CrownPeak
WEB EXPERIENCE MANAGEMENT

MVC Reference Architecture

Version 1.0

© 2014 CrownPeak Technology, Inc. All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from CrownPeak Technology.

Document History

Author/Editor	Date	Reason for Change	Version
CrownPeak	09/18/2014	Original Version	1.0

Table of Contents

Introduction

The CrownPeak CMS can be used to drive content to a wide range of platforms and services, thanks mainly to the de-coupled architecture and publishing model. This means that it is possible to publish not only web content, but also application code to be executed at runtime upon the publishing platform. In this way, delivery of application following any language framework can be achieved. This reference architecture describes how ASP.net MVC can be delivered from the CrownPeak CMS to a publishing platform that supports .NET Framework 4.5.

Core Objectives

The objective of this project is to show how the following can be achieved using the CrownPeak CMS and standard developer tools, without requiring further product development:

- Build an ASP.net MVC website through Visual Studio 2013:
    - Build an MVC Controller to handle bespoke functionality;
    - Build an MVC Controller to handle page content, look & feel;
    - Handle HTTP 404 & 500 errors gracefully;
    - Upload the published MVC website into the CMS:
        - Enable publication of the MVC website code to the publishing servers using standard CrownPeak workflows.
    - Create & manage navigation wrapping (or MVC View Layouts) from within the CMS;
    - Create & manage page content (or Views) from within the CMS;
    - Support CrownPeak Preview & In-Context editing as standard.

Architectural Challenges

When deploying a custom application using the CrownPeak CMS, typically the content is rendered into the published pages prior to publication making it static by nature. In a traditional MVC application, the page content (or model) is held separately to the layout (or View). This causes challenges for supporting both Preview & In-Context editing gracefully, as there are no in-built features for interpreting & executing these types of code within the CMS. As we care as much about the authoring/editing experience as we do about the customer journey, we have made the architectural decision that we should combine both page content and view together. Traditionalists may not favor this approach, as they lose some of the flexibility that MVC offers (being able to apply views to multiple models), however we feel that this is a necessary trade-off in order of offer a rounded experience.

Reference Architecture (Visual Studio 2013 Project)

CrownPeak has created a .NET MVC Website project, based upon the standard ‘Empty MVC Project’ template within Visual Studio 2013. This is currently located at: *https://github.com/Crownpeak/MVC-reference-architecture*

/App_Start/RouteConfig.cs

To the standard project, we have created two routes within /App_Start/RouteConfig.cs which catch the following:

```
routes.MapRoute(
name: "Contact",
       url: "contact/{action}",
       defaults: new { controller = "Contact", action = "Index" }
);
```

Any request to /contact/* will be caught by our route configuration and passed to the ContactController. This shows the example of how we can build custom functionality within an application, which extends that offered by the CrownPeak CMS. An example of this usage may be where a website has a login/management feature, which is not suitable for deployment within CrownPeak tools. If no custom functionality has been built, then this route can be ignored.

```
routes.MapRoute(
"CrownPeakPage",
"{*page}",
new
{
controller = "CrownPeakPage",
action = "Page"
      }
);
```

The second is a ‘catch-all’ for any route not handled by our custom route configurations. This is designed to ensure that the website behaves as a content managed site, understanding the content being requested and looking for the published content on disk.

/Controllers/CrownPeak/CrownPeakPageController.cs

The CrownPeakPageController is a custom controller, built by CrownPeak that is used to interpret the page being requested (from the Url), and look for the appropriate view file on disk (published by the CrownPeak CMS). In the event of the view file not being found, we can assume that the page doesn't exist, and therefore display an HTTP404 error to the user.

```
public class CrownPeakPageController : Controller
{
	public ActionResult Page()
	{
		if (Request.Url == null) return null;
		var absolutePath =
Request.Url.AbsolutePath.Replace(Request.ApplicationPath, "");
		if (!absolutePath.StartsWith("/")) absolutePath = "/" +
absolutePath;
		if (absolutePath == "/") absolutePath =
ConfigurationManager.AppSettings["CrownPeak:DefaultHome"];
		var viewFile = GetViewFileLocation(absolutePath);
		if (System.IO.File.Exists(Server.MapPath(viewFile)))
return View(viewFile);
		Response.StatusCode = 404;
		Return
View(GetViewFileLocation(ConfigurationManager.AppSettings["CrownPeak:
Default404"]));
	}

	private static string GetViewFileLocation(string viewFile)
	{
		return new
StringBuilder().Append(ConfigurationManager.AppSettings["CrownPeak:Vi
ewsRoot"]).Append(viewFile).Append(ConfigurationManager.AppSettings["
CrownPeak:ViewsFileExtension"]).ToString();
	}
}
```

You will note a number of calls to the System.Configuration.ConfigurationManager.AppSettings collection, these are configured within the web.config file, as follows:

```
<appSettings>
      <add key="CrownPeak:ViewsRoot" value="~/Views/CrownPeak" />
      <add key="CrownPeak:ViewsFileExtension" value=".cshtml" />
      <add key="CrownPeak:DefaultHome" value="/index" />
      <add key="CrownPeak:Default404" value="/404" />
      <add key="CrownPeak:Default500" value="/500" />
</appSettings>
```

Essentially, within this we are setting the default location for view files to be searched and the default file-extension. In addition, we are also setting the default URLs for the homepage (where /index is not specified on the URL), as well as the HTTP404 & 500 friendly page locations.

You will note that error handling & reporting has been deliberately omitted from this example, but is expected to have been included within any production deployments.

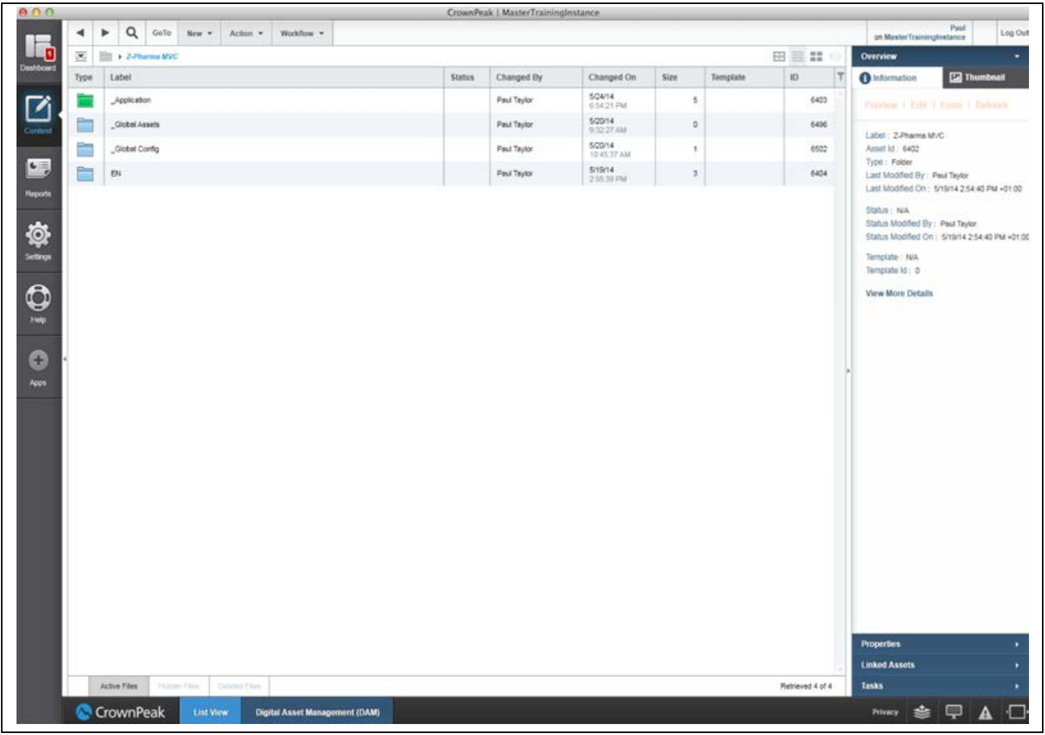
Reference Architecture (CrownPeak Site)

A website folder has been created within the CrownPeak MasterTrainingInstance which has been configured to run this application, and to publish content into the ASP.net MVC Website project.

The website is currently located at: *http://mtistage.cp-access.com/z-pharma-mvc/*

Folder Structure

The following folder structure has been created within the CrownPeak CMS website folder.



Type	Label	Status	Changed By	Changed On	Size	Template	ID
🟩	_Application		Paul Taylor	5/24/14 6:54:21 PM	5		6403
🟦	_Global Assets		Paul Taylor	5/20/14 9:32:27 AM	0		6496
🟦	_Global Config		Paul Taylor	5/20/14 10:45:37 AM	1		6502
🟦	EN		Paul Taylor	5/19/14 2:55:39 PM	3		6404

Label : Z-Pharma MVC
Asset Id : 6402
Type : Folder
Last Modified By : Paul Taylor
Last Modified On : 5/19/14 2:54:40 PM +01:00

Status : N/A
Status Modified By : Paul Taylor
Status Modified On : 5/19/14 2:54:40 PM +01:00

Template : N/A
Template Id : 0

View More Details

CrownPeak Master Training Instance

The _Application folder contains the published code from the Visual Studio 2013 project, has basic workflow attached, and has had its publishing properties set to publish to the root of the website (/). In addition, an IT ticket was created to enable this as an 'application' within IIS and to run under an application pool using the .NET Framework 4.0.

As with all standard CrownPeak projects, we have the concept of a Global Configuration file (for setting page title, metadata and global content). This project is no different, and the Global Configuration asset can be located within the Global Config folder.

Again, we have a Global Assets folder, which following best practice will contain assets available to all languages within the published site.

We also have an /EN folder, which following standard Translation Model Framework (TMF) patterns, will enable cloning of content on a per-language basis, as required.

Within the /EN folder, there are two sub-folders:

- Locale Config – this contains a Locale Configuration asset, which is based upon the same template as the Global Configuration one. This will be used to override and of the Global Configuration settings, to the locale-specific requirements. By default, if not overridden, the Global Configuration will be displayed;
- Layout – this is the MVC equivalent of the CrownPeak NavWrapper. It has no input template file, only an output template file. It gathers information from either Global Configuration or Locale Configuration during publishes, and writes this to the publishing severs. *You will note that the filename.aspx template file ensures that this file is always written to /Views/Shared/_Layout.cshtml.*

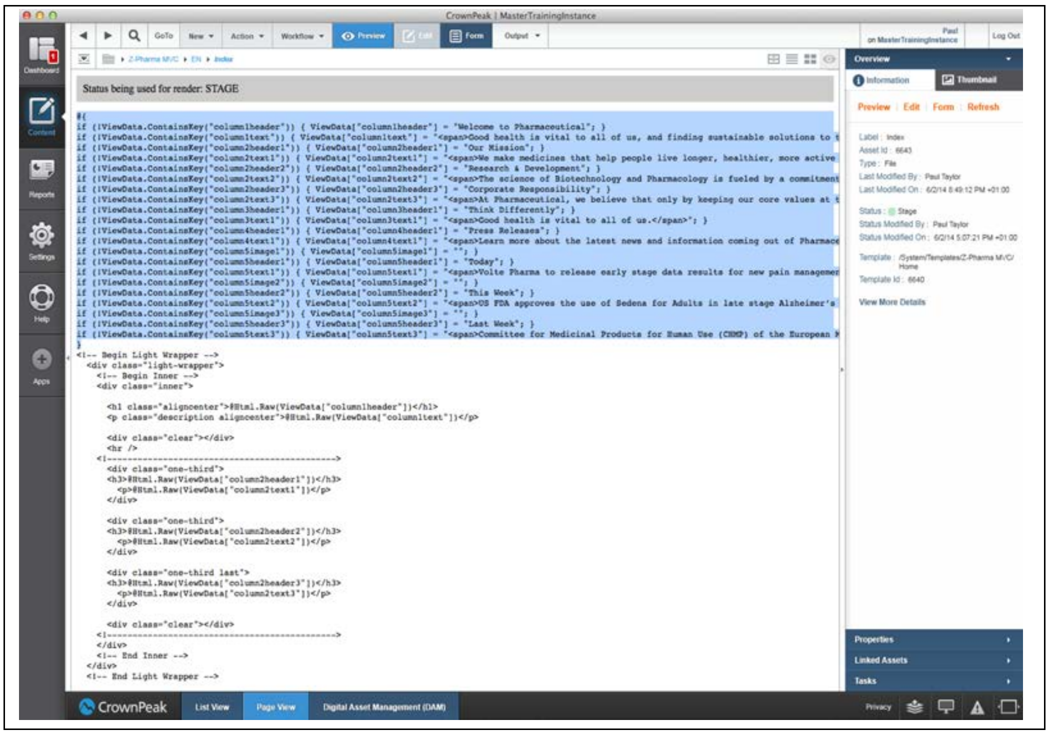
There is an 'Index' asset in the root of the /EN folder. This is the homepage asset. The filename.aspx template file ensures that this is always written to /Views/CrownPeak/index.cshtml. *It is expected that locale variants of this would be written to /Views/CrownPeak/{locale_variant}/index.cshtml.*

All of the output.aspx template files have been configured to support both Preview & In-Context editing, by manipulating what is rendered, depending upon the value of context.IsPublishing. See ‘Preview’ and ‘View Output’ options for further clarity.

Razor Syntax in Published Views

We have already established that we are accepting the combination of page content (models) and layout (views) within this reference architecture; therefore it would have been totally acceptable to simply bake page content into the view without the ability to manipulate this in any way. However, I wanted to show how it would be possible to further manipulate the content using runtime-based Razor syntax.

Each output.aspx template file renders the page content (whether Global or Locale Configuration, or page-based data) into the Razor syntax-based ViewData collection variable. This is done at the top of the template file.



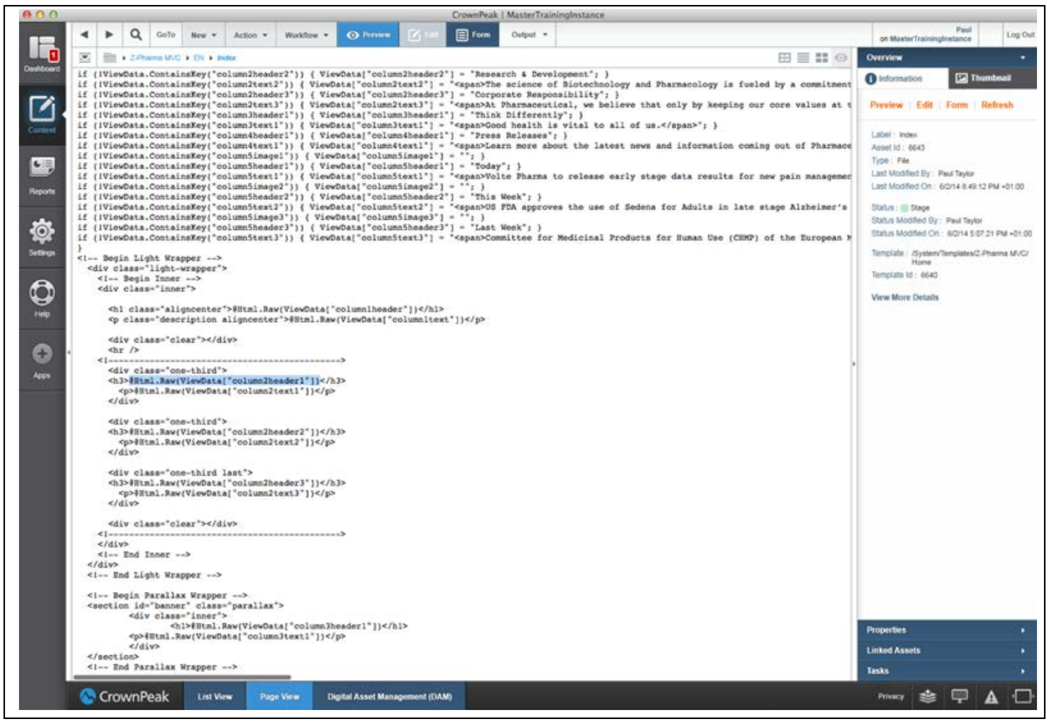
Status being used for render: STAGE

```
@{
if (!ViewData.ContainsKey("column1header")) { ViewData["column1header"] = "Welcome to Pharmaceutical"; }
if (!ViewData.ContainsKey("column1text")) { ViewData["column1text"] = "<span>Good health is vital to all of us, and finding sustainable solutions to t
if (!ViewData.ContainsKey("column2header1")) { ViewData["column2header1"] = "Our Mission"; }
if (!ViewData.ContainsKey("column2text1")) { ViewData["column2text1"] = "<span>We make medicines that help people live longer, healthier, more active
if (!ViewData.ContainsKey("column2header2")) { ViewData["column2header2"] = "Research & Development"; }
if (!ViewData.ContainsKey("column2text2")) { ViewData["column2text2"] = "<span>The science of Biotechnology and Pharmacology is fueled by a commitment
if (!ViewData.ContainsKey("column2header3")) { ViewData["column2header3"] = "Corporate Responsibility"; }
if (!ViewData.ContainsKey("column2text3")) { ViewData["column2text3"] = "<span>At Pharmaceutical, we believe that only by keeping our core values at t
if (!ViewData.ContainsKey("column3header1")) { ViewData["column3header1"] = "Think Differently"; }
if (!ViewData.ContainsKey("column3text1")) { ViewData["column3text1"] = "<span>Good health is vital to all of us.</span>"; }
if (!ViewData.ContainsKey("column4header1")) { ViewData["column4header1"] = "Press Releases"; }
if (!ViewData.ContainsKey("column4text1")) { ViewData["column4text1"] = "<span>Learn more about the latest news and information coming out of Pharmace
if (!ViewData.ContainsKey("column5image1")) { ViewData["column5image1"] = ""; }
if (!ViewData.ContainsKey("column5header1")) { ViewData["column5header1"] = "Today"; }
if (!ViewData.ContainsKey("column5text1")) { ViewData["column5text1"] = "<span>Volte Pharma to release early stage data results for new pain managemen
if (!ViewData.ContainsKey("column5image2")) { ViewData["column5image2"] = ""; }
if (!ViewData.ContainsKey("column5header2")) { ViewData["column5header2"] = "This Week"; }
if (!ViewData.ContainsKey("column5text2")) { ViewData["column5text2"] = "<span>US FDA approves the use of Sedena for Adults in late stage Alzheimer's
if (!ViewData.ContainsKey("column5image3")) { ViewData["column5image3"] = ""; }
if (!ViewData.ContainsKey("column5header3")) { ViewData["column5header3"] = "Last Week"; }
if (!ViewData.ContainsKey("column5text3")) { ViewData["column5text3"] = "<span>Committee for Medicinal Products for Human Use (CHMP) of the European M
}
<!-- Begin Light Wrapper -->
  <div class="light-wrapper">
    <!-- Begin Inner -->
    <div class="inner">

      <h1 class="aligncenter">@Html.Raw(ViewData["column1header"])</h1>
      <p class="description aligncenter">@Html.Raw(ViewData["column1text"])</p>

      <div class="clear"></div>
      <hr />
    <!------------------------------------------------>
      <div class="one-third">
      <h3>@Html.Raw(ViewData["column2header1"])</h3>
        <p>@Html.Raw(ViewData["column2text1"])</p>
      </div>

      <div class="one-third">
      <h3>@Html.Raw(ViewData["column2header2"])</h3>
        <p>@Html.Raw(ViewData["column2text2"])</p>
      </div>

      <div class="one-third last">
      <h3>@Html.Raw(ViewData["column2header3"])</h3>
        <p>@Html.Raw(ViewData["column2text3"])</p>
      </div>

      <div class="clear"></div>
    <!------------------------------------------------>
    </div>
    <!-- End Inner -->
  </div>
  <!-- End Light Wrapper -->
```

Label : Index
Asset Id : 6643
Type : File
Last Modified By : Paul Taylor
Last Modified On : 6/2/14 8:49:12 PM +01:00
Status : 🟩 Stage
Status Modified By : Paul Taylor
Status Modified On : 6/2/14 5:07:21 PM +01:00
Template : /System/Templates/Z-Pharma MVC/Home
Template Id : 6640
View More Details

View Data Collection

Having added page content to the ViewData collection, we can then use this within the runtime as we desire. In this example, we have simply rendered this content in the correct location within the templates.



```
if (!ViewData.ContainsKey("column2header2")) { ViewData["column2header2"] = "Research & Development"; }
if (!ViewData.ContainsKey("column2text2")) { ViewData["column2text2"] = "<span>The science of Biotechnology and Pharmacology is fueled by a commitment
if (!ViewData.ContainsKey("column2header3")) { ViewData["column2header3"] = "Corporate Responsibility"; }
if (!ViewData.ContainsKey("column2text3")) { ViewData["column2text3"] = "<span>At Pharmaceutical, we believe that only by keeping our core values at t
if (!ViewData.ContainsKey("column3header1")) { ViewData["column3header1"] = "Think Differently"; }
if (!ViewData.ContainsKey("column3text1")) { ViewData["column3text1"] = "<span>Good health is vital to all of us.</span>"; }
if (!ViewData.ContainsKey("column4header1")) { ViewData["column4header1"] = "Press Releases"; }
if (!ViewData.ContainsKey("column4text1")) { ViewData["column4text1"] = "<span>Learn more about the latest news and information coming out of Pharmace
if (!ViewData.ContainsKey("column5image1")) { ViewData["column5image1"] = ""; }
if (!ViewData.ContainsKey("column5header1")) { ViewData["column5header1"] = "Today"; }
if (!ViewData.ContainsKey("column5text1")) { ViewData["column5text1"] = "<span>Volte Pharma to release early stage data results for new pain managemen
if (!ViewData.ContainsKey("column5image2")) { ViewData["column5image2"] = ""; }
if (!ViewData.ContainsKey("column5header2")) { ViewData["column5header2"] = "This Week"; }
if (!ViewData.ContainsKey("column5text2")) { ViewData["column5text2"] = "<span>US FDA approves the use of Sedena for Adults in late stage Alzheimer's
if (!ViewData.ContainsKey("column5image3")) { ViewData["column5image3"] = ""; }
if (!ViewData.ContainsKey("column5header3")) { ViewData["column5header3"] = "Last Week"; }
if (!ViewData.ContainsKey("column5text3")) { ViewData["column5text3"] = "<span>Committee for Medicinal Products for Human Use (CHMP) of the European M
}
<!-- Begin Light Wrapper -->
  <div class="light-wrapper">
    <!-- Begin Inner -->
    <div class="inner">

      <h1 class="aligncenter">@Html.Raw(ViewData["column1header"])</h1>
      <p class="description aligncenter">@Html.Raw(ViewData["column1text"])</p>

      <div class="clear"></div>
      <hr />
    <!----------------------------------------------->
      <div class="one-third">
      <h3>@Html.Raw(ViewData["column2header1"])</h3>
        <p>@Html.Raw(ViewData["column2text1"])</p>
      </div>

      <div class="one-third">
      <h3>@Html.Raw(ViewData["column2header2"])</h3>
        <p>@Html.Raw(ViewData["column2text2"])</p>
      </div>

      <div class="one-third last">
      <h3>@Html.Raw(ViewData["column2header3"])</h3>
        <p>@Html.Raw(ViewData["column2text3"])</p>
      </div>

      <div class="clear"></div>
    <!----------------------------------------------->
    </div>
    <!-- End Inner -->
  </div>
  <!-- End Light Wrapper -->

  <!-- Begin Parallax Wrapper -->
  <section id="banner" class="parallax">
          <div class="inner">
                  <h1>@Html.Raw(ViewData["column3header1"])</h1>
          <p>@Html.Raw(ViewData["column3text1"])</p>
          </div>
  </section>
  <!-- End Parallax Wrapper -->
```

Label : Index
Asset Id : 6643
Type : File
Last Modified By : Paul Taylor
Last Modified On : 6/2/14 8:49:12 PM +01:00

Status : 🟩 Stage
Status Modified By : Paul Taylor
Status Modified On : 6/2/14 5:07:21 PM +01:00

Template : /System/Templates/Z-Pharma MVC/Home
Template Id : 6640