



SDL Digital Experience Accelerator Developer (.Net)

Quick Start Guide

Updated: 06 September 2016

Version 1.2 – Document owner: Richard Hamlyn



Exercises

GUIDE INTRODUCTION	3
INTRODUCTION TO MODEL, VIEW CONTROLLER (MVC) DEVELOPMENT	4
INTRODUCTION TO DXA	9
LAUNCHING YOUR DXA WEB APPLICATION IN MICROSOFT AZURE MARKETPLACE	15
SETTING UP AN AREA AND CREATING A VIEW MODEL AND VIEW	25
MANAGING RESOURCES	34
AVAILABLE COURSE OPTIONS (CLASSROOM)	36

Guide introduction

This short introduction to development with SDL Digital eXperience Accelerator (DXA) will introduce you to the basic concepts and knowledge required to commence development work.

The guide will:

- Introduce MVC development (optional).
- Introduce Digital eXperience Accelerator.
- Guide you to setting up a DXA development environment.
- Teach you how to create Views and Models in DXA.
- Introduce you to Resources in DXA.

The guide is split in to two main parts. Chapters 2-3 provide theoretical background information on MVC and DXA. If you are already comfortable with either of these topics then you should jump to Chapter 4 to start the practical sessions. Chapters 4-6 are practical sessions that will enable you to create your own development environment and commence some introductory development tasks that will provide a clear insight to the simplicity of development with SDL DXA.

Pre-requisites:

- .Net 4.5.
- Visual Studio.
- Azure account (this will be run on a free tier).
- Basic knowledge of Microsoft MVC.Net.

Notes:

DXA is also available in Java but this guide does not cover this.
You will connect to an SDL Web environment that has been pre-implemented with DXA.
You do not need to know any details of the Content Management System (CMS) to develop; however, the steps that have been pre-implemented in the CMS are detailed for your knowledge and background information.

Questions or help

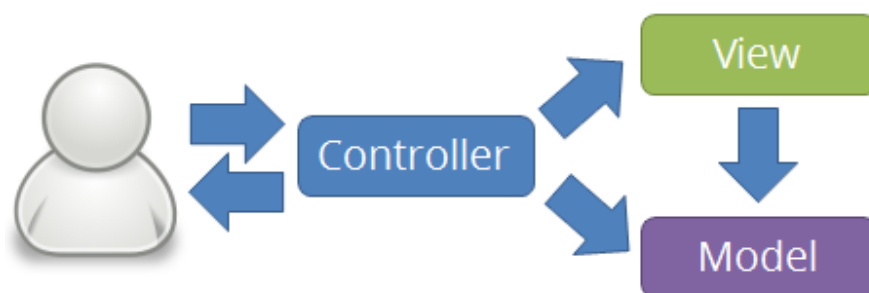
If you have any questions or you require any further help then please post to the SDL Community (<http://community.sdl.com>). Please remember to identify the guide, including the fact that it is the Quick Start Guide (.Net) you are referring to.

Introduction to Model, View Controller (MVC) development

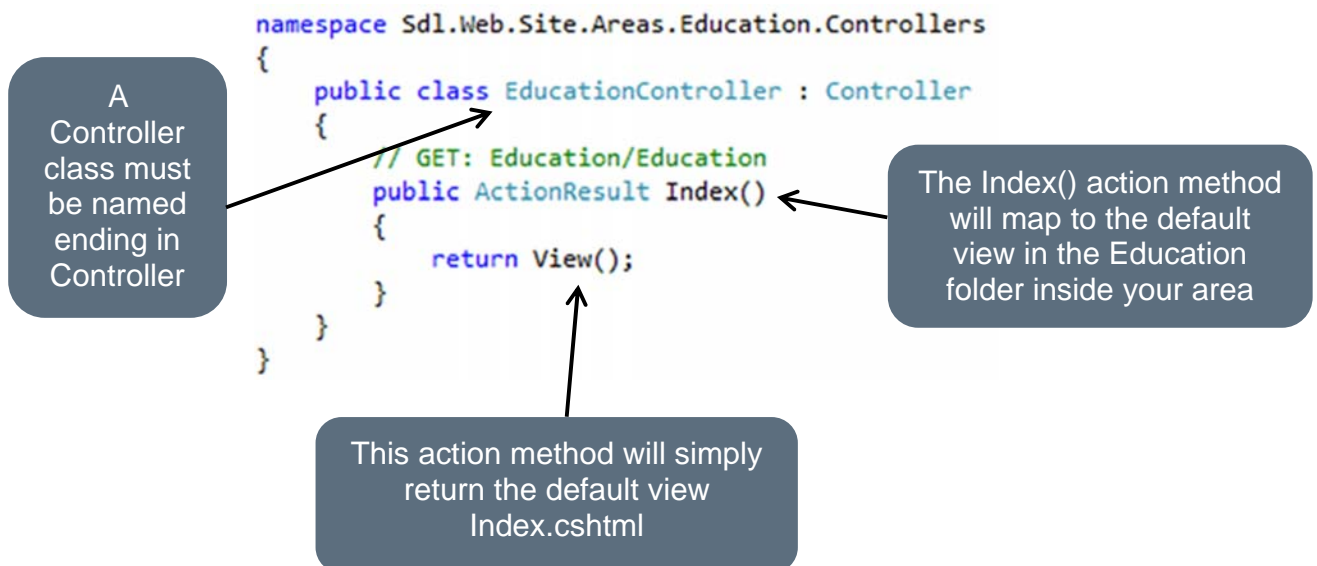
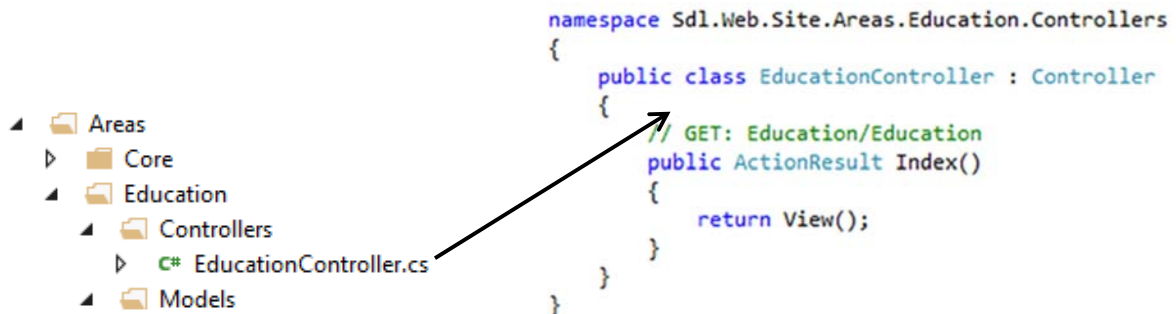


- **ASP.Net MVC Framework**
 - Model, View Controller (MVC) provides three logical layers:
 - Business layer (**Model**)
 - Handles logic for the application data.
 - Display layer (**View**)
 - Handles the presentation of the data, usually returning html to the requester.
 - Input Control (**Controller**)
 - Handles user interaction, such as URL requests from a user, insert data in to the model etc.
 - MVC separation helps team development as you can work on **views**, **models** and **controller** logic in isolation.
 - ASP.Net MVC is a lightweight framework that provides ASP.Net features.
 - MVC is a framework for building web applications using a MVC (Model View Controller) design:

Model, View, Controller (MVC)



- The traditional method of mapping files, such as: *host.com/products/product1.aspx*, does not apply to MVC.
- **Controllers**
 - The MVC framework maps URLs to server code through classes called **Controllers**, which are responsible for:
 - Processing HTTP requests
 - Handling user requests
 - Retrieving and saving data
 - Determining client responses



- You can add new Action methods to the **Controller**:

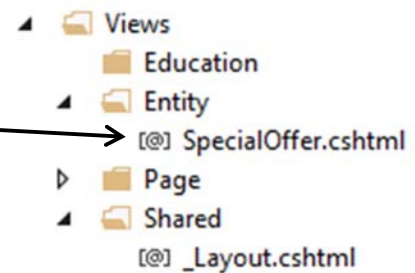
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Sdl.Web.Site.Areas.Education.Controllers
{
    public class EducationController : Controller
    {
        // GET: Education/Education
        public ActionResult Index()
        {
            return View();
        }

        // Get: Education/Education/SpecialOffer.cshtml
        public ActionResult SpecialOffer()
        {
            return View();
        }
    }
}
```

Note!

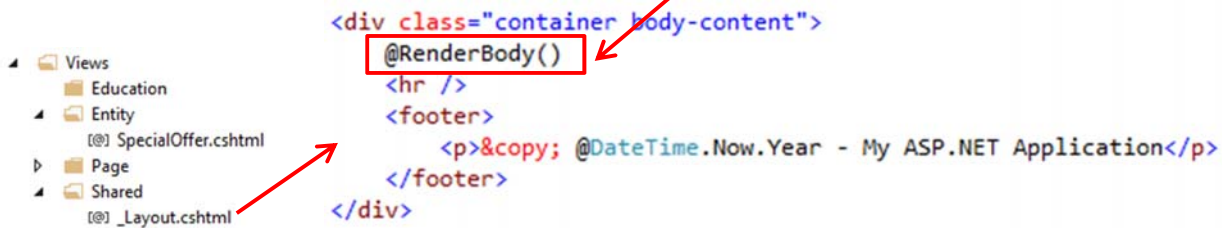
You can also add logic to an Action method



- **View**
 - A View is a template file that customizes HTML content returned to a visitor.
 - In the **Controller**, an “**ActionResult**” returns a View, such as “**SpecialOffer.cshtml**”.
 - The directory where the View is stored is important and must match the one defined in the Controller’s name
 - i.e. “**EducationController**” maps to the directory “**Education**”.
 - Views use Razor syntax.
 - Fixed page elements (headers, footers and navigation) defined in a page layout .../Views/Shared/_Layout.cshtml.

- Example Layout View

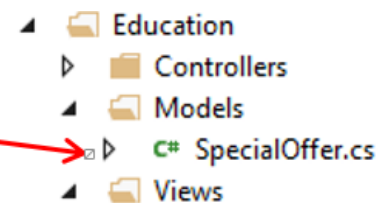
Inside the Layout a view is rendered at the **@RenderBody()** statement



- **Models**

- Models in MVC represent application data.
- A Controller's action method "**ActionResult**"
 - Passes a model object to the view
 - The View Template then can use the model to generate appropriate HTML.
- Example **Model**:

```
public class SpecialOffer : EntityBase
{
    public string Type { get; set; }
    public DateTime ExpiryDate { get; set; }
}
```



A Model in DXA would normally map to a respective Schema

- Example of a **Controller** updating a **Model**:

```
using Sdl.Web.Site.Areas.Education.Models;
namespace Sdl.Web.Site.Areas.Education.Controllers
{
    public class EducationController : Controller
    {
        // Get: Education/Education/SpecialOffer.cshtml
        public ActionResult SpecialOffer(string offerType)
        {
            var typeOfOffer = new SpecialOffer { Type = "Offer type is " + offerType };
            return View(typeOfOffer);
        }
    }
}
```

Include gives access to the Model

Set the special offer 'Type' and return the Special Offer object back to the View

- Example of a **View** accessing a **Model**:

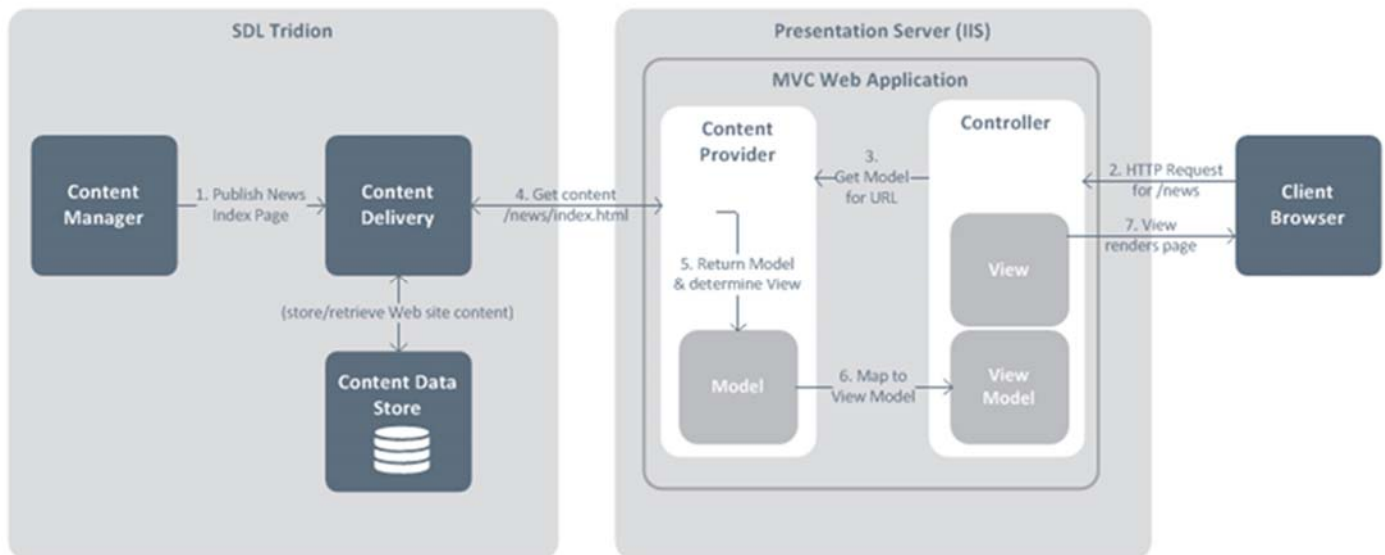
```
@model Sdl.Web.Site.Areas.Education.Models.SpecialOffer
@{
    ViewBag.Title = "Special Offer";
}
<h2>Today's type of Special Offer is: @Model.Type</h2>
```

Include gives access to the Model

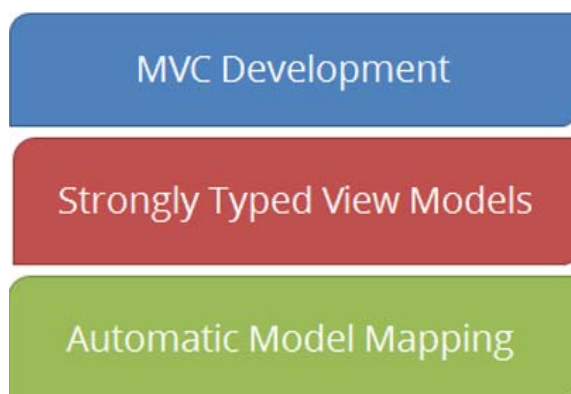
Model object is now available to the View using Razor syntax **@Model.Variable**

Introduction to DXA

- **Digital eXperience Accelerator (DXA)**
 - Provides an optimized and accelerated way to deliver SDL driven sites.
 - Empowers users to design, create and publish sites quickly.
 - Provides a foundation for an organization to build on.
 - Provides rich user experience.
 - Encourages a standard approach to creating and managing Content Managed websites.
- **Architecture**

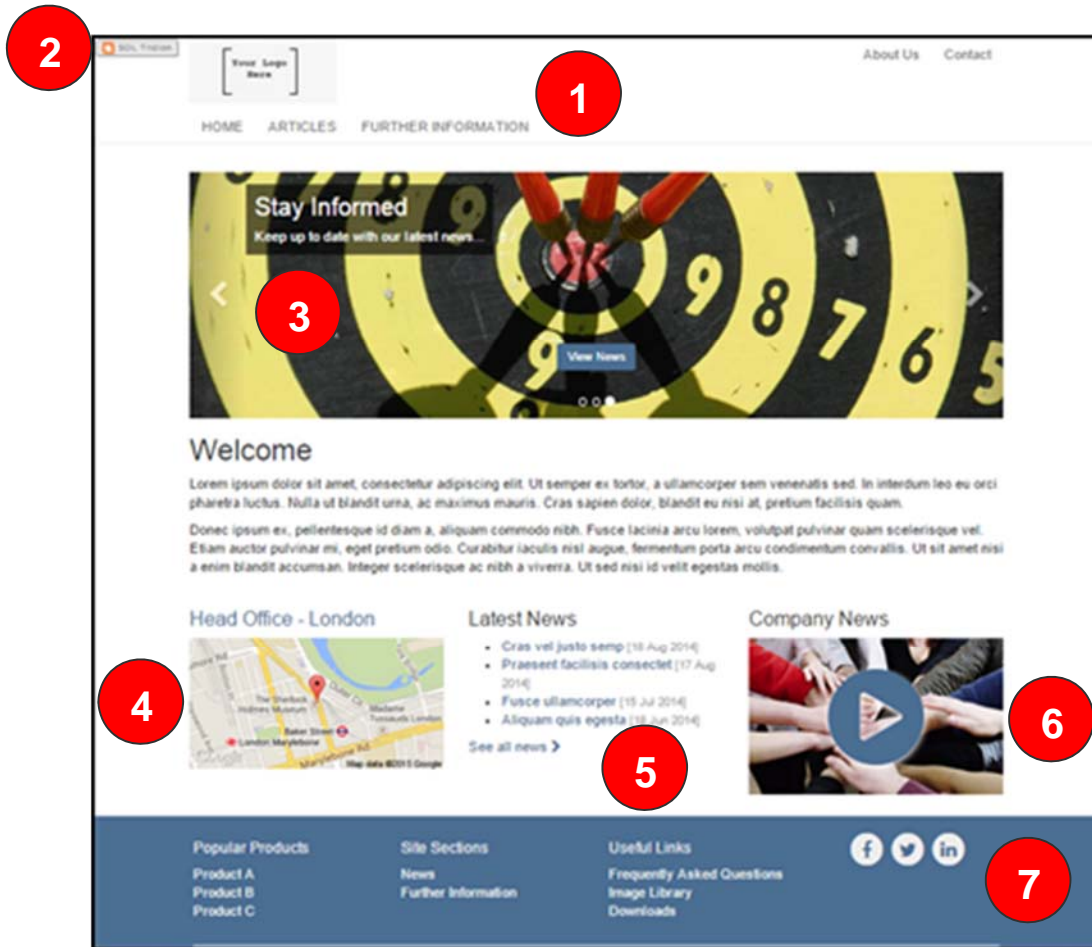


- **Key Functionality**



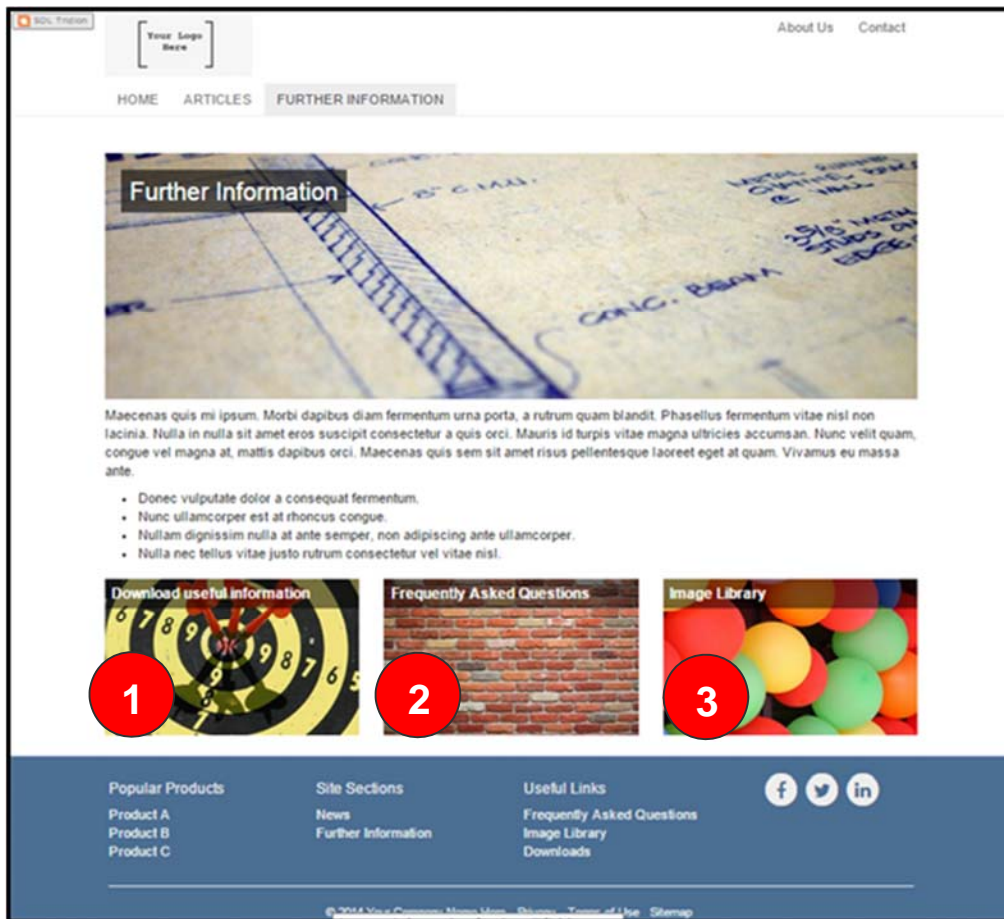
- Web Application.
 - ASP.Net MVC 5.
 - Requires a Content Data Store.
 - XPM implemented by default.
- BluePrint.
 - Default Publication hierarchy (Linear).
- Security Model.
 - Three roles pre-defined.
 - Developer.
 - Editor.
 - Site Manager.
- Implemented in Content Delivery.
- Uses ASP.Net Razor Views.
- Content Manager TBBs are used to:
 - Publish Navigation.
 - Configure the site.
 - Provide resource data assets.
 - Provide HTML design assets.
- Integration.
 - Most common social networks.
 - YouTube.
 - Google Maps.
 - Google Analytics.
- Navigation and sitemap.
 - Top and left navigation.
 - Breadcrumb
 - Sitemap and Google sitemap.

○ Homepage



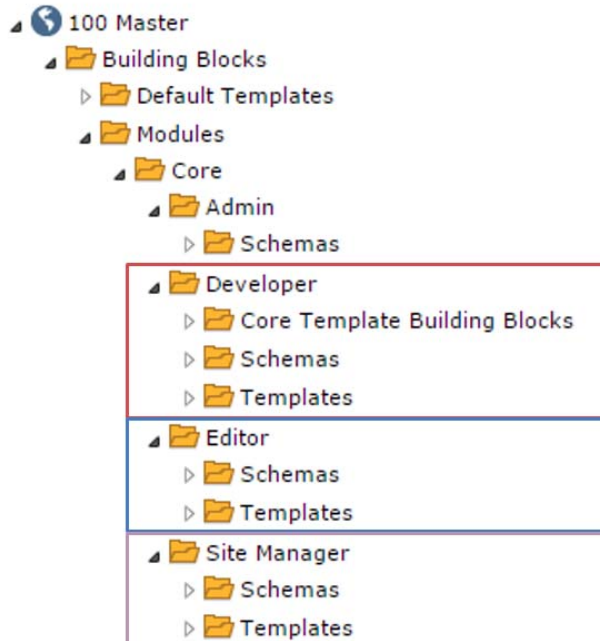
1. Header.
2. Experience Manager (XPM).
3. Rotating Banner.
4. Google Maps integration.
5. Dynamic List.
6. YouTube extension.
7. Footer.

- **Further Information page**



1. Download area.
2. FAQ.
3. Image library.

○ Schemas



Schemas are created in folders linked/restricted to roles



Developer

Edit site layout and functionality

Editor

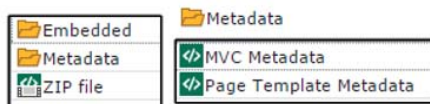
Edit content

Site Manager

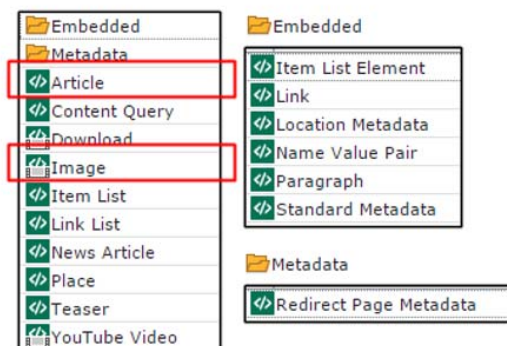
Edit content and configuration



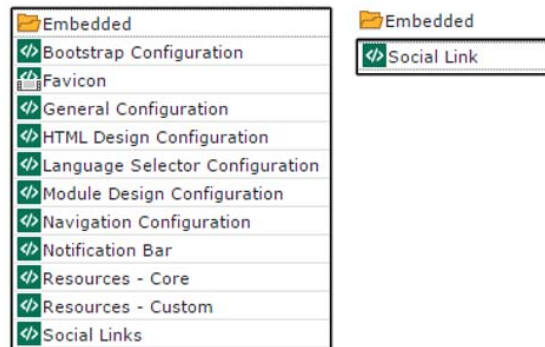
Developer



Editor



Site Manager



Article and Image Schemas have fields mapped to Schema.org schemas

Schema.org

Some DXA Schemas are mapped to Schema.org structure

Schema.org:

Common set of schemas for structuring website data in pages
Enables Search Engines to index and understand your site data
Implemented by all major search engines, including:



Provides search engine benefits, such as a Google results site search

[SDL: The Leader in Global Customer Experience Solutions](#)

www.sdl.com/ ▼

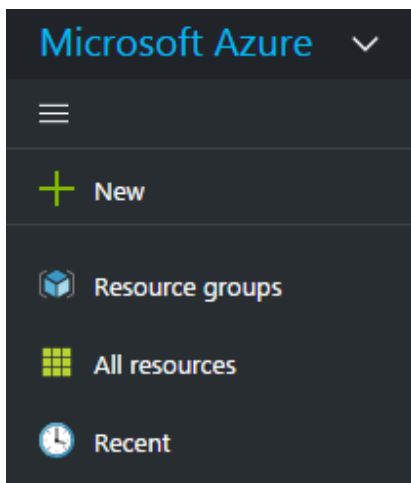
Bring your brand to the world & the world to your brand with integrated software & services for Language Translation & Digital Content Management by SDL.



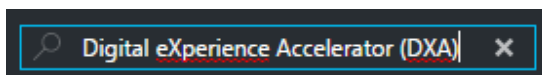
Launching your DXA Web Application in Microsoft Azure Marketplace

In this assignment, you will launch the latest DXA Web Application from the Azure Marketplace and setup your development environment in Visual Studio.


1. Log in to your Microsoft Azure account:
 - a. If you do not have an account you can create a free one at: <https://azure.microsoft.com/en-us/free/>
Note! If you have a Microsoft account, you can use that i.e. myname@hotmail.com or myname@outlook.com etc.
 - b. If you have an account, navigate to: <https://portal.azure.com/>
 - i. You will be requested to login to your Azure account.
 - ii. Once you have signed in you will see the Azure Dashboard:
2. Create your Digital eXperience Accelerator Web Application:
 - a. From your Azure Dashboard, select '**New**'.



- b. Click in the '**Search**' box and type '**Digital eXperience Accelerator**' and hit enter. You will see Azure Marketplace finds the latest DXA web Application:




You will see the Web Application in the Filter panel:


Digital eXperience Accelerator (DXA) ×		
Results		
NAME	PUBLISHER	CATEGORY
 Digital eXperience Accelerator (DXA)	SDL	Web + Mobile


- c. Select the DXA Web Application from the Results window above.
- d. Click on the **'Create'** button that slides in to the dashboard.


Create


- e. Now complete the details of the Web Application:

* App name
 
 .azurewebsites.net

* Subscription
 

* Resource Group 
☒ Create new ☐ Use existing



* App Service plan/Location
 

☒ Pin to dashboard

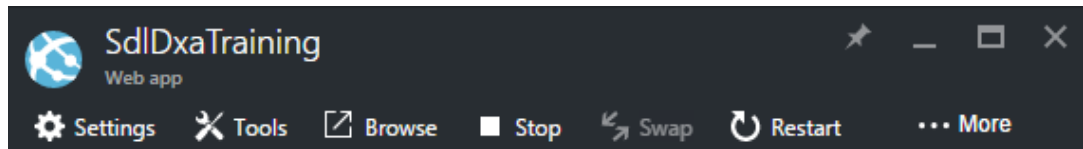
Create



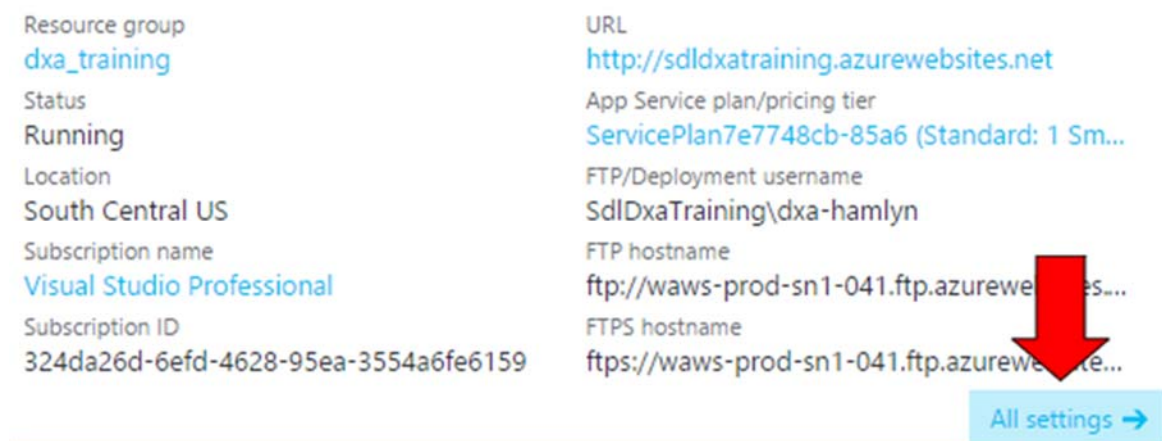
Remember to select **'Pin to dashboard'** so you can find the Web Application easily later. You can enter different details as you wish.

- f. Click on Create.
 - i. You will see your Web Application get built and deployed. This will take a short time to complete.

- ii. Once it has completed deploying you will see the Web Application in your dashboard and then it will automatically load:



- g. You now need to add a GIT Repository so we can update the application and connect from your own Visual Studio.
- h. Click on '**All settings**' from your Web Application:



- i. You can also see the Web Application's URL, which, at this point, connects to the default SDL Web environment. We need to update this so we can connect to the new development CMS.
- i. Now, to the right of '**All settings**' you will see the setting's options. Scroll down to '**Publishing**' and click on '**Deployment credentials**' as shown below:



- j. Complete the credentials' details you want to use:



If you already use Azure, you may already have a deployment account setup, which you can use for this session. If you do not then follow the step below.

* FTP/deployment user name 



* Password 

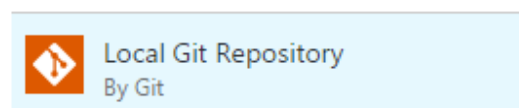
* Confirm password

- i. Enter a unique deployment name and your chosen password; you will need these later so remember what you use!
- k. Click on '**Save**'
- l. Now go back to the settings menu and under '**Publishing**' select '**Deployment Source**':



- m. Click on '**Configure source**' and select '**Local Git Repository**'.



You can use any other source repository if you wish.

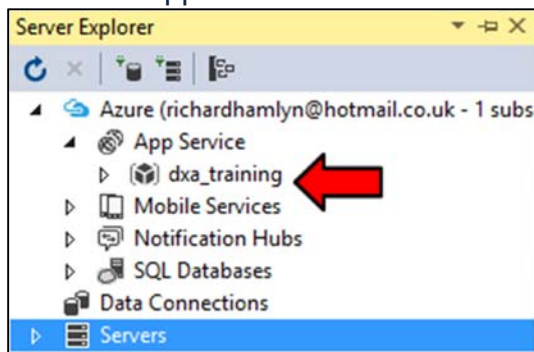
- n. Click '**OK**' and you will see the '**Git clone url**' in your properties window of the project.

Git clone url
https://dxadeveloper@sdltraining.scm.a...

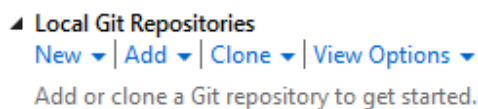
- o. Copy the '**Git clone url**'. You will need this later when we setup Visual Studio.

3. Setup your development environment in Visual Studio.

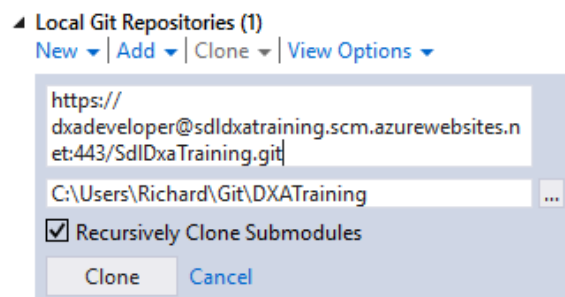
- a. Open Visual Studio (this document demonstrates VS 2015)
- b. Sign in to Visual Studio and open the **Server Explorer** window
- c. Expand the '**Azure**' node and, if you are signed in to Visual Studio, you will see your new Web Application.



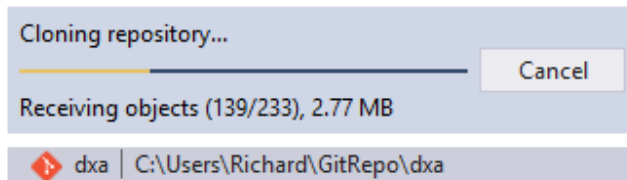
- d. Still in Visual Studio, open the '**Team Explorer**' window and select '**Clone**' from the Local Git Repository option in this window:



- e. Add the location on your local drive where you want to store your Local Git Repository and the Git clone URL from Azure (you should have copied this previously):



- f. Click **'Create'**.
- g. You will now see your local Git Repository has been created for you.

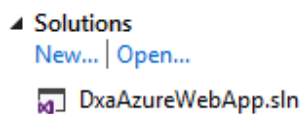


- h. Double click on your cloned repository in Team Explorer.

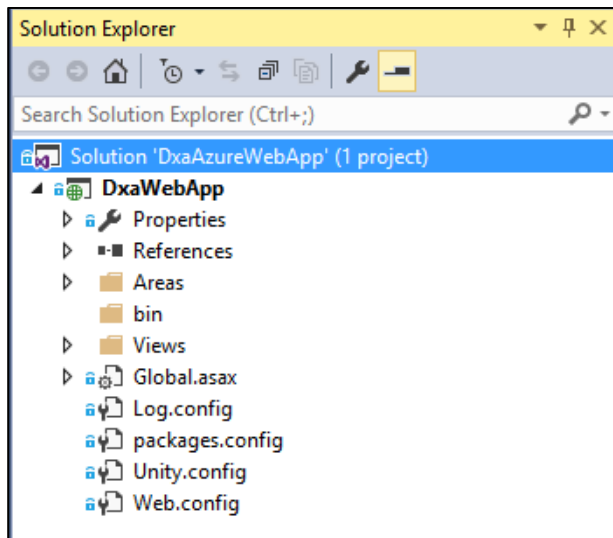


Enter the deployment credentials you created in Azure previously.
It will take a little time to clone the repository!

- i. You will see the following solution option:

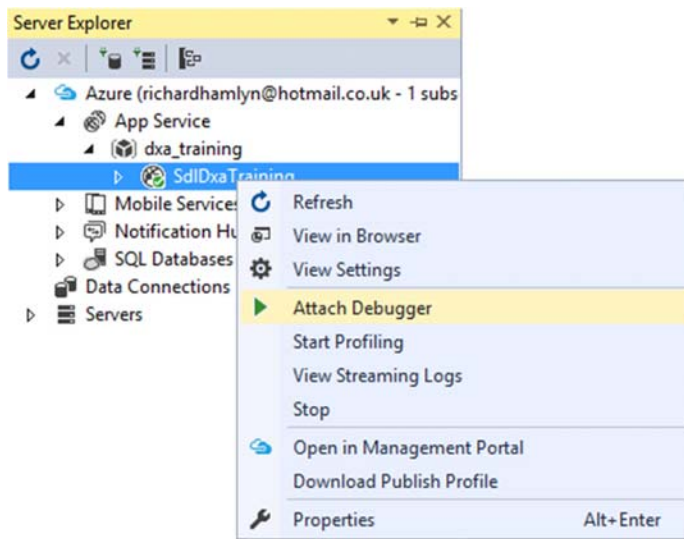


- j. Double click on your DXA solution to load it in to Visual Studio.
- k. Switch back from Team Explorer to Solution Explorer and you will find your Web Application.

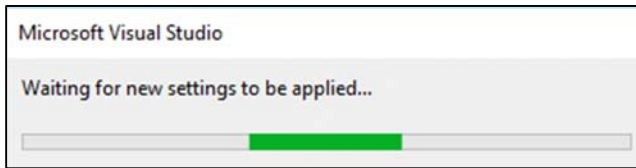


You can now run your project!

4. Setting up Debugging in your new project.
 - a. Open the '**Server Explorer**' window and right click on your project node.
 - b. Select '**Attach Debugger**'

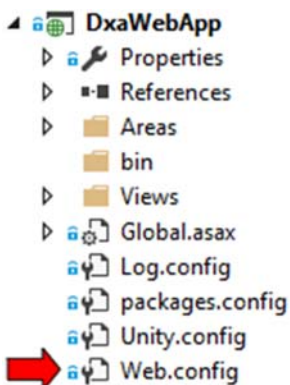


- c. Once the settings have been applied, you will be able to set breakpoints and run your local project.



5. Configuring your application to use the training (or your own) Content Interaction Service.

- a. In this guide, you will use a training server that has DXA pre-implemented.
- b. In Visual Studio, Solution Explorer, open the Web Application's **Web.config**.



- c. On line 8 you will see a reference to an 'appsettings' file that we no longer want to use. Delete the `file="SdlSettings.config"` entry so that line 8 looks like this:

```

8  <appSettings>
9    <add key="webpages:Version" value="3.0.0.0" />
10   <add key="webpages:Enabled" value="false" />

```

- d. Now uncomment line 19-22 and add the following EndPoint for the training Discovery Server that will provide access to a read only Content Interaction Service.

```

16  <!-- After Azure Web Gallery deployment, your DXA Web Application is connected to a read
17       If you want to use your own CIS environment, remove the reference to SdlSettings.co
18  -->
19  <add key="discovery-service-uri" value="http://sdl-training.com:8082/discovery.svc" />
20  <add key="oauth-enabled" value="false" />
21  <add key="oauth-client-id" value="cduser" />
22  <add key="oauth-client-secret" value="CDUserP@ssw0rd" />
23

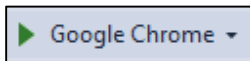
```

Note! The Discovery Service for the training instance resides at **<http://sdl-training.com:8082/discovery.svc>**

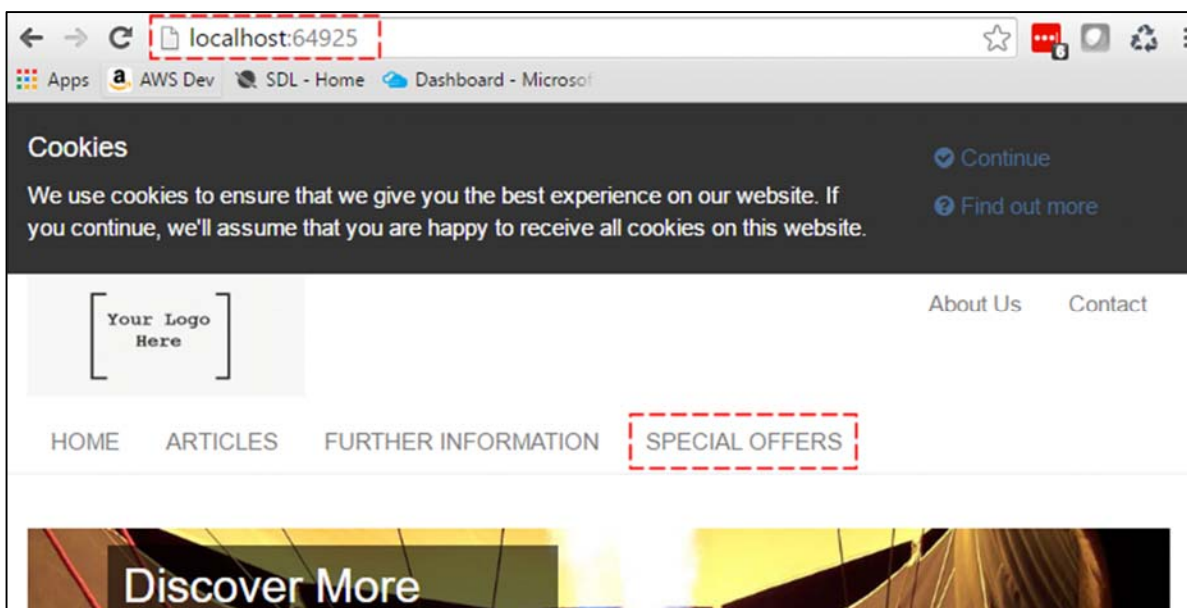
- e. Change oauth-enabled to false, as we do not need the security for this training.

6. Run the application to test connectivity.

- a. From within your Visual Studio project, hit the '**Run**' button selecting your preferred browser. In this guide, we demonstrate using Chrome.



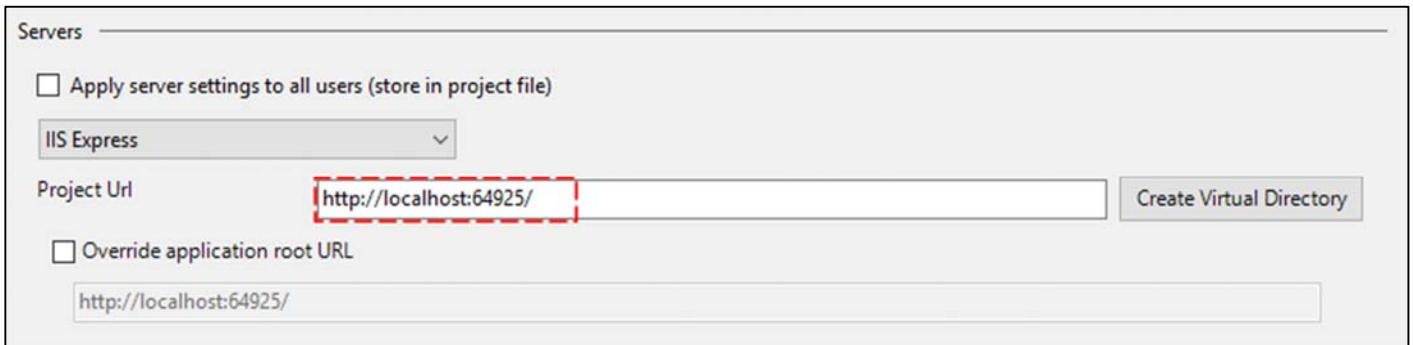
- b. The project will now load using the content from the training server (sdl-training.com).



The default port that Visual Studio uses is 64925. The SDL Web Topology is setup to work with <http://localhost:64925>, so it is important that you use this port.

Continued...

- c. To change the port in Visual Studio
 - i. Right-click your project in Solution Explorer and select 'Properties'.
 - ii. Select 'Web' and ensure the Project URL is set as follows:



Servers

☐ Apply server settings to all users (store in project file)

IIS Express

Project Url http://localhost:64925/ Create Virtual Directory

☐ Override application root URL

http://localhost:64925/



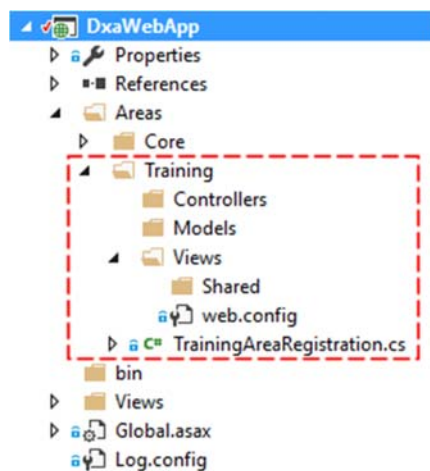
You will see a section of the website called 'Special Offers'. This is not a standard part of DXA. The following sections of this guide will get you to extend the DXA website by implementing the Special Offers section of the site.

Setting up an Area and Creating a View Model and View

In this assignment, you will create custom functionality by creating a new Area to separate your functionality from that of the DXA Core. You will also create a **ViewModel** and a **View** for the Special Offer content that has been created in the CMS.

Detailed steps:

1. Create the required folder for your new area in Visual Studio:
 - a. In Solution Explorer, right-click on the project node and select 'Add' – 'New Area'.
 - b. Name the Area '**Training**' and click 'Add'.



Note!

Areas help separate your code and functionality from that of the Core in DXA.

The above Area structure will be created. This ensures that the extended code you write in a project is separated from that of the 'Core'.

- c. Now, open Areas/Training/web.config (shown above)
- d. Add the following namespaces:

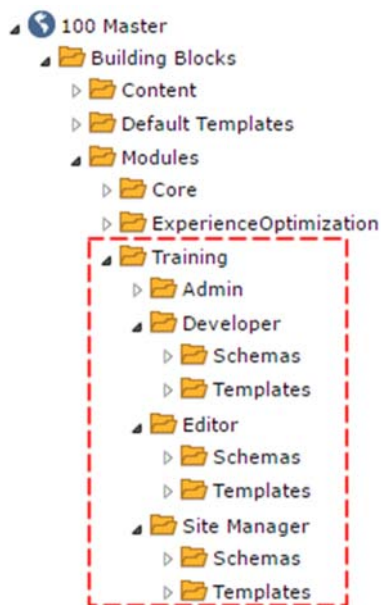
```
<add namespace="Sdl.Web.Mvc.Configuration"/>
<add namespace="Sdl.Web.Mvc.Html"/>
<add namespace="Sdl.Web.Common.Configuration"/>
<add namespace="Sdl.Web.Common.Models"/>
<add namespace="Sdl.Web.Common.Extensions"/>
```

- e. You are now ready to create your first new View Model!

2. Create a View Model and View for the new Special Offers that the Business will manage from the CMS.



The following Module was created in the CMS already and this maps to our Area that we are creating now:



Information only!

You do not need to do anything with the content in the CMS.



Do not worry if you do not understand the CMS elements of this, it is added to the document to explain what has already been setup but is not essential information for a Developer. You can simply assume that the business have already implemented the Content side of the site and you are providing the Delivery side.

The following Schema exists in the CMS and you will map this to a Model in your web application next.

General

General	Design	Metadata Design	Source	Workflow	Info
* Name:	<input type="text" value="Special Offer"/>				
* Description:	<input type="text" value="Special Offer"/>				
* Schema Type:	<input type="text" value="Schema"/>				
* Root Element Name:	<input type="text" value="SpecialOffer"/>				
Translate Name:	<input type="checkbox"/>				
Aggregate Translation Items:	<input type="checkbox"/>				

Design (Fields)

	XML Name	Description	Type
* headline	Headline	Text	
* image	Image	Multimedia Link	
* description	Description	Text	
* link	Link	Component Link	

Information only!

You do not need to do anything with Schemas in the CMS.

Metadata (Fields)

	XML Name	Description	Type
*	expiryDate	Expiry date	Date

This means we need a Model that can bind to CMS Components based on this Schema with the following fields:

headline: *Special Offer title*

image: *Jpeg image*

description: *Information about the Special Offer*

link: *A link to a related article to that of this Special Offer*

expiryDate: *The expiration of the Special Offer*

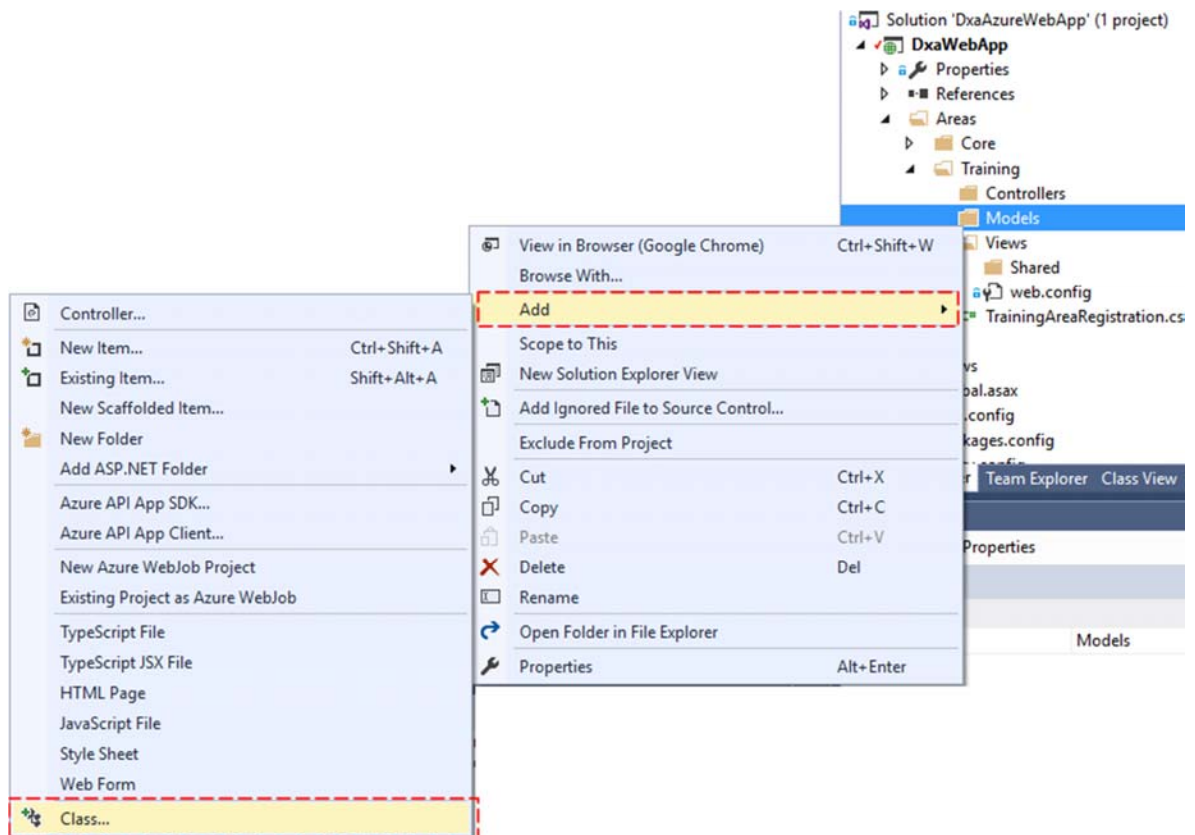
Inside the CMS, the Content Editors for the Special Offers promotion have already created some content that we can use for testing later:

3. Create the View Model that will map to the Special Offer Schema created by the business in the CMS.



A View Model is a simple class to represent the content you want to display. The class has strongly typed properties to enable you to easily refer to them in your View.

- a. Right-click on your '**Models**' folder for your Area in Solution Explorer and select '**Add**' – '**Class**'.



- b. Name the new Class '**SpecialOffer.cs**' and click '**Add**'.
- c. Add the following code to the class to create your Special Offer Model.

Code Box

```
using System;
using Sdl.Web.Common.Models;

namespace Training.Models
{
    [SemanticEntity(Vocab = "http://schema.org", EntityName = "Offer", Prefix = "s", Public
= true)]
    public class SpecialOffer : EntityModel
    {
        [SemanticProperty("s:name")]
        public string Headline { get; set; }
        [SemanticProperty("s:image")]
        public Image Image { get; set; }
        [SemanticProperty("s:description")]
        public string Description { get; set; }
        [SemanticProperty("s:url")]
        public string Link { get; set; }
        [SemanticProperty("s:validThrough")]
        public DateTime ExpiryDate { get; set; }
    }
}
```



It is important that your Entity View Models inherit from the **Sdl.Web.Models.EntityModel** base Entity class, or at least implement the **IEntity** interface as this contains properties that are required by the Web application to store semantic information.

d. Adding Semantic context to the View Model:

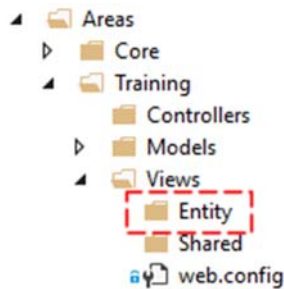
- i. The above View Model uses the **SemanticEntity** and **SemanticProperty** attributes to add Semantics to the Model.



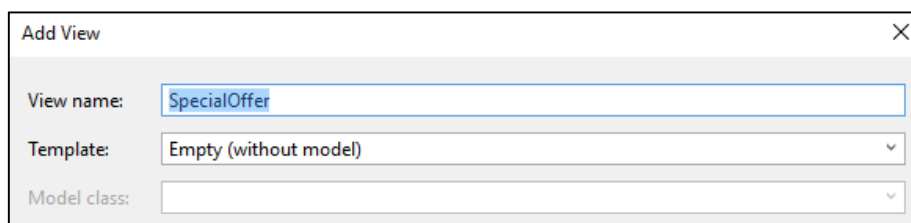
The example above uses the Schema.org vocabulary to define the **SpecialOffer** model to the Schema.org **Offer** and provides corresponding semantic meaning to the properties from the Schema.org definition of an Offer.

4. Create the View:

- Define a View to render the View Model created in the previous steps.
- In Solution Explorer of Visual Studio, create a new folder named 'Entity'.



- Right click the 'Entity' folder and select 'Add' – 'View'.
- Name the View 'SpecialOffer.cshtml' and click 'Add'.



- Add the following code to the View:

Code Box

```

@model Training.Models.SpecialOffer

<div class="content" @Html.DxaEntityMarkup(Model)>
    <br />
    <div @Html.DxaPropertyMarkup(Model, "Image")>
        @Html.Media(Model.Image, "10%", 1)
    </div>
    <h1 @Html.DxaPropertyMarkup(Model, "Headline")>@Model.Headline</h1>
    <p @Html.DxaPropertyMarkup(Model, "Description")>@Model.Description</p>
    <a href="@Model.Link" class="btn btn-primary" @Html.DxaPropertyMarkup(Model,
"Link")>More info</a>
    <br /><br />
    <p class="meta small">Valid until <span @Html.DxaPropertyMarkup(Model,
"ExpiryDate")>@Html.Date(Model.ExpiryDate)</span></p>
</div>

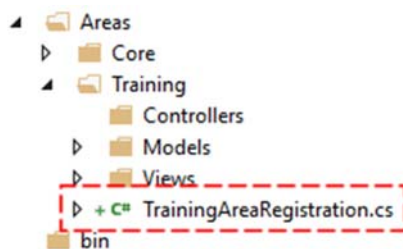
```

Semantic mapping is incorporated in to the View using **@Markup** helper Entity and Property methods.



Semantics enable Editors to edit the content inline in Experience Manager and make the content understandable by search engines when they index your content.

5. Now register your View (you must register all Views in your Area):
 - a. In Solution Explorer, expand '**Area/Training**' and open the '**TrainingAreaRegistration.cs**':



- b. Add the following code to the Registration Class:

Code Box

```
using Sdl.Web.Mvc.Configuration;
using Training.Models;

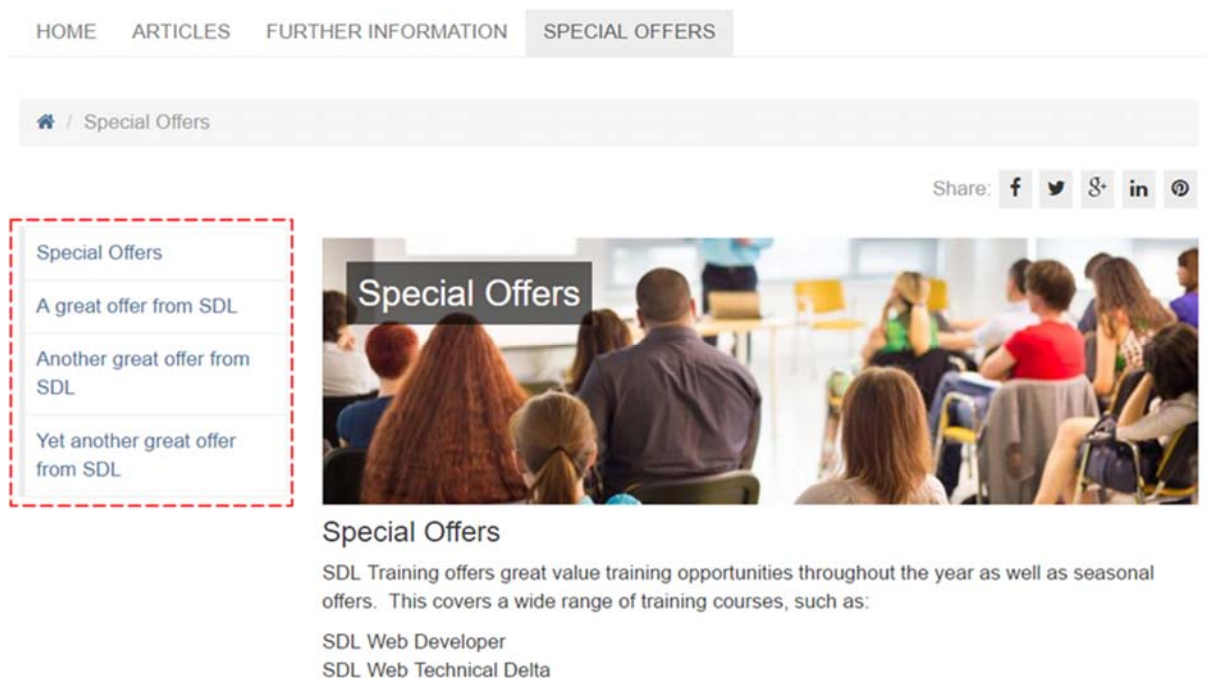
public class TrainingAreaRegistration : BaseAreaRegistration
{
    public override string AreaName
    {
        get
        {
            return "Training";
        }
    }

    protected override void RegisterAllViewModels()
    {
        // Entity Views
        RegisterViewModel("SpecialOffer", typeof(SpecialOffer));
    }
}
```

6. Test the application by running the project in Visual Studio.
 - a. Once your project is running, you will see the home page of the site. In the menu you will see a link to '**Special Offers**'. Click on this link.



- b. In the Special Offers index page you will see a menu on the left with three Special Offers that were created by the business. Click on any of the offers.



- c. When you click on one of the offers, the View and View Model you created are used to populate the page. You can try changing some of the dynamics of the View if you want to try something different.

- Special Offers
- A great offer from SDL
- Another great offer from SDL
- Yet another great offer from SDL



A great offer from SDL

Hit the bullseye with the fantastic offer

[More info](#)

Valid until Sunday, December 25, 2016

Managing Resources

Resources provide snippets of content used across the website. This allows the business to manage and control parts of the website that would normally require IT intervention. In this part of the guide, you will implement a Resource that has already been Published by the business. This will give them control over the text in a site-wide button.

1. The following Resource was created in the CMS by the business and has been Published. This means it is available to the web application.



Configuration Name:
Special Offer Resources

Settings:

- * Name: moreInfoText
- * Value: More Info

Information only!

This shows how the business would create/update a Resource in the CMS.

2. Write out Resource values in the **SpecialOffer** View:
 - a. Update your '**SpecialOffer**' view link (in your **Area** in the project) as follows to write out the '**moreInfoText**' resource:
 - b. Use the **Html.Resource** and replace the View's existing link with the code below.

Code Box

```
<a href="@Model.Link" class="btn btn-primary" @Html.DxaPropertyMarkup(Model, "Link")>@Html.Resource("training.moreInfoText")</a>
```



Resource JSON files are serialized to the file system in **"/system/{version}/resources"**.

3. Run your project and navigate to one of your Special Offers.



A great offer from SDL

Hit the bullseye with the fantastic offer



Valid until Sunday, December 25, 2016

- a. You will see that the text in the button is being injected from the Resources. This has now been applied site wide and is controlled by the business so they can update as and when they need to.

Available course options (classroom)

This guide is intended to provide a quick introduction to development with Digital eXperience Accelerator and SDL Web 8. There are further options for learning that go far beyond the depth of this guide and that are available through the SDL Web Developer course provided by SDL Product Training.

SDL Web 8 Developer Course

The course provides a technical introduction to developing SDL Web powered web sites using the Model, View, Controller (MVC) design pattern.

The course simulates a pre-built environment created through SDL's Digital eXperience Accelerator (DXA) delivering best-practice examples and methodology to accelerate time to market.

This hands-on course introduces a developer, or technical user, to the key advantages of developing with DXA. The course uses Visual Studio or Eclipse as the chosen IDE and assumes you are familiar with terminology and basic methodology associated with developing an MVC site.

During the course, all participants will build a new site from scratch by employing DXA methodology that can be used in any implementation.

Topics covered:

- Introduction to SDL Web 8
- Introduction to MVC Module
- Areas and Modules Module
- Views and View Models Module
- Managing Resources Module
- Managing Configuration Module
- Managing Page Includes Module
- Web site layout (Page and Regions)
- DXA Templates
- Creating Custom Controllers
- Creating Functional Modules
- Automation Development
- Workflow Development

For further details please contact SDL Product Training at learn@sdl.com

Website: [SDL Training](#) (link)