration of the editor, as shown here, has reduced functionality, but it is possible to enable many more functions: snippets, tables, class selectors, etc.

7. Click **Accept**.
8. Click **Save** on the toolbar to store your changes.

3.2.2: Other ways of creating content

Detailed Steps

1. Select the /sitecore/Content/Home/About Us item.
2. Click on the **Duplicate** button on the **Home** tab.
3. Name the item *Legal Notice* and click **OK.**
4. Notice the field values have been copied over. Change the **Heading** field to *Legal Notice*.
5. Click **Save**.

Editors can also use *Copy* to duplicate the item in a different location.

Lab 3.3: Setting Field Sources

In this lab, you will finish creating the rest of the items required for the site. Remember that you need to provide users with the basic scaffolding of the site, the main sections, folders and so on where they will then create their content.

Detailed Steps

1. In the **Content Editor**, navigate to the */sitecore/content* item.
2. Click the **Insert from template** button to create new content.
3. Choose the template **/Common/Folder** and name it *Global.*

> Usually you find a folder outside of the Home item that contains items that *are not* actual pages of the site.

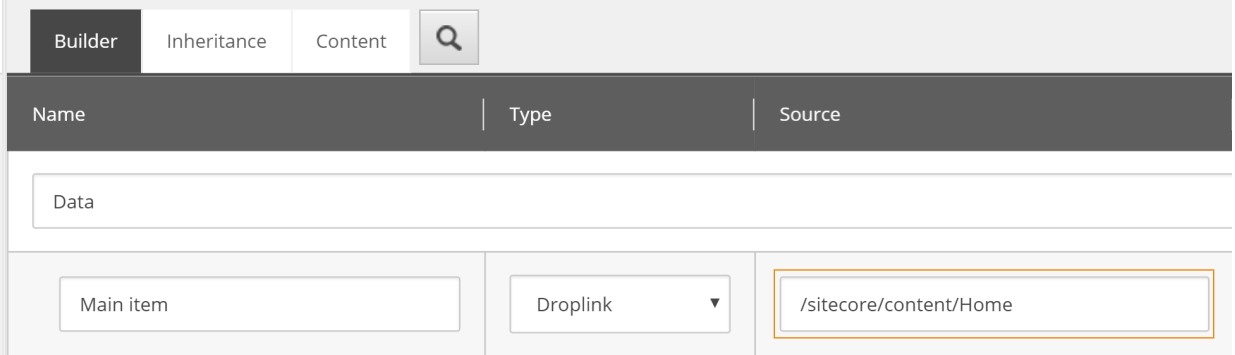
4. With the new *Global* item having the focus, use **Insert from template** to create another item, based on the **Footer** template named *Footer*.
5. Enter the data in the Copyright field: *Copyright © 2018 Version 1.0.0.0*
6. **Save** your changes.

3.3.1: Setting field sources

Notice that the other fields in the Footer item, which are droplink and a multilist, do not allow you to set any values. This is because they require the source to be set on them.

Detailed Steps

1. Go to the **Footer** template */sitecore/templates/Jobs/Footer.*
2. Set the value of the **Source** for the **Main Item** field to point to */sitecore/content/home.*



3. Do the same for the **Other Links** field.
4. **Save** your changes by pressing **Ctrl+S.**
5. Go back to the **Footer** item */sitecore/content/Global/Footer***.**
6. In the **Main Item** field, select *About Us* from the drop-down menu.
7. In the **Other Links** field, select **Legal Notice** by double-clicking it.
8. **Save** your changes by pressing **Ctrl+S.**

Lab 3.4: Default Field Values

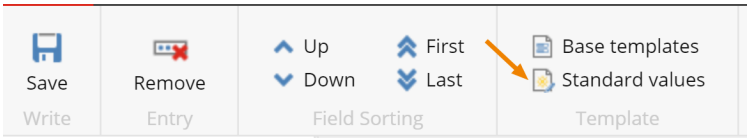
In this lab, you will create some Standard Values for the templates you have already created and set some default field values. You will then create new content and see the effect of the default values assigned.

3.4.1: Creating Standard Values

Detailed Steps

1. In the **Content Editor**, select the **_Base Page** that you created earlier.

2. On the **Builder Options** tab of the toolbar, click **Standard Values**.



   This will create a brand new item named __Standard Values as a child of the template.

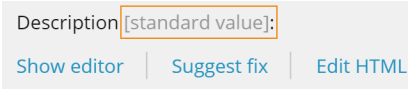
3. In the **Heading** field enter the text: *$name*.
4. In the **Intro** field add 1 paragraph of *lorem ipsum*.
5. Press **Ctrl+S** to save your changes.
6. Repeat the same steps to create **Standard Values** for the **_Base Description** template. Add two paragraphs of *lorem ipsum* to the **Description** field. Don't forget to save your changes.

3.4.2: Creating content, overriding field values and resetting them

Detailed Steps

1. Create a new item named *Privacy Policy* based on the **GenericPage** template under the **Home** item.

   Notice the Heading field has been initialized with the name of the item. The Intro and Description fields both have *lorem ipsum* text and the [standard value] flag next to the field name indicates it is coming from the Standard Values item.



2. Delete the contents of the **Intro** field and **save** your changes.

   Notice the field is now empty, and the [standard value] indicator has gone. Remember empty and null are not the same thing.

3. On the **Versions** tab of the toolbar, click **Reset**.

   This dialog shows you the current values for all the fields of the item and the defaults set in the Standard Values.

4. Select the check box beside the **Intro** field and click **Reset**.

   Now the value of the field is *null* and therefore it shows the Standard Values once more.

5. Navigate to the **Home** item.

> Notice the Heading field shows $name. Remember that tokens are only replaced when you create the item. This item existed before we added the token, therefore it displays the $name token text as a default value.

6. Change the value of the **Heading** field to *Careers* and save the item.

Lab 3.5: Finish the Scaffolding of the Site

In this lab, you will finish completing all the remaining items you need for the site.

You have created a few fields in the Job Details template that are also depending on other items: Location, Hours and JobType. You did this to create a better editing experience for your user. The regular users will have drop-down menus to choose from a set collection of values. Those are defined as items. Before you create them, you have to make a template for those items.

3.5.1: Create templates

Detailed Steps

1. Open the **Content Editor** and navigate to the **/sitecore/Templates/Jobs** folder.
2. Create a new template named *Option* (See module 2 about creating templates).
3. Set the icon as *Applications/32x32/bullet_ball_glass_red.png.*
4. **Save** your changes.
5. Create another template named *Options Folder.*
6. Set the icon to *Applications/32x32/folder_blue.png.*
7. **Save** your changes.
8. Create a new **Standard Values item** for the Options Folder template.
9. Set **Option** as the insert option for the Options Folder.

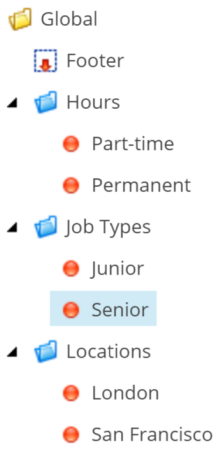
3.5.2: Create some scaffolding folders and sample items

Detailed Steps

1. Navigate to the **/sitecore/content/Global** item.
2. Add the **Options Folder** as an insert option for the **Global** item.
3. Create three items under Global based on the **Options Folder** template. Name them: *Locations*, *Hours* and *Job Types*.

> **NOTE:** You can use the Insert option directly and do not need to use *Insert from template.*

4. Create sample items under each of the folders.

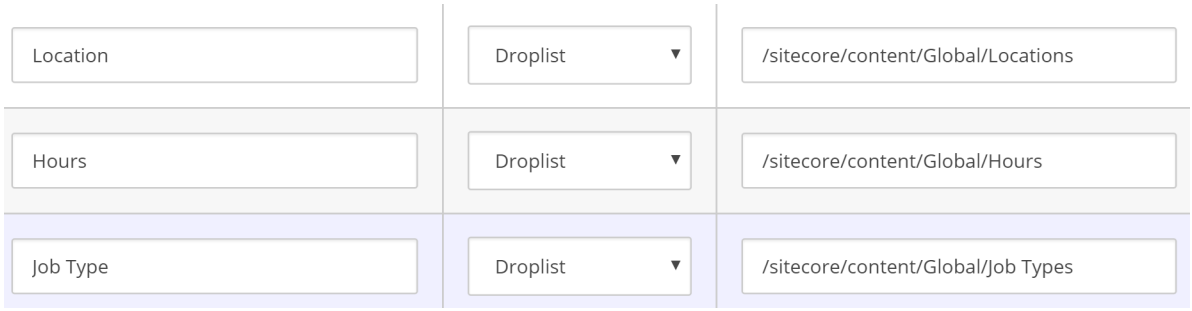


3.5.3: Set field sources for the Job Details template

In order to finish your initial configuration of the content structure, you need to set the field source for the droplist fields you created in the Job Details data template.

Detailed Steps

1. Open the **JobDetails** template.
2. Set the **field source** for the Location, Hours and Job Type as follows:



3. **Save** your changes.
4. Navigate to the **/sitecore/content/Home/Jobs** item.
5. Create a new item named *Web Designer* based on the **JobDetails** template.
6. Set field values for all fields. You should be able to select values from the drop-downs in the Job Information field section.

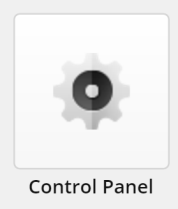
Lab 3.6: Working with Item Versions

In this lab, you will define a new language. You will then create some content in different versions to familiarize yourself with the options available.

3.6.1: Adding a new language

Detailed Steps

1. From the Launchpad open **Control Panel**.

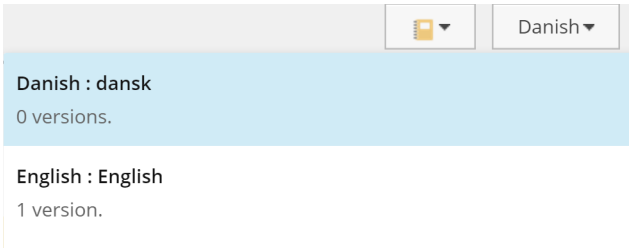


2. In the **Localization** section, select **Add a new language**.
3. From the drop-down select **Danish : dansk**.
4. Click **Next** twice.
5. **Delete** the contents of the *Spellchecker file name* textbox if anything is specified in it.
6. Click **Next** and then **Close**.

3.6.2: Create item language versions

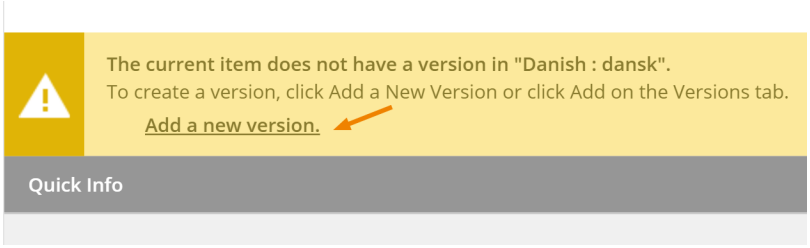
Detailed Steps

1. From the Launchpad, open the **Content Editor** and ensure that you are on the Home item.
2. Use the language drop-down from the right-hand side of the Content Editor and select **Danish**.



The language drop-down tells you that there is one version in English and no versions in Danish. You do not automatically have versions in all languages, you must add them as needed.

3. Click the **Add a new version** link inside the warning.



4. Set the **Heading** field with the value *Karriere*.
5. Save your changes.
6. Switch between the two languages using the language drop-down and notice how the heading field changes value.

3.6.3: Adding language versions to the Standard Values item

The **Intro** field has a default value (lorem ipsum) in English but it is empty in Danish. This is because the Standard Values item, just like any item, can also have language versions. When you created the Standard Values earlier you set values on its English version; if you want default values in Danish you need to create a Danish version.

Detailed Steps

1. Navigate to the Standard Values item of the *_Base Page template (/sitecore/templates/Jobs/_Base Page/__Standard Values).*
2. Use the Language drop-down to select **Danish**.
3. Click the **Add a new version** link.
4. Fill the heading field with $name and add *lorem ipsum* to the Intro field.
5. **Save** your changes.
6. Navigate back to the **Home** item. Notice how the Intro field now shows a default value.

NOTE: It is possible to set fallbacks for languages - this way if a value is not defined in a language it could use the content from a different language. You can find out more on the Sitecore documentation site (http://doc.sitecore.net).

MODULE 4: Working with Sitecore Publishing

In this module, you will learn how to publish to make content available in the content delivery environment. You will become familiar with the end user experience in Sitecore and learn about publishing options and restrictions.

By the end of this module, you will be able to:

- Use the publishing feature.
- Describe publishing options.
- Set publishing restrictions.
- Describe the purpose of the Core, Master and the Web database.
- Describe a publishing target.

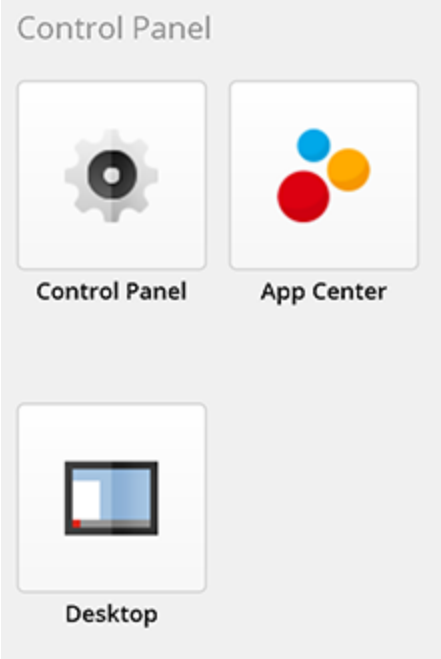
Lab 4.1: Viewing Content in Different Databases

In this lab, you will check the content in the other Sitecore databases using the Desktop interface.

4.1.1: Looking at the content of the Web database

Detailed Steps

1. From the Sitecore **Launchpad,** open the **Desktop** interface (the icon is in the Control Panel section).



2. In the bottom-right corner, click the database icon to switch to the **Web** database.

3. Click the **Sitecore logo** in the bottom-left corner to open the Sitecore menu.

4. Choose the **Content Editor** from the Sitecore menu. This is the same interface you used previously, except it is now on the Desktop instead of using the full browser tab.

   Notice that the content and templates that you have created are not there. This is because you are now looking at the contents of the Web database, not the Master database, which is where you have made your changes.

   > **NOTE:** You should *never* make changes directly to the Web database because the changes will be overridden during the next publish operation. However, it is still useful to be able to access the Web database if you ever need to check if content has actually been published or not.

Lab 4.2: Publishing Your Content

In this lab, you will perform a publish operation and check the effect it has on the content of the Web database.

4.2.1: Publishing

Detailed Steps

1. Ensure you area still on the **Desktop** interface
2. Use the database selector to select the **Master** database
3. Press **F9** to open the Publishing dialog.
4. Select **Incremental** as the publishing mode.
5. Click **Select all** to publish all languages.
6. Click **Publish**.
7. When the publishing report appears, click **Close**.
8. Use the database selector and choose the **Web** database. Open the **Content Editor** and verify that it now contains all the changes you have made so far.
9. When you have finished, use the database selector and select the **Master** database again.

   > **NOTE:** You need to select the **Master** database again, otherwise the changes in the following labs would only be made on the Web database.

Lab 4.3: Enabling Live Mode

In this lab, you will enable **live mode**. This will make Sitecore extract the data for the live site from the Master database and ignore the content in the Web database. This means when you start developing the presentation later on you will not have to do any publishing to see changes reflect in the live site.

Detailed Steps

1. Open the folder with the webroot of the Sitecore installation located at

C:\inetpub\wwwroot\jobs.tac.local\Website.

2. Navigate to the *App_Config\Include\Examples* folder.
3. Locate the *LiveMode.config.example* and rename it *LiveMode.config.*

> The files inside the App_Config\Include folder modify the web.config. The main advantage of using include files is that you can better control configuration changes between environments and you also make deployments and upgrades a lot easier.

MODULE 5: Creating and Applying Presentation

In this module, you will learn about layouts, components, and placeholders. You will also learn how to set presentation through the presentation details of an item or standard values. You will also learn how to change the presentation through the Experience Editor.

By the end of this module, you will be able to:

- Describe how Sitecore layouts and components are tied to the ASP.NET MVC framework.
- Retrieve field values from Sitecore and make the fields editable in the Experience Editor.
- Explain the difference between inline-editable and non-editable field types.
- Apply presentation to a Sitecore item or Standard Values.
- Define the context item.
- Set up a control or component to target an item that is not the context item.
- Identify components in a HTML template.
- Describe the presentation details and their related definition items.

Lab 5.1: Creating a Layout

In this lab, you will create a layout and add the static HTML from the templates to it. You will then bind it on the presentation details so it gets used to render the page response.

Detailed Steps

1. Use the **Content Editor** to navigate to the *sitecore/Layout/Layouts* item.
2. Use insert options to create a new item based on the template **Layout Folder** and name it *Jobs*.
3. Inside this folder, create a new item based on the MVC Layout template. Name it *Default*. Click **Next**.
4. Select the **Jobs** item and click **Next**.

   This is to ensure the item is created inside the folder that you have created earlier. It is best practice to keep all your items inside folders. This makes them much easier to identify and move to other solutions.

5. Select the **Views** folder and click **Create**.

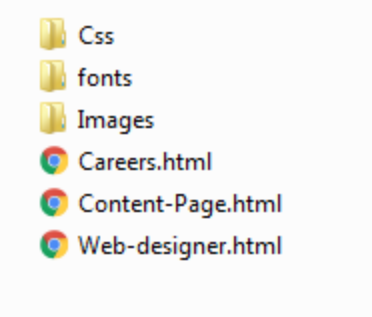
   You are selecting a location in the file system. Sitecore will now create both the definition item and the file.

6. Click **Close** to finish the wizard.
7. Verify that the **Path** field contains the location of the file that you just created.

5.1.1: Edit the layout file

Detailed Steps

1. Using your text editor of choice (Visual Studio Code or similar) open the **Careers.html** from the Student Resources folder provided with this course.



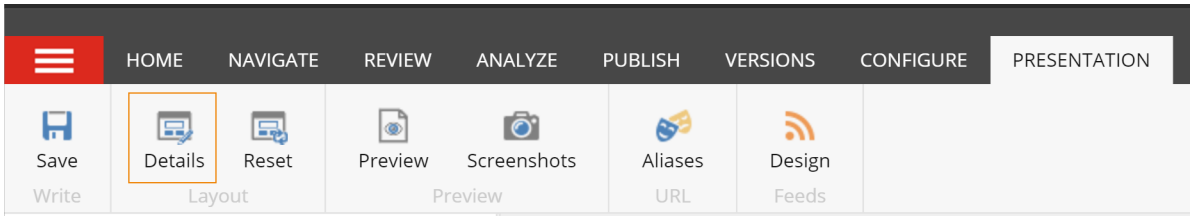
2. Copy everything in it to your clipboard.
3. Open the **Default.cshtml file.** The file will be located inside the **Views** folder of the webroot (C:\inet-pub\wwwroot\jobs.tac.local\Website)
4. Delete everything from the `<!DOCTYPE>` directive to the end of the file and replace it with the contents of your clipboard.
5. **Save** the changes.

5.1.2: Assign the layout in the presentation details

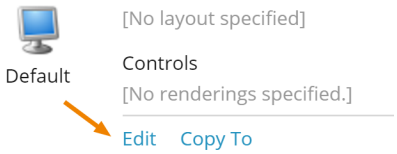
Now that you have created the layout and added the correct code, you can bind it to the presentation details.

Detailed Steps

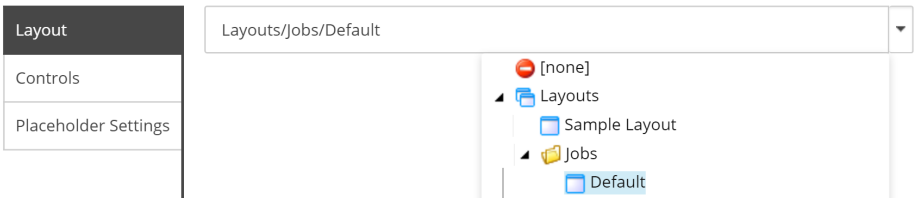
1. Open the Standard Value item for the **_Base Page** template located in */sitecore/templates/Jobs/_Base Page/__Standard Values.*
2. Click the **Details** button on the **Presentation** tab.



3. In the **Default** device click **Edit**



4. In the **Layout** tab, use the drop-down to select the **/Jobs/Default** layout



5. Press **OK** twice to close the dialog.
6. Using another tab in the browser, navigate to the http://jobs.tac.local page. Notice that a page is rendered. However, it is missing the CSS and Images.
7. From the Student Resources folder, copy the CSS, fonts and Images folders and paste them on the webroot of your solution(C:\inetpub\wwwroot\jobs.tac.local\Website).
8. Refresh the browser page and verify it now has all the styling.

Lab 5.2: Making Content Editable in the Experience Editor

In this lab, you will start making the site editable. You will replace some of the hardcoded HTML for content stored in Sitecore. You will then test how content is editable inline using the Experience Editor.

We now know that there is no magic to editing content in Sitecore. Editors will not be able to change the content just because there is HTML on the page. They change the content stored in Sitecore, therefore, you need to populate the HTML with the content from Sitecore items.

Rendering the content of inline editable fields (like text fields) is very easy. Other fields, like multilists, or droplinks, require more work because the way to render them depends on what you are using them for.

Detailed Steps

1. Using a text editor, open the **Default.cshtml** which is in the Views folder of your webroot (C:\inetpub\wwwroot\jobs.tac.local\Website)
2. Locate the **<title> tag** in the head section of the html

3. Replace the value inside the tag with

   @Html.Sitecore().Field("Heading",new {DisableWebEdit=true})

   > **NOTE:** You have added the extra parameter to disable inline editing. This field is rendered in the header of the HTML, inline editing would not make sense. It would also add strange code when in Experience Editor editing mode, and they would show in the title bar of the browser.

4. Locate the **<small> tag** (it's the only one in the file you can just search for it).
5. Replace the content inside with:

   @Html.Sitecore().Field("Intro")

   > ***NOTE:*** *In this case you have not added the extra parameter since you want the text to be editable inline.*

6. Just above the <small> tag you will find a <h1> tag. Replace the word *Careers* with

   @Html.Sitecore().Field("Heading").

7. Further up, there is a <header> tag and just below that, there is an <img> tag. Replace the entire <img> tag with:

   @Html.Sitecore().Field("Image",new {@class="image-background"}).

   > **NOTE:** You are adding the class attribute to render the class for the <img> tag. The field method will render the entire tag for you.

8. Save your changes.
9. On a browser tab open http://jobs.tac.local. Notice the content is now coming from Sitecore. There is no image since none is specified in Sitecore.

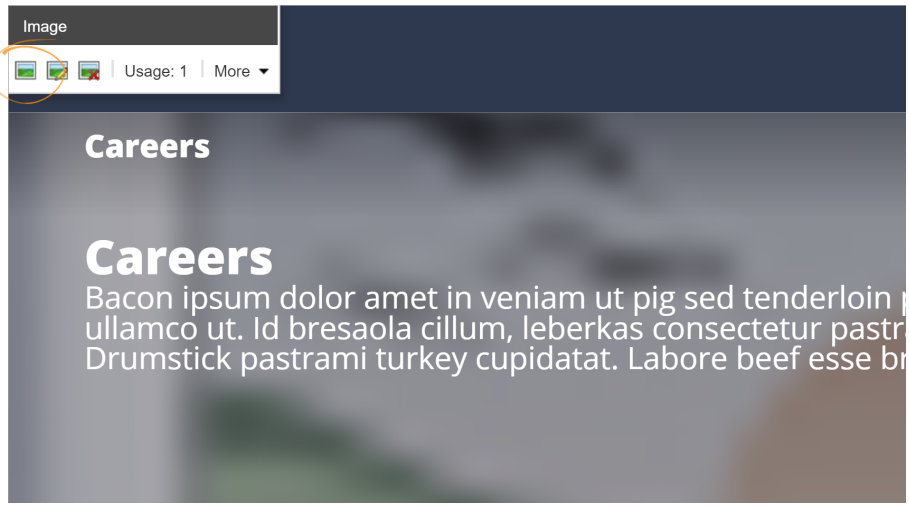
   > It is best practice to keep the site in the browser. You then simply need to refresh the page every time you make a change to the presentation.

5.2.1: Editing content in the Experience Editor

Detailed Steps

1. Open the **Experience Editor**. You can do it from the Content Editor (through the Publish tab) or from the Launchpad.
2. Click the **Intro text**. Notice that it is editable
3. Click the **banner image** just below the intro text. Even though there is currently no image, it is also editable.

4. In the Image contextual toolbar, click the first button **Choose an image**.



5. Click **Upload media** in the top-left corner of the dialog.
6. Click the **Browse for media files** and locate the career.jpg file which is inside the Images folder of the Web-root (C:\inetpub\wwwroot\jobs.tac.local\Website\Images). Alternatively you can also drop the file directly in the dialog.
7. Add the text *Careers* in the **Alternate Text** textbox.
8. Click **Upload media**.
9. Select the career image and click **Select**.
10. **Save** the changes made to the item.

Lab 5.3: Componentizing the layout

In this lab, you will split the HTML from the layout into smaller, more manageable and reusable units: components. You will then statically reference those components on the layout so they are still shown on the page.

5.3.1: Creating a View Rendering component

You will notice that the upper part of the design (delimited with the comment [Page Header] in the HTML) is actually the same in all the designs. You will now make it into a component.

Detailed Steps

1. In the **Views** folder of the webroot, create a new file named *PageHeader.cshtml*.
2. Using the **Content Editor** open the **/sitecore/layout/Renderings** item.
3. Create a new item using the template **Rendering Folder** (it is one of the insert options) and name it *Jobs*.
4. Inside this new folder, create a new item named *PageHeader* based on the **View Rendering** template (again it is one of the insert options).

5. In the **Path** field, enter the relative path of the view that you created, such as: */Views/PageHeader.cshtml.*
6. Save the changes on the item.

5.3.2: Splicing the HTML

Detailed Steps

1. Open the **Default.cshtml** file in the Views folder of the webroot.
2. Locate and **cut** the part of HTML between the comment tags <!-- [Page Header] -->.
3. Open the **PageHeader.cshtml** from the same folder and paste the code.
4. **Save** the changes on the file.
5. In the original location of the HTML you cut in step 2 add:

   @Html.Sitecore().Rendering("/sitecore/layout/renderings/jobs/pageheader")
6. Save your changes
7. Navigate to the home page using a browser and check the page is still displaying correctly.

5.3.3: Create other components

You can now follow the same instructions to create other components. Splice the HTML delimited by the <!-- [Footer Content] --> comment and add it to a component named *FooterContent*.

Lab 5.4: Adding Placeholders and Definition items

In this lab, you will continue to create components to splice the HTML in the layout. Because you will work with the part of the design that changes between the three different pages, you will have to create placeholders and use dynamic binding.

The next part of the page you need to componentize, accounts for most of the code you have left in the layout and is delimited by the [PageContent] comment. You will notice that the structure is different in the home page (Careers.html) compared to the other two types of pages. This is why you will have to create a placeholder, so you can add the correct HTML depending on the type of page requested. Using dynamic binding you will be able to re-use the same layout for all pages.

5.4.1: Creating more components

Create **3** new **ViewRendering** components named *HomePageContentHeader*, *FeaturedJob* and *JobList*. You can look at precise instructions for creating a View Rendering in the previous lab. To review: for each component you create you need to:

Detailed Steps

1. Create a new .cshtml file in the **Views** folder of the webroot.

2. Create a new **ViewRendering** item under */sitecore/layout/Renderings/Jobs*.
3. Make sure the **Path** field of the item contains the relative path to the file in the webroot.

5.4.2: Componentizing the rest of the layout

Now you are ready to cut the rest of the HTML from the layout and place it on smaller components.

Detailed Steps

1. Cut all of the content delimited with the comment **[Title and Summary]** (i.e. a <header> with everything it contains) from the layout (Default.html) and paste it on the **HomePageContentHeader** component.
2. Add a placeholder at the location of the layout where you just cut the HTML. Remember that to add a placeholder you just need to give it a key, in this case use the key *pageContentHeader*:

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

3. Cut the content delimited by the [Job List] comment and paste it on the JobList.cshtml view.
4. In the **JobList.cshtml** file, cut out the first div ( ) and remove the last div closing tag ( ).
5. Go back to the layout (Default.cshtml) and paste the in the original location of the code you just cut. Add a placeholder with the key *pageContentMain*:

   ```
   <div class="col-md-8">

   @Html.Sitecore().Placeholder("pageContentMain")

   </div>
   ```

6. Cut the content delimited by the [**Job Spotlight**] comment and paste it on the **FeaturedJob.cshtml** view.
7. In the **FeaturedJob.cshtml** file, cut the first div (`<div class="col-md-4">`) and remove the last div closing tag.
8. Go back to the layout (Default.cshtml) and paste the in the original location of the code you just cut. Add a placeholder with the key *pageContentRight*:

   ```
   <div class="col-md-4">

   @Html.Sitecore().Placeholder("pageContentRight")

   </div>
   ```

9. Make sure you save your changes in all the files.

Congratulations! You have finished componentizing the layout. You now have your scaffolding HTML. The <body> of your layout should look as follows:

```
<body class="header-static ">
    <div id="main-container">
    <!-- [Page Header] -->
        @Html.Sitecore().Rendering("/sitecore/layout/renderings/jobs/pageheader")
    <!-- [/Page Header] -->
    <!-- [Page Content] -->
        <main role="main">
      <!-- [Title and Summary] -->
          @Html.Sitecore().Placeholder("pageContentHeader")
      <!-- [/Title and Summary] -->
      <section class="section section-full">
        <div class="container">
          <div class="row">
          <!-- [Job List] -->
          <div class="col-md-8">
            @Html.Sitecore().Placeholder("pageContentMain")
          </div>
          <!-- [/Job List] -->
          <!-- [Job Spotlight] -->
          <div class="col-md-4">
            @Html.Sitecore().Placeholder("pageContentRight")
          </div>
          <!-- [/Job Spotlight] -->
          </div>
        </div>
      </section>
        </main>
    <!-- [/Page Content] -->
    <!-- [Footer Content] -->
        @Html.Sitecore().Rendering("/sitecore/layout/renderings/jobs/footercontent")
    <!-- [/Footer Content] -->
    </div>
</body>
```

If you try to view the page on the browser, you will see the middle part of the page is missing. This is because you have replaced the code with placeholders. In the next lab, you will assign components to those placeholders and restore the original look of the page.

Lab 5.5: Assigning the components to the placeholders

In this lab, you will use the presentation details to assign components to placeholders. This will enable you to reconstruct the original look and feel of the page.

5.5.1: Setting presentation details

You will now follow recommended practices and assign components to placeholders in the presentation details of the Standard Values item. This will configure it as a default, so any items based on that template will have the same presentation details, unless they are overridden.

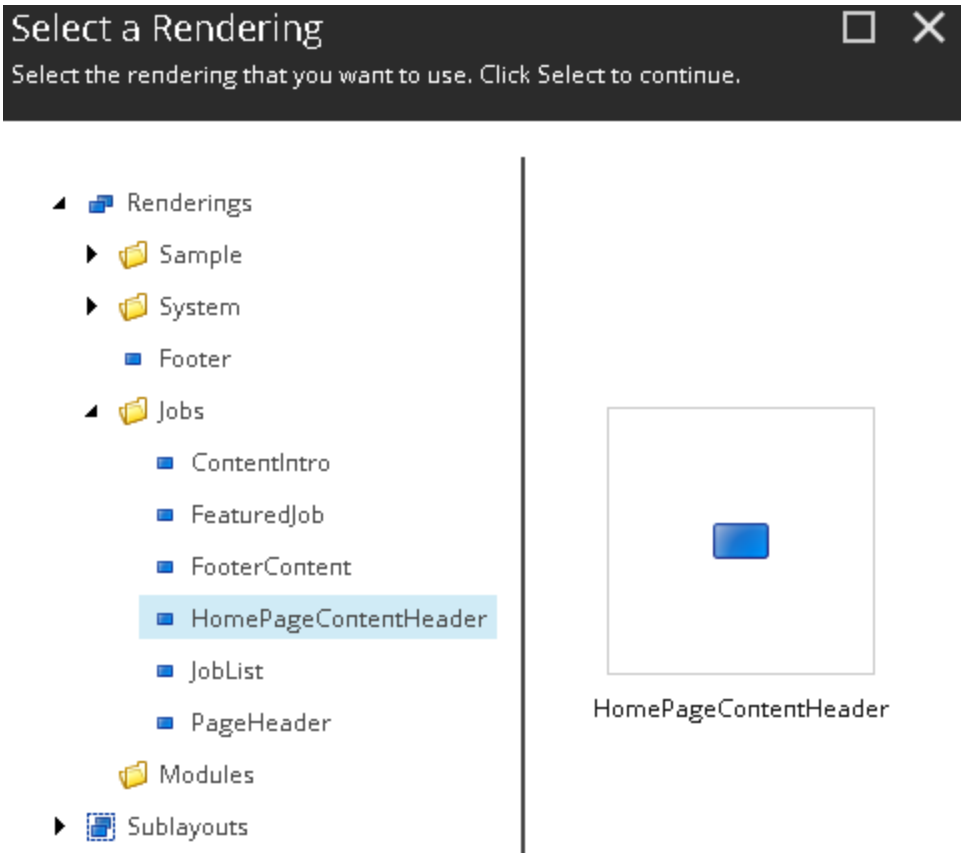
Detailed Steps

1. In the **Content Editor** open the **Standard Values** for the Site template (*/sitecore/Templates/Jobs/Site/__Standard Values*).

   > **NOTE:** If the Standard Values item does not exist, you can create it. Select the template and use the Standard Values button in the Builder Options tab.

2. Open the presentation details by clicking on the **Details** button on the **Presentation** tab.
3. Click on the **Edit** command for the **Default** device.
4. Select the **Controls** tab.
5. Click **Add**.

6. Select the **Renderings/Jobs/HomePageContentHeader** component, and add it to the placeholder *pageContentHeader*.



7. Click **Select**.
8. Repeat steps 5 - 7 again to add the **JobList** component to the **pageContentMain** placeholder.
9. Repeat once more to add **FeaturedJob** to the **pageContentRight** placeholder.
10. Click **OK** twice to return to the Content Editor.

You should now be able to see the entire page again if you navigate with your browser to http://jobs.tac.local.

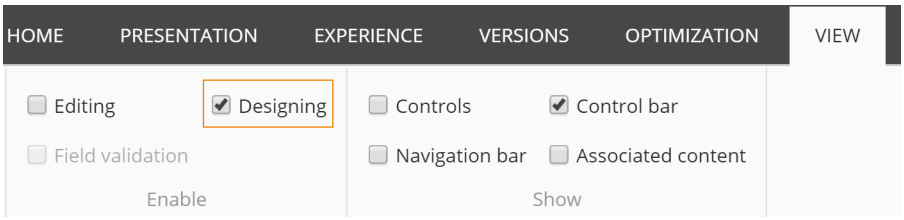
Lab 5.6: Setting the allowed controls

In this lab, you will test what happens to your page when it is viewed in the Experience Editor. In particular you will see how the designing features are severely limited unless you configure them through Placeholder Settings.

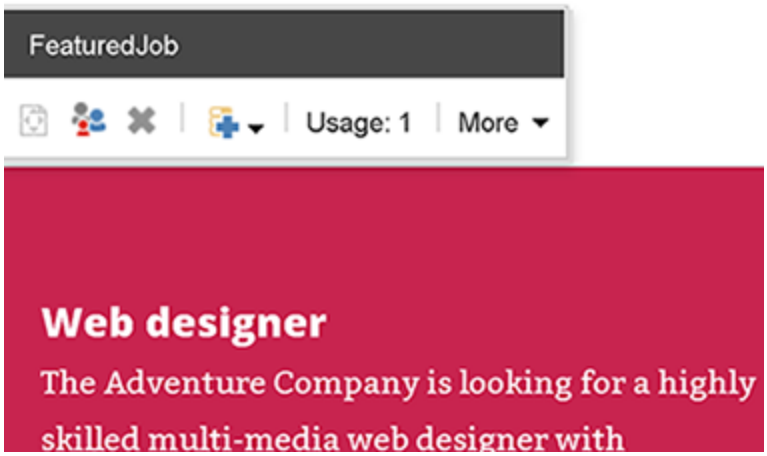
5.6.1: Using the Experience Editor

Detailed Steps

1. Open the **Experience Editor.** You can do it from the Sitecore Launchpad or from the Publishing tab of the Content Editor.
2. In the **View** tab, ensure that **Designing** is checked.



2. Click on the **Featured Job** on the right hand side of the page



Notice that the feature to remove this component is not available. This is because currently none of the placeholders are editable, since you have not created any placeholder settings items. When a placeholder is not editable you can't add or remove components from it using the Experience Editor designing mode.

5.6.2: Create a placeholder settings item

You will now create a placeholder settings item to enable the ability to add and remove components from the placeholder through the Experience Editor.

Detailed Steps

1. In the **Content Editor** open the item */sitecore/Layout/Placeholder Settings*.
2. Create a new item named *Jobs* with the template **Placeholder Settings Folder** (one of the insert options).

3. Under this new item Jobs, create a new item named *pageContentRight* based on the template **Placeholder** (one of the insert options).
4. Scroll down to the **Allowed Controls** field and click on the **Edit** command.

   > **NOTE:** This is an example of a TreelistEx field - it only shows the tree to select content from when you click the Edit button. This saves resources when the item is just rendered in the Content Editor, and does not need to be edited. Most of the time, it is better to use TreelistEx fields instead of Treelists.

5. Double click on the *Layout/Renderings/Jobs/FeaturedJob* item to select it.
6. Click **OK** to close the dialog.
7. Save your changes.

5.6.3: Add and remove a component through the Experience Editor

If you repeat the steps in the first part of the lab to select the *FeaturedJob* component in the Experience Editor, you will notice that now the *Remove component* button is now enabled.

> **NOTE:** If your Experience editor was already open, simply reload the page to reflect the changes made on the Placeholder Settings.

Detailed Steps

1. From the **Experience Editor**, having selected the *FeaturedJob* component click on the **Remove Component** button. Notice the component disappears and is replaced by a hashed, gray area. This is an empty placeholder.
2. Click on the grey area. A toolbar appears telling you the name of the placeholder, *pageContentRight*, as well as an **Add here** button.
3. Click on the **Add here** button to add a new component.
4. Notice Sitecore gives you a list of available components. This list is populated with the contents of the Allowed Controls field you set earlier in the Placeholder Settings item.
5. Select the **FeatureJob** component and click **Select**.
6. The component should reappear on the page.
7. Click save on the top left of the page (otherwise the browser will prompt you to save changes next time you refresh the page).

Lab 5.7: Completing the JobList component

In this lab, you will remove all the static content from the JobList component. To do that you need to know how to do two things: get the children of an item and get the URL of an item.

Detailed Steps

1. Open the **JobList.cshtml** view.

2. Remove all the <li> tags except one.
3. Surround the remaining <li> with a foreach look as follows

```
<ol class="media-list">
    @foreach (Sitecore.Data.Items.Item item in Model.Item.Children["Jobs"].Children)
    {
        <li class="media">
            ...
        </li>
    }
</ol>
```

4. Replace the static text inside the <li> tag with the contents of the **Heading** and **Intro** field from the item object in the foreach loop:

   e.g. `@Html.Sitecore().Field("Heading", item)`

5. Finally replace the URL in the <a> tag for a call to the LinkManager, to get the item's URL.

   `<a href="@Sitecore.Links.LinkManager.GetItemUrl(item)">`

6. **Save** the changes to the file.
7. Open the homepage in the browser and notice the content now comes from Sitecore.
8. Using the **Content Editor** create a new **JobDetails** item named *HR Manager* and check that it appears on the page.

Lab 5.8: Applying a Data Source to an Existing Component

In this lab, you will set the data source of a component using the Experience Editor. You will then modify the component's code so it uses that datasource to retrieve the content to be displayed.

Frequently, the content that you actually want to display, is not found in the context item. For example, on the Home page of the site, the *FeaturedJob* component promotes one particular job. The information about the job it displays should not be stored in the Home page item because it is not related to the Home page. Furthermore, the information is already displayed in the item describing that job.

What you really need to do is to tell the component to get the data from a different item. However, you also want to give users the ability to choose which job to promote. That is what data sources allow in components; enabling users to select a different item to be used in the output of the component.

5.8.1: Changing the component's code to show field values

The FeaturedJob component still has hard coded HTML so you will replace it with references to the Field method.

Detailed Steps

1. Using a text editor, open the **FeaturedJob.cshtml** View

2. Replace the static text inside the <h4> with a call to the *Field()* method to render the **Heading** field.

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

3. Replace the static text inside the <p> tag to render the contents of the **Intro** field.
4. **Save** your changes.
5. Open the **Home** page of the site in the browser.

Notice how the information displayed comes from the Home item itself. This is because the)data source has not been set, so it defaults to the current context item.

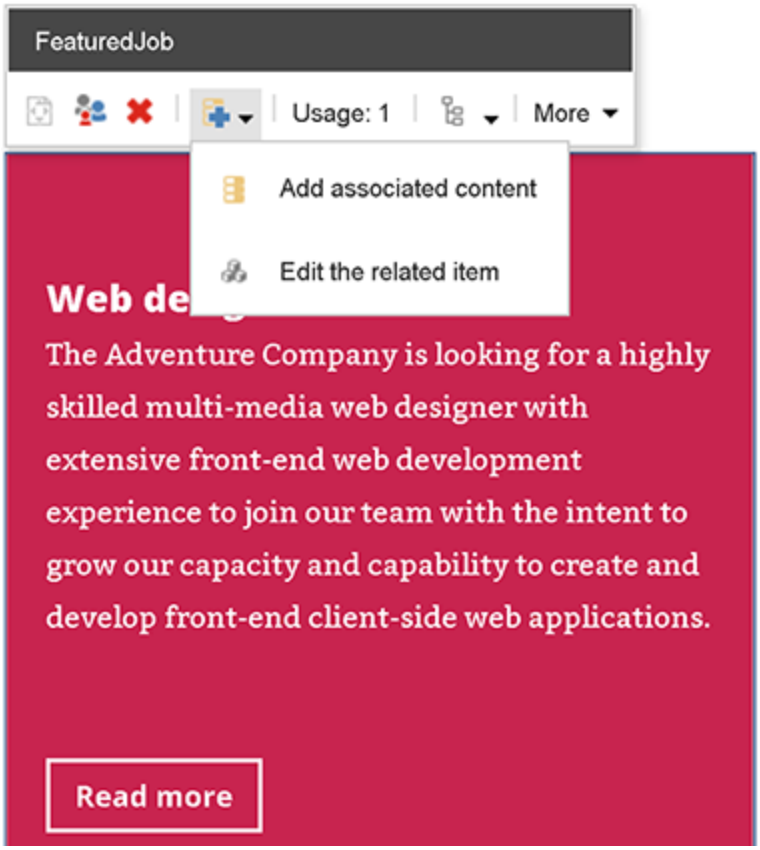
5.8.2: Setting the data source of a component

Now you will set the data source of the component. This can be done while editing the presentation details. You can do it here using the Experience Editor, just like an editor would.

Detailed Steps

1. Open the **Experience Editor**.
2. Ensure **Designing** mode is enabled (checkbox in the **View** tab).
3. On the Home page, select the **FeaturedJob** Component.

4. The component toolbar will appear. Click the **Add associated content** button.



5. In the *Select the Associated Content* dialog, expand the tree and select one of the job items (for example: /sitecore/Content/Home/Jobs/Web Designer).
6. Click **OK** to close the dialog.
7. Notice how the component now displays the content from the job item. Click **Save** in the top-left corner.

Lab 5.9: Datasources in Static Components

In this lab, you will explore how data sources can be useful even in components where editors don't have the ability to change the component. You will use a data source to complete the footer component.

The footer component you have created still has hard coded HTML. Its content needs to always come from the footer item you created previously (/sitecore/content/Global/Footer).

You could hard code this reference in the component itself. You have a few other options you will explore in this lab.

5.9.1: Setting the data source in a static component

Detailed Steps

1. Go to the **FooterContent.cshtml** file and replace the *Copyright text* in the bottom of the file (inside the <small> tags) for the contents of the Copyright field:

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

2. **Save** your changes.
3. Navigate to the Home page of the site in your browser. Notice how the copyright text is not shown. This is because it is looking for it in the context item (the home item) since the data source is not set.
4. In the Default.cshtml layout file, find the line where you statically add the footer component and change it to be as follows:

   ```
   @Html.Sitecore().Rendering("/sitecore/layout/renderings/jobs/footercontent",
   new {DataSource="/sitecore/content/Global/Footer"})
   ```

   You are specifying a data source for the component, even though it is added statically.

5. **Save** the changes and refresh the Home page in the browser.

   Notice how now the copyright text is showing on the page.

5.9.2: Adding a data source through configuration

You still have a hard coded reference in your code to the Footer item. It has been moved to the layout, but it would be even better if you could remove it altogether.

Detailed Steps

1. In the **Default.cshtml** layout, remove the new{} object you added in step 4 earlier.

   ```
   @Html.Sitecore().Rendering("/sitecore/layout/renderings/jobs/footercontent")
   ```

2. In the Content Editor navigate to the definition item of the footercontent component (/sitecore/layout/renderings/jobs/footercontent).

3. Locate the **Data source** field

> **NOTE:** In some versions of Sitecore, there is a small bug where this field is named *Data source*. If that is the case, you should rename it to remove the space. You can use the following steps to do this:
>
> 1. In the **Configure** tab, **Template** group, click the **Edit** button.
> 2. In the Template Manager dialog pop up, locate the **Data source** field: *Templates/System/Layout/Renderings/View rendering/Data/ Data source*
> 3. Rename it **Datasource**
> 4. Click the **Save** button.
> 5. Press the cross in the top right to abandon the Template Manager.

4. Enter the path (or Id) of the Footer item (/sitecore/content/Global/Footer).
5. **Save** the changes.
6. Navigate to the Home page in the browser. Notice the copyright text is still there.

5.9.3: Completing the Footer Component

You will now complete the rest of the code for the footer component. Right now most of the component is still hard coded, except for the copyright text. You will add some code so all the content comes from Sitecore.

The component has two parts. The left-hand side shows the heading and intro of one item (which is referenced from the *Main Item* field of the Footer item) and a series of links is displayed on the right hand side (which should point to the items referenced in the *Other Links* field).

You can follow these steps to add the rest of the content.

5.9.4: Reading the main item field

Detailed Steps

1. Open the **FooterContent.cshtml** file for editing.
2. At the top of the file, add a code block with the following content

```
@{

Sitecore.Data.Fields.ReferenceField mainitemfield = Model.Item.Fields["main
item"];

var mainitem = mainitemfield.TargetItem;

}
```

3. Find the static text (it's insde the tag) showing the *About us* text, and replace it with references to the "heading" field and "intro" field from the item represented by the mainitem variable.

   e.g. `@Html.Sitecore().Field("heading",mainitem)`

4. **Save** the file and verify the content appears in the Home page. Ensure the Footer item actually has a value in the main item field.

5.9.5: Reading the other links field

Detailed Steps

1. Add the following lines in the code section at the top of the **FooterContent.cshtml**

   ```
   Sitecore.Data.Fields.MultilistField otherlinksfield = Model.Item.Fields
   ["Other links"];

   var otheritems = otherlinksfield.GetItems();
   ```

2. Locate the <ul> section and remove all the <li> tags inside *except one*.
3. Surround the remaining <li> with a foreach loop. Then replace the static text with the contents of the heading field.
4. Finally replace the link with a call to the **LinkManager.GetItemUrl()** method. Your code will look like this

```
<ul class="nav nav-stacked">
    @foreach (var otheritem in otheritems)
    {
        <li class="">
            <a href="@Sitecore.Links.LinkManager.GetItemUrl(otheritem)">
                @Html.Sitecore().Field("heading", otheritem)
            </a>
        </li>
    }
</ul>
```

5. **Save** changes and check the result on the Home page.

Whenever you are rendering fields that are not editable inline, you have to do a bit more extra work. You will typically do as you did here: retrieve the field's contents and then use them in your code to generate the HTML.