AxCMS.net

Integrating ASP.NET MVC framework with AxCMS.net

Some facts about ASP.NET MVC Framework

ASP.NET MVC Framework is based on the Model-View-Controller architectural pattern, separating the data access, business logic and presentation using the routing capabilities of ASP.NET.

The main benefits of using ASP.NET MVC Framework are:

- Friendly URL's which are intuitive to use (you can navigate the site by just changing URL or removing it's parts)
- Clear separation of concerns (you always know where to put your Business, Data Access, and Presentation logic)
- Greater control over HTML output in MVC Views than in WebForms (which in turn, gives you easier integration with JavaScript)
- Test-Driven Development (With ASP.NET MVC Framework it's easy to test controller classes without the need of emulating .aspx pages like in unit testing for Web Forms)

It is a **different paradigm** of developing with ASP.NET **from Web Forms**, instead of basing user interaction directly with .aspx pages with **code-behind**, **Postback**, **ViewState**, **events**, you create a set of **routing rules** which will analyse the URL of user's **http request** and based on that information, call the **Action** of appropriate **Controller** class which will look for appropriate **Model,** pass data from it to .aspx page (**View**) and return the parsed result in **http response**. A different **Action** can be called for same URL based on either browser's **GET** or **POST** verb.

Model - usually located under **Models** folder in your project, class or a set of classes responsible for handling the data access and business logic of your application (Model can be separated into two tiers to allow better separation of concerns).

View - in ASP.NET MVC framework, a **View** is an .aspx page inheriting from **System.Web.Mvc.ViewPage** (or strongly typed **System.Web.Mvc.ViewPage<ModelClass>**).
By convention, MVC View page should contain only the logic nessesary to render the information from entity(Model) it receives from **Controller** class.

Controller - a class inheriting from **System.Web.Mvc.Controller,** it responsibilities are receiving data from appropriate **Model** class and passing it to appropriate **View**. Methods of **Controller** class are called **Actions**, when request comes in from user's browser, it is run against a set of routing rules and passed to the appropriate Controller's **Action** method.

The main resource for starting out with ASP.NET MVC Framework is:

http://www.asp.net/mvc/

Using ASP.NET MVC Framework in context of AxCMS.net

It is possible to make Live part of your site run on ASP.NET MVC routing engine, that way you will be able to develop **Controller** and **Model** classes (**Modeltables** can be **placed** inside the **Live** database of your project) inside the **YourProject.BL** project and have GUI editing capabilities of AxCMS.net for **Views**, with the benefits that AxCMS.net offers like user management, content management, etc. on backend.

Installing ASP.NET MVC Framework

ASP.NET MVC framework can be installed through standard .msi installer,

http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=1491

For deploying ASP.NET MVC based projects to machines without ASP.NET MVC installed, MVC specific dlls can be extracted to be shipped with your project:

http://haacked.com/archive/2008/11/03/bin-deploy-aspnetmvc.aspx

References to be added to YourProject.BL

System.Web.Routing,

System.Web.Abstractions,

System.Web.Mvc

Editing MS/LS Web.config files

You need to edit your LS and MS configs to include MVC specific assemblies.

For LS, you also need to register Routing related assemblies, as well as Modules and Handlers.

Also, you may need to register some .net 3.5 assemblies like Linq if they are not registered already, so inline code works on view pages.

Check that your <pages> opening tag looks like this and it's <namespaces> and <controls> contain the following:

```
<pages
 validateRequest="false"
 pageParserFilterType="System.Web.Mvc.ViewTypeParserFilter, System.Web.Mvc, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
 pageBaseType="System.Web.Mvc.ViewPage, System.Web.Mvc, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
userControlBaseType="System.Web.Mvc.ViewUserControl, System.Web.Mvc, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
<namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Routing" />
      <add namespace="System.Linq" />
      <add namespace="System.Collections.Generic" />
</namespaces>
 <controls>
<add assembly="System.Web.Mvc, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"
namespace="System.Web.Mvc" tagPrefix="mvc" />
 </controls>
</pages>
```

And <assemblies> inside the <compilation> tag includes the following assemblies:

```
<compilation>
<assemblies>
   <add assembly="System.Core, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
   <add assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
   <add assembly="System.Web.Abstractions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
   <add assembly="System.Web.Routing, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
   <add assembly="System.Web.Mvc, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
   <add assembly="System.Data.DataSetExtensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
   <add assembly="System.Xml.Linq, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
   <add assembly="System.Data.Linq, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
</assemblies>
</compilation>
```

Then you need to register URL routing module (LS.web.config only)

for IIS6:

```
<httpModules>
   <add name="UrlRoutingModule" type="System.Web.Routing.UrlRoutingModule, System.Web.Routing, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
</httpModules>
```

or for IIS7

```
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true">
    <remove name="UrlRoutingModule" />
    <add name="UrlRoutingModule" type="System.Web.Routing.UrlRoutingModule, System.Web.Routing, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
  </modules>
</system.webServer>
```

Enable Wildcard mapping in IIS

To enable url's without extensions to be processed by ISAPI, in **IIS/virtual directory/properties/Application configuration mappings** , click **Insert** near Wildcard application maps and **add path to ISAPI dll** there –

for x32 machines - c:\windows\microsoft.net\framework\v2.0.50727\aspnet_isapi.dll

for x64 machines - c:\windows\microsoft.net\framework64\v2.0.50727\aspnet_isapi.dll

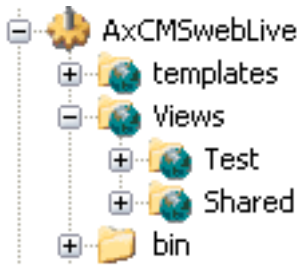
Registering View directories in IIS

In IIS, under your AxCMSwebLive_YourProject create virtual directory named **Views** (path not relevant)**,**

and another virtual directory under it named **Shared** with path pointing to:

C:\Projects\AxCMS_YourProject\AxCMSwebLive_YourProject\publish

You can create additional virtual directories called with names of your **Controllers** pointing to same path if you like.



Modifying Global.asax and adding Global.asax.cs

To enable MVC route/controller mappings, you need to create a new Global.asax.cs file in

c:\Projects\AxCMSwebLive_YourProject

directory and edit it **Global.asax** to look like this:

<%@ Application Src="Global.asax.cs" Inherits="YourProject.Global" %>

This should be in contents of your **Global.asax.cs**

(change the YourProject to your actual project name)

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.SessionState;
using System.Xml.Linq;
using System.Web.Mvc;
using System.Web.Routing;
using AxCMS.AxCMSwebLive;
using Axinom.Framework;
using Axinom.Framework.Data;

// change this name to your project actual name
namespace YourProject
{
   public class Global : AxCMS.AxCMSwebLive.Global
   {

      public static void RegisterRoutes(RouteCollection routes)
      {
```

```
//Ignore some extensions so usual .AxCMS pages work as well in Live

            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.IgnoreRoute("{resource}.aspx/{*pathInfo}");

//If you are using custom extension other than .AxCMS, change it here

            routes.IgnoreRoute("{resource}.AxCMS/{*pathInfo}");

//Implement your routing logic here:

            routes.MapRoute(
            "UserRoute",                        // Route name
            "Users/{id}",                       // URL with parameters
            new { controller = "Test", action = "Users", id = "1001" }  // Parameter defaults
           );

            routes.MapRoute(
            "TestRoute",                        // Route name
            "Index/{page}",                      // URL with parameters
            new { controller = "Test", action = "Index", id = "" }  // Parameter defaults
           );
       }

      protected override void Application_Start(Object sender, EventArgs e)
      {
          base.Application_Start(sender, e);
          RegisterRoutes(RouteTable.Routes);
      }
   }
}
```

Developing Controllers and Models

Create **Controllers** directory in **YourProject.BL** and place your classes derived from **System.Web.Mvc.Controller** there. By convention, **Controller** class must be named **ClassName**Controller, with Controller as last part of name.

You can place **Models** also in **YourProject.BL** project.

Models are representing both Business Logic and Data Access in MVC, so you may want to further separate them and implement the Data Access logic through web services.

Developing MVC View Templates

View pages can be based on normal templates with slight difference:

To have access to **ViewData** and **HtmlHelper** classes, template should inherit from:

System.Web.Mvc.ViewPage

(or strongly typed **System.Web.Mvc.ViewPage<ModelClass>**)

You can create pages based on these templates as usual, to follow ASP.NET MVC convention of no code-behind you should **only put only code for parsing entity(Model) fields there** and **use static controls that render to inline code when published.**

Developing static elements for View Pages

It is possible to develop dynamic controls for View pages as well, as MVC View pages still inherit from Page, but by convention MVC View pages should not have code-behind files and render content editing logic with inline code. To follow this convention, we can make static controls that are rendered as inline code after publishing.

To have access to **ViewData** and **HtmlHelper** classes, control should inherit from:

System.Web.Mvc.ViewUserControl

(or strongly typed **System.Web.Mvc.ViewUserControl<ModelClass>**)

Copying Views to appropriate directories and removing ViewState references from View Pages completely using publishing plugin

You can extend AxCMS publishing process with creating a plugin class and adding handlers for it's **AfterPageUpdate** and **AfterPageDelete** events.

While <form runat="server"></form> tag is present on .aspx web page, minimal ViewState tag will still be present in html. To remove it,

In **AfterPageUpdate** handler,

you can access the page you just published

as **C:\Projects\AxCMS_YourProject\AxCMSwebLive_YourProject\publish\MyView.aspx**

and parse it to remove **<form runat="server"></form>** tag, this will ensure that no ViewState related data is present in the html of live View page.

Then, you can copy the parsed file to appropriate directory under Views in

C:\Projects\AxCMS_YourProject\AxCMSwebLive_YourProject\publish\Views\ViewName\MyView.aspx

There is more than one way to determine the path where published View page goes, you can use a name prefixing convention, or determine it by classified category from **PagePublishEventArgs** argument of **AfterPageUpdate** handler.

In **AfterPageDelete** handler,

use the same logic for determining View Path to delete the file from **\Views\ViewName** directory.