Sitecore Bootcamp

Sitecore 8.X

TABLE OF CONTENTS

[Getting Started 3](#_Toc484783965)

[Introduction to Sitecore 3](#_Toc484783966)

[Food for thought 6](#_Toc484783967)

[Installing Sitecore 6](#_Toc484783968)

[Setting up your development Environment 8](#_Toc484783969)

[Building Your First Module 11](#_Toc484783970)

[Experience Editor 15](#_Toc484783971)

[Update your module to allow editing using the Experience Editor (Visual Studio) 15](#_Toc484783972)

[Editing content using the Experience Editor (Sitecore) 16](#_Toc484783973)

[Adding modules to a page using the Experience Editor (Sitecore) 17](#_Toc484783974)

[Food for thought 17](#_Toc484783975)

[Troubleshooting 18](#_Toc484783976)

[Problem: Cannot build solution. Invalid reference error. 18](#_Toc484783977)

[Problem: Error loading assembly when viewing site. 18](#_Toc484783978)

[Problem: xDB or analytics error 18](#_Toc484783979)

[Problem: Permission Errors 19](#_Toc484783980)

[Problem: Http error in Sitecore admin area 19](#_Toc484783981)

[Problem: Error in Experience Editor "Failed item resolve … does not contain a property that represents the item ID" 19](#_Toc484783982)

[Want To Learn More? 20](#_Toc484783983)

# Getting Started

Welcome to Sitecore Bootcamp. This bootcamp will get up and running with Sitecore 8.x. Part 1 is a brief introduction to Sitecore including some of the main features and terminology. We use the Habitat demo site provided by Sitecore. Part 2 walks you through installing Sitecore and setting up IIS to view a site on your local machine. In part 3, we setup your local development environment. In part 4 we build our first Sitecore module. Lastly, in part 5, we extend our module to use the experience editor allowing inline content editing.

This series focuses on Sitecore MVC (which is not exactly the same as .NET MVC). We'll assume you have some basic knowledge of C# and .NET MVC to get started. After you have gone through this series, you will be able to navigate within Sitecore's admin area, create modules in Sitecore, and leverage the Experience Editor. Keep in mind this series does not include any architectural concepts. Sitecore has published its "Helix" guidelines. Perficient has it's own Sitecore framework "Ignition" that aligns closely with the Helix guidelines. Ignition saves Sitecore developers time by automatically doing some of the manual work we are doing in this series. But it is helpful to know the manual way before you jump into Helix or Ignition.

# Introduction to Sitecore

1. What is Sitecore
   1. Content management system
   2. Content personalization
   3. Marketing
      1. A/B testing
      2. Goal tracking
      3. User journey paths
2. Pages
   1. A page has a layout
      1. Main design wrapper
         1. Header
         2. Navigation
         3. Content regions
         4. Footer
   2. A layout has placeholders
      1. Named regions where content can be placed
      2. Can be nested inside other placeholders
      3. Can restrict what kind of content is allowed
   3. Renderings go in placeholders
      1. Call an MVC Controller/Action Method
      2. Link to a datasource template
         1. An instance of an item template with actual content
         2. Can be reused as a backing datasource for multiple renderings
      3. Are displayed via a controller rendering
      4. Have an MVC Model interface that matches the item template
3. Login -> Dashboard
   1. <http://sitecorebootcamp.localhost/sitecore>
      1. Default username:password = admin:b
4. Analytics
   1. Campaigns and Goals
   2. User paths through site
5. Media Library
   1. Upload images and files
6. Control Panel
   1. Background system settings
7. User Management
8. Desktop -> Sitecore Start Menu

Using the Desktop is a good habit to get into when you first start with Sitecore.

* 1. Access all the tools from the dashboard
  2. Access developer tools not available on the dashboard
  3. Open multiple tools at the same time in tabs
  4. Switch between databases

1. Experience Editor
   1. Not always supported by the template or renderings
   2. Edit Mode
   3. Preview Mode
   4. Inline editing
   5. Content Frame
   6. Save Changes
   7. Publish
2. Content Editor
   1. Content Tree
      1. "Pages"
         1. /sitecore/content
         2. Url determined by path in the content tree
      2. Layouts
         1. /sitecore/Layout/Layouts
      3. Placeholders
         1. /sitecore/Layout/Placeholder Settings
      4. Renderings
         1. /sitecore/Layout/Renderings
      5. Media Library
         1. /sitecore/Media Library
         2. Upload images and files
      6. Templates
         1. /sitecore/Templates
   2. Form editing
   3. Save Changes
   4. Publish Item
3. Items
   1. Everything in Sitecore is an item
      1. An abstract object that can be used to model any type of data or entity
   2. Some items (ie. content pages) have presentation and can be displayed in a browser by going to their url
      1. Some items like structured content (ie. blog posts, products) have presentation and can be displayed on their own as a page in a browser by going to their url
      2. Some content items are used behind the scenes and not displayed (ie. settings, metadata)
   3. Content items
   4. Non content items
      1. Other items (ie. renderings, templates) exist outside of the content tree and do not have their own presentation and can not be directly displayed in a page.
4. 3 Databases
   1. Master
      1. Version History
      2. Changes saved not published to web
         1. Not visible to site visitors
   2. Web
      1. Published content
      2. Can be scaled across multiple content delivery servers
      3. You can switch to the web database and browse the content editor if you notice expected content is missing from the site
   3. Core
      1. Sitecore settings
5. Publish Changes
   1. Moves content from Master to Web
      1. Publishes the most recent version for every language
   2. 3 types
      1. Item
         1. Publish the current item
      2. Smart
         1. Publish the differences between master and web
      3. Full (AKA Republish)
         1. Publish the most recent versions from the master database
         2. Very time consuming, especially on a large site
6. Versioning
   1. Not automatic by default
   2. You have to click new version button
7. Workflow
   1. Sitecore has built-in, definable workflow capabilities
   2. You can set who can publish, content approval flow, delayed publishing, automatic publishing, automatic versioning, etc
8. Languages
   1. Sitecore supports multiple languages
   2. You can define which languages are available for your site
   3. You can edit content and create versions for each language

### Food for thought

1. Sitecore is like a box of building blocks. You can build whatever you can imagine.
2. It does not come with pieces pre-built for you. You have to build everything.
   1. Maintain a library of components
   2. Use a Sitecore framework
      1. Ignition
3. You can develop for Sitecore using Webforms or MVC.
   1. Sitecore MVC is NOT the same as .NET MVC.
      1. The idea is the similar, but Sitecore becomes a controller that actually manages calling your controllers.
      2. Not all .NET MVC concepts work in Sitecore MVC.

# Installing Sitecore

1. Prerequisites
   1. SQL Server
      1. 2012 was used to create this guide
   2. Visual Studio
      1. 2015 was used to create this guide
   3. IIS
      1. Version 8.5.9600 was used to create this guide
2. Setup SQL Server to allow Mixed Authentication Mode
   1. Open SQL Server Management Studio
   2. Connect to local server
   3. Right click Server in Object Explorer - > Properties
   4. Click Security page
   5. Server Authentication
      1. Click SQL Server and Windows Authentication
3. Setup a SQL Server user for Sitecore
   1. Open SQL Server Management Studio
   2. Connect to local server
   3. Right click Security in Object Explorer -> Logins -> New Login
   4. General Page
      1. SQL Server Authentication
      2. Set the Login name and password
         1. You can reuse the same login on your local machine for multiple Sitecore instances. But it would be recommended to have a different login for each site in a production environment.
      3. Uncheck enforce password policy
         1. This may not be recommended in a production environment
   5. Server Roles Page
      1. Check boxes for public and sysadmin
         1. This may not be recommended in a production environment
4. Install Sitecore
   1. Download Sitecore
      1. https://dev.sitecore.net/Downloads.aspx
      2. You must have a Sitecore Developer account
   2. Download the license file
      1. <https://hub.perficient.com/communities/Sitecore/SitePages/Home.aspx>
      2. Documents and Assets/Sitecore Installers and Licenses
   3. Run Sitecore Installer
      1. Cannot be installed from a network drive
      2. Installation Wizard
         1. New instance
         2. Accept Terms
         3. Installation Type
            1. Complete
         4. Instance Name
            1. SitecoreBootcamp
         5. License File
         6. SQL Server
            1. This is the name of database server (not the Sql Server instance name)

Shown when you run Sql Server Management studio and connect to your local server.

* + - * 1. Sitecore credentials created in Step 2.
      1. Install path
         1. <Your wwwroot path>\SitecoreBootcamp
      2. Website Name
         1. SitecoreBootcamp

We will rename this SitecoreBootcamp.localhost in the next step. The installer will not allow you to enter special characters in the name.

* 1. IIS Settings
     1. Open IIS Manager
     2. Right click Sites/SitecoreBootcamp in Connections -> Edit Bindings
     3. Click on the entry for "SitecoreBootcamp" -> Edit
     4. Set the Host name to "SitecoreBootcamp.localhost"
        1. You should always a domain extension
           1. Cookies don't work correctly without a TLD
        2. You should use .local or .localhost
           1. Check with your project leader to see if they have a preference
        3. You should NOT use.dev
           1. .dev was purchased by Google as a TLD and tries to route to an actual non-existant domain
     5. Restart the website
  2. Hosts File
     1. Open notepad.exe as an admin
        1. Right click -> Run as admin
     2. Open C:\Windows\System32\drivers\etc\hosts
     3. Change the line   
        "127.0.0.1 SitecoreBootcamp"   
        to   
        "127.0.0.1 SitecoreBootcamp.localhost"
     4. Save
  3. Launch Site
     1. <http://sitecorebootcamp.localhost>
        1. The first page load may take some time as Sitecore rebuilds it caches.

# Setting up your development Environment

1. Open Visual Studio in Admin Mode

If you prefer not to run as admin, you will need to do file system publish instead of web publish in step 11.

* 1. Right click on shortcut icon -> Run as administrator
  2. Right click on shortcut icon -> Open File Location -> Right Click on shortcut icon -> Properties -> Shortcut Tab -> Advanced -> Check "Run as administrator" -> Apply

1. Create a new project
   1. Templates -> Visual C# -> Web -> ASP.NET Web Application
   2. Name
      1. "SitecoreBootcamp"
   3. Location
      1. Point to your projects directory
   4. Check box to create directory for solution
   5. Empty Project Template
   6. Add folders and core references for MVC
   7. Uncheck host in azure
2. Add Sitecore NuGet package source
   1. Open the Options dialog
      1. Tools -> NuGet Package Manager -> Package Manager Settings
      2. Tools -> Options
      3. Quick Launch -> NuGet -> Package Manager Settings
   2. Inside the Options Dialog
      1. NuGet Package Manager -> Package Sources
   3. Click the plus icon
      1. Name: Sitecore NuGet
      2. Source: <https://sitecore.myget.org/F/sc-packages/api/v3/index.json>
      3. Click Update
3. Install Sitecore NuGet packages

While NuGet is the preferred method, you can manually copy the Sitecore DLLs to a \_lib folder and reference in each project that needs to build code against Sitecore.

* 1. Open the Package manager window
     1. Project -> Manage NuGet Packages
     2. Quick Launch -> NuGet => Manage NuGet Packages for Solution
  2. Select Sitecore NuGet from the package source dropdown
  3. Install options (for each of the install packages named below)
     1. Select the version that matches your installed Sitecore version
     2. Dependency behavior = Ignore Dependencies
     3. File conflict actions = Prompt
  4. Install packages
     1. Sitecore.Client
     2. Sitecore.ContentSearch
     3. Sitecore.ContentSearch.Linq
     4. Sitecore.Kernel
     5. Sitecore.MVC
  5. Note
     1. Installing the Sitecore package takes a long time to collect all of the dependencies and installs many packages not needed to do Sitecore development.

1. Install Glass.Mapper NuGet package

Glass is the most-recommended ORM (object relational mapper) for Sitecore. There are other ORMs available. Using the regular Sitecore API for data mapping is also perfectly feasible, but a bit cumbersome.

* 1. Open the Package manager window
     1. Project -> Manage NuGet Packages
     2. Quick Launch -> NuGet => Manage NuGet Packages for Solution
  2. Select nuget.org from the package source dropdown
  3. Install "Glass.Mapper.Sc"
     1. Select the latest stable version
        1. Older version of Sitecore may not be compatible with the latest version of GlassMapper
     2. Dependency behavior = Lowest
     3. File conflict actions = Prompt
  4. Allow Glass.Mapper to upgrade Sitecore packages
  5. Allow dependencies to Install
     1. Castle.Core
     2. Glass.Mapper.Sc.Core
  6. Show all files in solution explorer
  7. Include GlassMapper files in your solution (if not done automatically)
     1. App\_Config/Include/Glass.Mapper.Sc.\*
     2. App\_Config/Include/z.Glass.Mapper.Sc.\*
     3. App\_Start/GlassMapperSc\*

1. Downgrade Sitecore packages
   1. If Glass.Mapper upgraded sitecore packages, downgrade Sitecore.Kernel back to the version that matches your installed Sitecore version and let it auto downgrade any other packages.
2. Edit App\_Start/RouteConfig.cs
   1. Remove the default route "{controller}/{action}/{id}"
      1. This route is not needed by Sitecore
3. Update Views/web.config
   1. Add the following lines to system.web.webPages.razor/pages/namespaces at the top of the list.
      1. <add namespace="Sitecore.Mvc" />
      2. <add namespace="Sitecore.Mvc.Presentation" />
   2. This allows you to use these namespaces in your views without having to include them in every cshtml file.
4. Copy Sitecore web.config to your project directory
   1. Copy ~/wwwroot/SitecoreBootcamp/Website/Web.config to ~/SitecoreBootcamp/SitecoreBootcamp
      1. Overwrite the default web.config in your project directory
      2. The default web.config does not have the sitecore specific configurations. When you publish the solution, it will overwrite the web.config in the wwwroot directory.
5. Build Solution
6. Create Publish Settings
   1. Build -> Publish
   2. Profile = Custom
      1. Profile Name = <firstinitial><lastname> (ie JDoe)
   3. Connection

If you are not running as admin, you will have to do a file system publish.

* + 1. Publish Method = Web Deploy
    2. Name = Localhost
    3. Server = localhost
    4. Site name = SitecoreBootcamp (The site name in IIS. Not necessarily the same as the url used to access the site.)
    5. Validate Connection -> Green check mark
  1. Settings
     1. Configuration = Debug
        1. You can also create your own configuration if you need to have settings specific to your local machine.
     2. File publish options
        1. Uncheck all three 3 boxes
  2. Publish

1. Validate publish settings
   1. Check existence of ~/wwwroot/SitecoreBootcamp/Website/bin/SitecoreBootcamp.dll
2. Launch Site
   1. <http://sitecorebootcamp.localhost>
      1. The first page load after any publish may take some time as Sitecore rebuilds it caches.
3. Log into Sitecore
   1. <http://sitecorebootcamp.localhost/sitecore>
      1. Default username:password is admin:b

# Building Your First Module

1. Log into Sitecore
   1. <http://sitecorebootcamp.localhost/sitecore>
      1. Default username:password is admin:b

In real projects, you will organize items in sub-folders. The structure will be dictated by the solution architect, client, or framework.

1. Click Desktop -> Sitecore Start Menu –> Content Editor
2. Create "User Defined" folders to store your content (Sitecore)
   1. Right click on /sitecore/Layout/Layouts -> Insert -> Layout Folder
   2. Right click on /sitecore/Layout/Renderings -> Insert -> Rendering Folder
3. Create a Layout (Sitecore)
   1. Right click on /sitecore/Layout/Layouts/User Defined -> Insert -> Layout
   2. Create layout "Main Layout"
   3. Location = Layouts/User Defined
   4. File Location = /Website/Views
   5. Change the Path field to /Views/MainLayout.cshtml
   6. Click Save
4. Create a Page Template (Sitecore)
   1. Right click on /sitecore/Templates/User Defined -> Insert -> New Template
   2. Name = "Page"
   3. Base Template = "Templates/System/Templates/Standard template" (default)
   4. Location = Templates/User Defined
   5. On the builder tab
      1. Section = Meta
      2. Fields
         1. Title = Single-Line Text (default)
         2. Keywords = Single-Line Text (default)
         3. Description = Multi-Line Text
   6. Click Save
5. Set the Page Template Icon (Sitecore)

Icons help content editors easily see content type while looking at the content tree. Choosing a custom icon is not required.

* 1. Click /sitecore/Templates/User Defined/Page
  2. Click Configure in the Sitecore navigation bar -> Icon -> More Icons
  3. Click the Icons tab
     1. Icons are sorted by categories in a drop down at the top of the dialog
  4. Find an icon that represents your template
     1. The icon will appear next to each instance of this template
     2. I used Applications/Document\_Plain
  5. Click Ok on the Icon dialog
  6. Click Save

1. Set the Page Template Standard Values (Sitecore)
   1. Click /sitecore/Templates/User Defined/Page
   2. Click Options in the Sitecore navigation bar -> Standard Values
      1. Standard values apply to all instances of this template, but can be overwritten by an individual instance as needed.
      2. They are a good place to provide default values for fields, default layouts, modules (ie navigation, headers, footers)
   3. Click /sitecore/Templates/User Defined/Page/\_\_Standard Values
   4. Click Presentation in the Sitecore navigation bar - >Details
   5. Click Edit on the Shared Layout tab Next to Default
   6. On the Layout tab, select Layouts/User Defined/Main Layout
      1. This will make sure all new pages that use this "Page" template all use the "MainLayout" by default.
   7. Click Ok on the Device Editor dialog
   8. Click Ok on the Layout Details dialog
   9. Click Save
2. Create an Item Template (Sitecore)
   1. Right click on /sitecore/Templates/User Defined -> Insert -> New Template
   2. Name = "Content Blurb"
   3. Base Template = "Templates/System/Templates/Standard template" (default)
   4. Location = Templates/User Defined
   5. On the builder tab
      1. Section = Content
      2. Fields
         1. Heading = Single-Line Text (default)
         2. Subtitle = Single-Line Text (default)
         3. Content = Rich Text
   6. Click Save
3. Set the Item Template Icon (Sitecore)
   1. Click /sitecore/Templates/User Defined/Content Blurb
   2. Click Configure in the Sitecore navigation bar -> Icon -> More Icons
   3. Click the Icons tab
      1. Icons are sorted by categories in a drop down at the top of the dialog
   4. Find an icon that represents your template
      1. The icon will appear next to each instance of this template
      2. I used Applications/text\_align\_justified
   5. Click Ok on the Icon dialog
   6. Click Save
4. Create a Rendering (Sitecore)
   1. Right click on /sitecore/Layout/Renderings/User Defined -> Insert -> Controller Rendering
      1. View renderings use Sitecore as the "Controller" and call a cshtml directly. This can be useful for simple modules.
      2. It is a good practice to use controller renderings instead of view renderings. It will let your code be more flexible in future with minimal impact to your code and templates.
   2. Create rendering "Content Blurb"
   3. Fields
      1. Controller = ContentBlurb
      2. Controller Action = ContentBlurb
      3. Datasource Template = Insert Link to /sitecore/Templates/User Defined/Content Blurb
         1. This creates the link between the template in Sitecore and your MVC Controller.
   4. Click Save
5. Create an Error View (Visual Studio)
   1. Right click on Views folder -> Add -> MVC 5 View Page (Razor)
   2. Name = Error
   3. Add html/razor (see attached)
6. Create a Page Model (Visual Studio)
   1. Right click on Models folder -> Add -> New Item
   2. Choose Visual C#/Code/Interface
   3. Name = IPage.cs
   4. Add the SitecoreType attribute decorator to the interface
      1. [SitecoreType(TemplateId = "{GUID}", AutoMap = true)]
         1. This helps sitecore automatically populate properties
      2. Update the GUID from the Page Template
         1. Click /sitecore/Templates/User Defined/Page (sitecore)
         2. Click on the Content tab
         3. Click Quick Info to expand
         4. Click on the Item ID and copy to the clipboard
   5. Add string properties for the fields in the Sitecore template
   6. Remove any unnecessary using statements
7. Create a Layout View (Visual Studio)
   1. Right click on Views folder -> Add -> MVC 5 View Page (Razor)
   2. Name = MainLayout
   3. Add html/razor (see attached)
8. Create an Item Template Model (Visual Studio)

This guide uses interfaces for models. You may also see concrete classes used for models.

* 1. Right click on Models folder -> Add -> New Item
  2. Choose Visual C#/Code/Interface
  3. Name = IContentBlurb.cs
  4. Add the SitecoreType attribute decorator to the interface
     1. [SitecoreType(TemplateId = "{GUID}", AutoMap = true)]
     2. Update the GUID from the Item Template
        1. Click /sitecore/Templates/User Defined/Content Blurb (Sitecore)
        2. Click on the Content tab
        3. Click Quick Info to expand
        4. Click on the Item ID and copy to the clipboard
  5. Add string properties for the fields in the Sitecore template
  6. Remove any unnecessary using statements

1. Create an Item View (Visual Studio)
   1. Right click on Views folder -> Add -> New Folder
   2. Name = ContentBlurb
   3. Right click on Views/ContentBlurb folder -> Add -> MVC 5 View Page (Razor)
   4. Name = ContentBlurb.cshtml
   5. Add html/razor (see attached)
2. Create an Item Controller (Visual Studio)
   1. Right click on Controllers folder -> Add -> Controller
   2. Empty MVC 5 Controller
   3. Name = ContentBlurbController
   4. Extend GlassController instead of Controller
   5. Add ContentBlurb action method (see attached)
3. Build and deploy the solution (Visual Studio)
   1. Build -> Build SitecoreBootcamp (or Build Solution)
   2. Build -> Publish SitecoreBootcamp
   3. Refresh home page in browser
      1. This allows Sitecore to rebuild its caches
4. Create a new Page (Sitecore)
   1. Right click on /sitecore/Content/Home -> Insert -> Insert from template
   2. Select Templates/User Defined/Page
   3. Item Name = page1 (no spaces)
   4. Set title, keywords, description
   5. Click Save
5. Create a new Content Blurb (Sitecore)
   1. Right click on /sitecore/Content/Home/page1 -> Insert -> Insert from template
   2. Select Templates/User Defined/Content Blurb
   3. Item Name = CB1
   4. Set heading, subtitle, content
   5. Click Save
6. Add Content Blurb to a Page (Sitecore)
   1. Click on /sitecore/Content/Home/Page1
   2. Click Presentation -> Details in the Sitecore navigation bar
   3. Click Final Layout -> Default -> Edit in the Layout Details dialog
   4. On the Controls tab -> Click Add
   5. Select Renderings/User Defined/Content Blurb
   6. Placeholder = content
      1. This is defined in the MainLayout.cshtml
   7. Click Select to close the rendering dialog
   8. Click the Content Blurb rendering and click Edit

Datasources allow you to reuse the same content in multiple places on the site.

* 1. On the Control Properties dialog -> Data Source -> Browse and choose /sitecore/Content/Home/Page1/CB1
     1. Notice you can only select items of type "Content Blurb" because Sitecore knows the item type for this control.
  2. Click Ok on each dialog to close
  3. Click Save

1. Publish Site (Sitecore)
   1. Click Publish in the Sitecore navigation bar -> Publish dropdown -> Publish Site
   2. Select Republish
      1. This will publish EVERYTHING
         1. A good idea since this is the first publish
      2. It ensures that all of your templates, layouts, renderings, and items get published.
         1. Subsequent edits to items can be published using Item or Smart Publish
   3. Click select all for languages (if you have multiple languages)
   4. Click Publish
   5. Confirm republish
2. Load Page (Browser)
   1. <http://sitecorebootcamp.localhost/page1>
      1. The url is case insensitive
   2. Content blurb item appears on the page
   3. View source to verify keywords and description are present
3. Load Item (Browser)
   1. <http://sitecorebootcamp.localhost/page1/cb1>
      1. Notice that you get the Sitecore error page
      2. It says it cannot find the requested layout with GUID {000-00-00-00-000}
      3. The CB1 content item does not have a layout defined in its presentation details like the page item does. This means it can go on a page, but isn't a page itself.

# Experience Editor

### Update your module to allow editing using the Experience Editor (Visual Studio)

1. Update the Item Template Model
   1. Add Guid ID property to the interface
      1. This is different from the TemplateId GUID in the SitecoreType decorator.
      2. This ID property lets Sitecore match content to the correct module on the page.
2. Update the Item View
   1. Change @Model.Field to @Html.Glass().Editable(a => a.Field, null, true)
      1. This will use GlassMapper to display an editing box while logged in and using the experience editor. It will render the field as text when not logged in.
3. Build and deploy the solution
   1. Build -> Build SitecoreBootcamp (or Build Solution)
   2. Build -> Publish SitecoreBootcamp
   3. Refresh home page in browser
      1. This allows Sitecore to rebuild its caches

### Editing content using the Experience Editor (Sitecore)

1. Log into Sitecore
   1. <http://sitecorebootcamp.localhost/sitecore>
2. Click Desktop -> Sitecore Start Menu -> Experience Editor
3. Click the down arrow at the right of the Sitecore navigation bar to unhide options
4. On the view tab -> Check Navigation bar
5. In the navigation bar -> Click the arrow to the right of home and choose Page1 -> Click Go.
   1. You can also browser directly to <http://sitecorebootcamp.localhost/page1>
   2. You can also use the sites navigation (if it exists)
   3. You can select Publish -> Experience Editor from the Content Editor for a page
6. On the home tab -> Mode -> Click Edit (if not selected)
7. Moving your mouse around the page highlights modules and editable regions
   1. The entire module
      1. Setup personalization
      2. Delete module
      3. Move module
      4. Edit content using Content Editor
   2. Each individual editable field
      1. Inline editing
8. Edit the title, subtitle and content
9. Be sure to click the save button in the Sitecore navigation bar
10. Click Home -> Mode -> Other -> Preview to view the site as a visitor would see it after publishing
11. Click Home -> Publish
    1. Smart Publish
    2. Check box to publish related items
       1. We are editing the page. The CB1 item is related. If this box is not check, the edits to CB1 would not be published.
    3. Click Publish
12. Open a different browser or a private tab and confirm your changes were published successfully.
    1. The experience editor will likely try to load on all pages after you have loaded it once until you log out or click close on the toolbar.

### Adding modules to a page using the Experience Editor (Sitecore)

1. Update the placeholder settings
   1. Open the content editor
   2. Click /sitecore/Layout/Placeholder Settings/content
   3. Allowed Controls -> Edit
   4. Browse to Layout/Renderings/Content Blurb
   5. Click the right arrow between the columns to add it to the selected list
      1. This allows you to add the content blurb module to a page from within the ExEditor
   6. Click Ok to close the dialog
   7. Save
   8. Click Publish in the Sitecore navigation bar -> Globe Icon -> Okay
      1. This will publish the current item
2. Add module to page
   1. Return to /page1 in ExEditor mode
   2. Click Home -> New Component in the Sitecore navigation bar
      1. The page will display "Add here" buttons where new content can be added
   3. Click Add here button where you want to add the new module
   4. In the dialog, choose the type of content to add
      1. Only modules allowed in the placeholder will appear
      2. Click the box to open the properties dialog
         1. If you miss this step, you can get to properties from module region -> More -> Component Properties
   5. Click Datasource -> Browse
   6. Click Create New Content
      1. Name = CB2
      2. Parent = /sitecore/Content/Home/Page1
   7. Click Ok
   8. Edit content fields
   9. Click Save
   10. Click Home -> Publish
       1. Smart Publish
       2. Check box to publish related items
       3. Click Publish

### Food for thought

1. Modules should be "aware" that they are in the Experience Editor - that is, they should gracefully handle error conditions (ie. missing data or datasources) without throwing errors. Modules could also provide instructions or additional information about their purpose right in the EE.
2. Enabling the experience editor for non-simple types (other than text) is more complex and requires "edit frames".

# Troubleshooting

### Problem: Cannot build solution. Invalid reference error.

Details: Ensure the following references exist

1. Castle.Core
2. Glass.Mapper
3. Glass.Mapper.Sc
4. Glass.Mapper.Sc.Mvc
5. Sitecore.Client
6. Sitecore.ContentSearch
7. Sitecore.ContentSearch.Linq
8. Sitecore.Kernel
9. Sitecore.Mvc
10. System.Web.Mvc

Solution: Uninstall NuGet packages and reinstall.

Solution: Add references to NuGet packages.

### Problem: Error loading assembly when viewing site.

Solution: Ensure that the assembly version defined in the web.config matches the deployed dll file in /bin

### Problem: xDB or analytics error

Solution: Install MongoDB

1. Download MongoDB
   1. <https://www.mongodb.com/download-center?jmp=nav#community>
2. Run MongoDB Installer
   1. Accept Terms
   2. Complete Install
3. Create a Windows service for MongoDB
   1. Follow the directions on <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>
4. Note
   1. The pieces of Sitecore that use MongoDB are enabled by default, but can be disabled.
   2. MongoDB may not be required in production.

### Problem: Permission Errors

* Icons missing from Sitecore dashboard
* Cannot create a layout
* Server Error in '/' Application (usually in dialogs)
* Cannot upload files to the media library

Solution: Create missing directories and set writable permissions

1. Create directories in ~/wwroot (if any do not exist after installing Sitecore)
   1. ~/Website/temp
   2. ~/Website/sitecore/shell/applications/debug
   3. ~/Website/sitecore/shell/controls/debug
   4. ~/Website/sitecore/shell/override/debug
   5. ~/Website/sitecore modules/debug
2. Make directories writable in ~/wwwroot (if any are not writable after installing Sitecore)
   1. ~/Website/temp
   2. ~/Website/sitecore/shell/applications/debug
   3. ~/Website/sitecore/shell/controls/debug
   4. ~/Website/sitecore/shell/override/debug
   5. ~/Website/sitecore modules/debug
   6. ~/Website/app\_data
   7. ~/Website/layouts
   8. ~/Website/upload
   9. Give modify permission to any of these accounts that exist
      1. IUSR
      2. NETWORK SERVICE
      3. IUSER\_<machinename>
      4. IIS\_IUSRS

### Problem: Http error in Sitecore admin area

Details:

* You were logged into Sitecore
* Your session timed out
* You log back in
* Sitecore tries to return you to where you were with an escaped string in the address bar

Solution: Remove everything in the address bar after /sitecore

### Problem: Error in Experience Editor "Failed item resolve … does not contain a property that represents the item ID"

Solution: Add the Guid ID {get, set} property to the model.

# Want To Learn More?



Copyright ©2017

All Rights Reserved.

www.perficient.com