Hello World ASP.NET MVC in C1 CMS

2017-02-15

Orckestra, Europe
Nygårdsvej 16
DK-2100 Copenhagen
Phone +45 3915 7600

www.orckestra.com

Contents

1 Using MVC in C1 CMS

C1 CMS is an ASP.NET-based content management system that allows you to quickly build and manage websites.

The following a step-by-step tutorial for programming tasks you can do in C1 CMS such as embedding an ASP.NET MVC Web application within a website. It is intended for ASP.NET Web developers who are familiar with the MVC pattern in general and ASP.NET MVC and ASP.NET Razor in particular.

Since you will write code that will implement some functionality on a C1 CMS website, we assume that as an ASP.NET developer you work with Visual Studio 2015 or later.

We also assume that you have downloaded and launched C1 CMS in Visual Studio 2015 and installed a sample website. If you haven’t, please refer to “Visual Studio and IIS” and take steps described in it.

C1 CMS allows you to use an ASP.NET MVC Web application on a web page as a CMS function (MVC function). You should take the following steps to complete this task:

1. Install the MVC Functions add-on.
2. Create an ASP.NET MVC Web application and add it to the website.
3. Register it as a CMS Function on the website.
4. Add the MVC application on a page as a CMS Function.

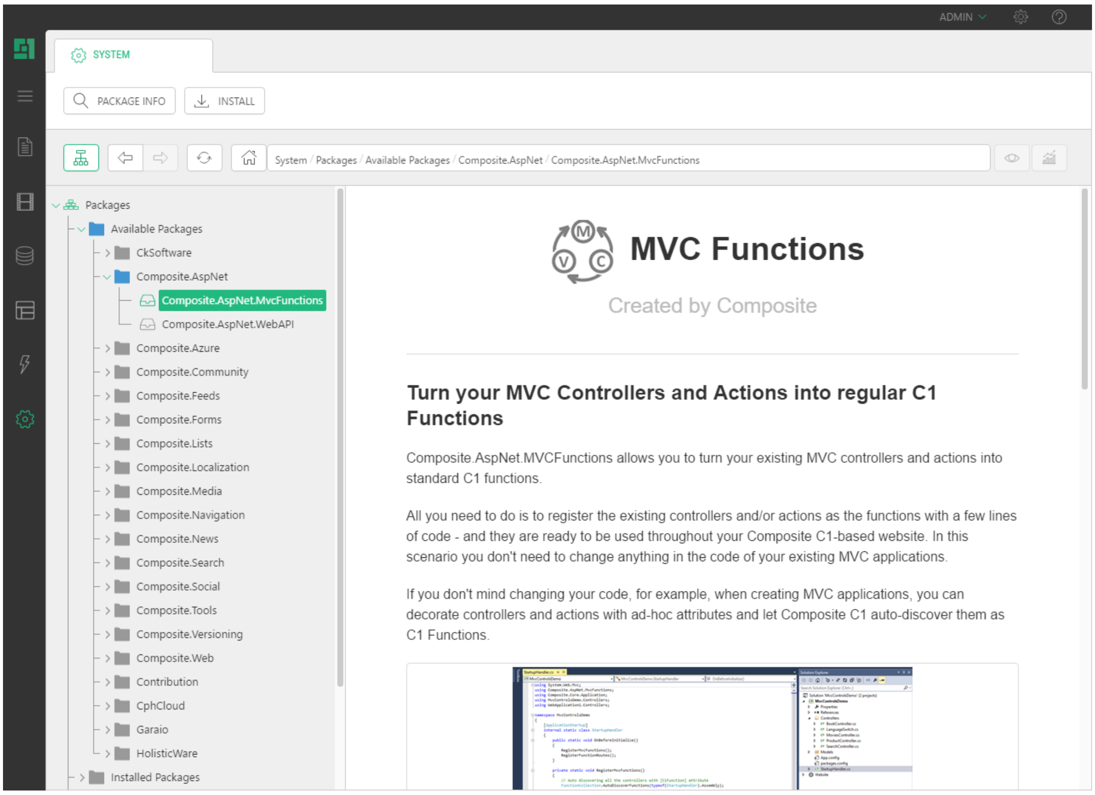
2 Install MVC Functions

The Composite.AspNet.MvcFunctions add-on prepares your website for MVC and enables you to use your MVC application on a page as a MVC function.

You should thus install it only once.

To install Composite.AspNet.MvcFunctions:

1. Launch C1 CMS in Visual Studio.
2. Log into the CMS Console.
3. In the **System** perspective, expand **Packages**, **Available Packages**, **Composite.AspNet**.
4. Select **Composite.AspNet.MvcFunctions** and click **Install**.



MVC Functions

Created by Composite

Turn your MVC Controllers and Actions into regular C1 Functions

Composite.AspNet.MVCFunctions allows you to turn your existing MVC controllers and actions into standard C1 functions.

All you need to do is to register the existing controllers and/or actions as the functions with a few lines of code - and they are ready to be used throughout your Composite C1-based website. In this scenario you don't need to change anything in the code of your existing MVC applications.

If you don't mind changing your code, for example, when creating MVC applications, you can decorate controllers and actions with ad-hoc attributes and let Composite C1 auto-discover them as C1 Functions.

Figure 1: Installing Mvc Functions

5. Complete the wizard.

3 Create an MVC Application and add it to the website

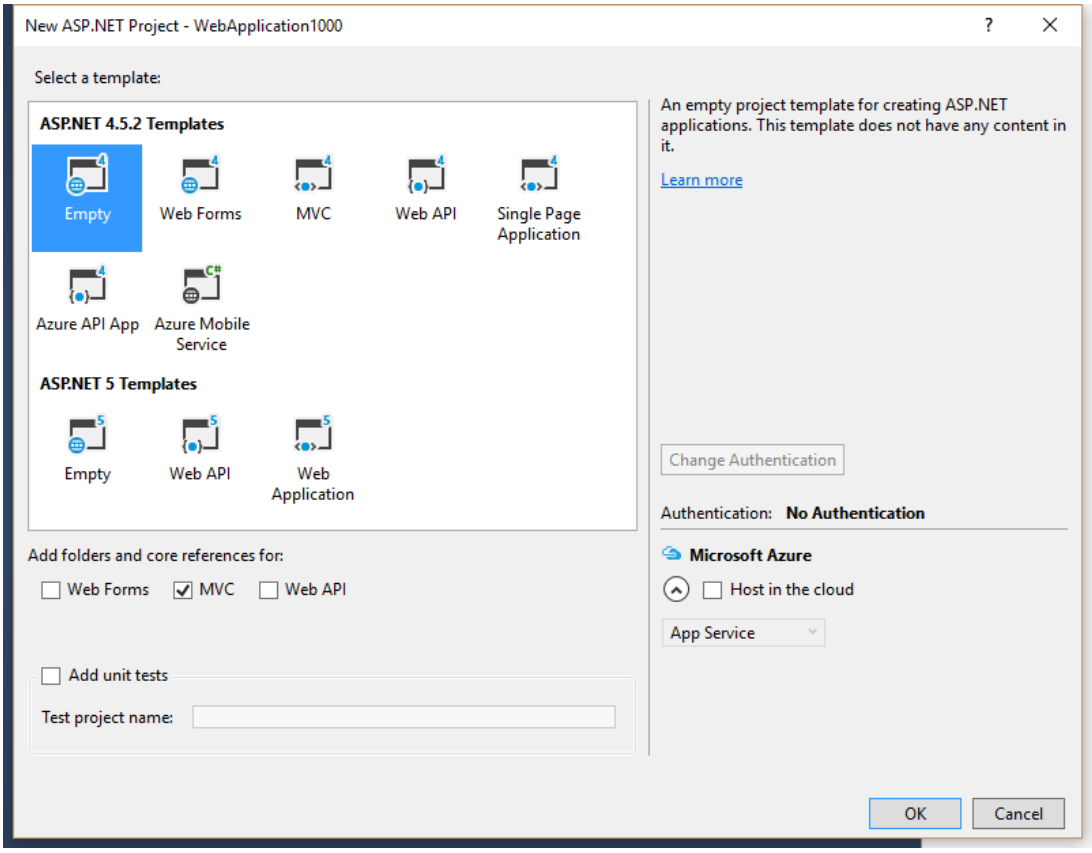
For the purpose of this tutorial we will create a very simple ASP.NET application with one controller - Home.

3.1 Controller

For the purpose of this simple tutorial, we assume that you project is called “Composite.Samples.SimpleMvc”. You may choose a different name and make sure to update the samples below accordingly.

To create a simple MVC Web application, first, create the controller:

1. Create an ASP.NET Application project in Visual Studio (**File** / **Project** / **ASP.NET Web Application**).
2. Select the “Empty” template and select the “MVC” option.



New ASP.NET Project - WebApplication1000

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Add folders and core references for:

☐ Web Forms ☑ MVC ☐ Web API

☐ Add unit tests

Test project name:

☐ Host in the cloud

Figure 2: Creating an empty ASP.NET Web application project with MVC

3. In your project, add a controller called “HomeController” below the “Controller” folder.
4. Replace its content generated by default with the following code:

```
using System.Web.Mvc;
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewData["Method"] = "Normal get";
            return View();
        }
        [HttpPost()]
        public ActionResult Index(string postVar)
        {
            ViewData["Method"] = "A post";
            ViewData["Posted"] = postVar;
            return View();
        }
    }
}
```

5. Save all these files (**File | Save All**).
6. Build the project.
7. Copy the resulting DLL (Composite.Samples.SimpleMvc.dll in this tutorial) to your website’s /Bin folder.

3.2 View

Now on your website, create a view for the controller. (If you have created the view in your project, copy it to the path on your website and update its content as described below.)

1. On your website, create a subfolder called “Home” at ~/Views.
2. Add a new Empty Page (Razor) below and call it “Index”: ~/Views/Home/Index.cshtml
3. Replace its content generated by default with the following code:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    </head>
    <body>
        <div class="alert">Method is @ViewBag.Method: @ViewBag.Posted
</div>
        <form method="post">
            <input type="text" name="postVar" />
            <input type="submit" />
        </form>
    </body>
</html>
```

4. Save the file.

4 Register the controller as a CMS function

Now you need to register the Home controller on the website as an MVC function.

1. On your website, add a class to ~/App_Code called “Startup”: ~/App_Code/Startup.cs. (Create the folder if it is not there.)
2. Replace its content generated by default with the following code:

```
using Composite.AspNet.MvcFunctions;
using Composite.Core.Application;
using System.Web.Mvc;

namespace DemoMVC
{
    [ApplicationStartup]
    internal static class StartupHandler
    {
        public static void OnBeforeInitialize()
        {
            var functions = MvcFunctionRegistry.NewFunctionCollection();

functions.RegisterController<Composite.Samples.SimpleMvc.Controllers.HomeCo
ntroller>("Test.SimpleMvc.Home");

            functions.RouteCollection.MapMvcAttributeRoutes();
            functions.RouteCollection.MapRoute(
                "Default",
                "{controller}/{action}/{id}",
                new { action = "Index", id = UrlParameter.Optional }
                );
        }

        public static void OnInitialized()
        {

        }
    }
}
```

3. Save the file.
4. Restart the server.

There are different ways of registering controllers and actions from you MVC application as CMS functions. Please see ”Registering Functions” for more infromation.

Alternatively, you may want to have the startup code that registers your MVC functions in a project, build it and copy the output DLL in ~/Bin rather than creating the class in ~/App_Code.

5 Use an MVC application as a CMS function on a page

Now you can use the Home controller from your MVC application on a page as a regular CMS function.

First of all, select where you want your MVC application to appear:

1. Log into the CMS Console.
2. From the **Content** perspective, select the page where you want to insert your MVC function.
3. Click **Edit Pag**e on the toolbar. The page will open in Visual Editor.

Then, insert the MVC function (registered as “Test.SimpleMvc.Home” in this tutorial):

1. On the page’s toolbar, click **Insert**, then **Function**.
2. In the **Select Function** window, expand **All Functions** / **Test** / **SimpleMvc** and select the **Home** function.

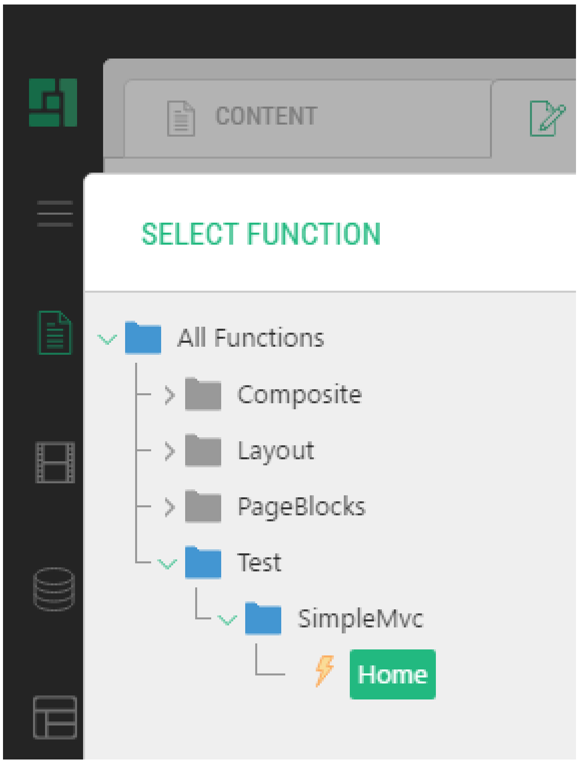
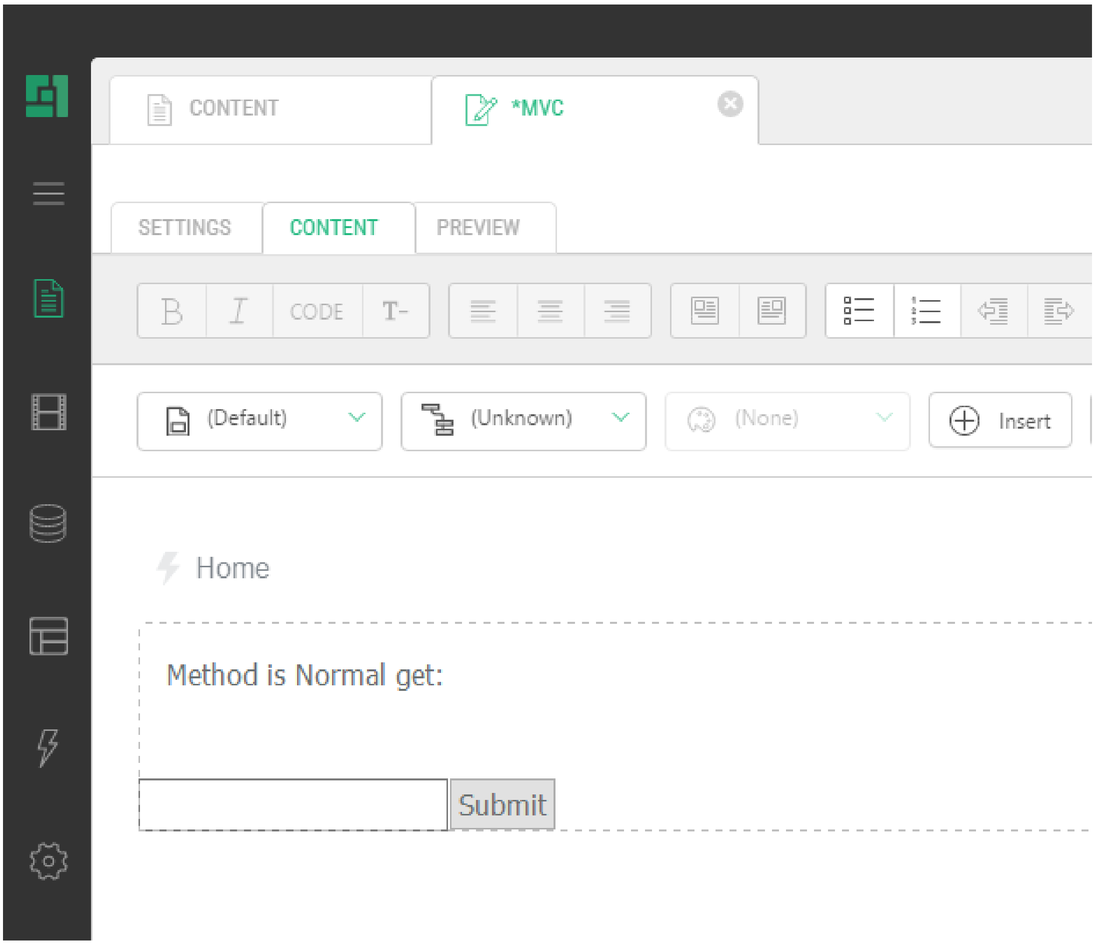


Figure 3: Selecting the regsitered Mvc function

3. Click **OK**.

The function preview will appear on the edited page.

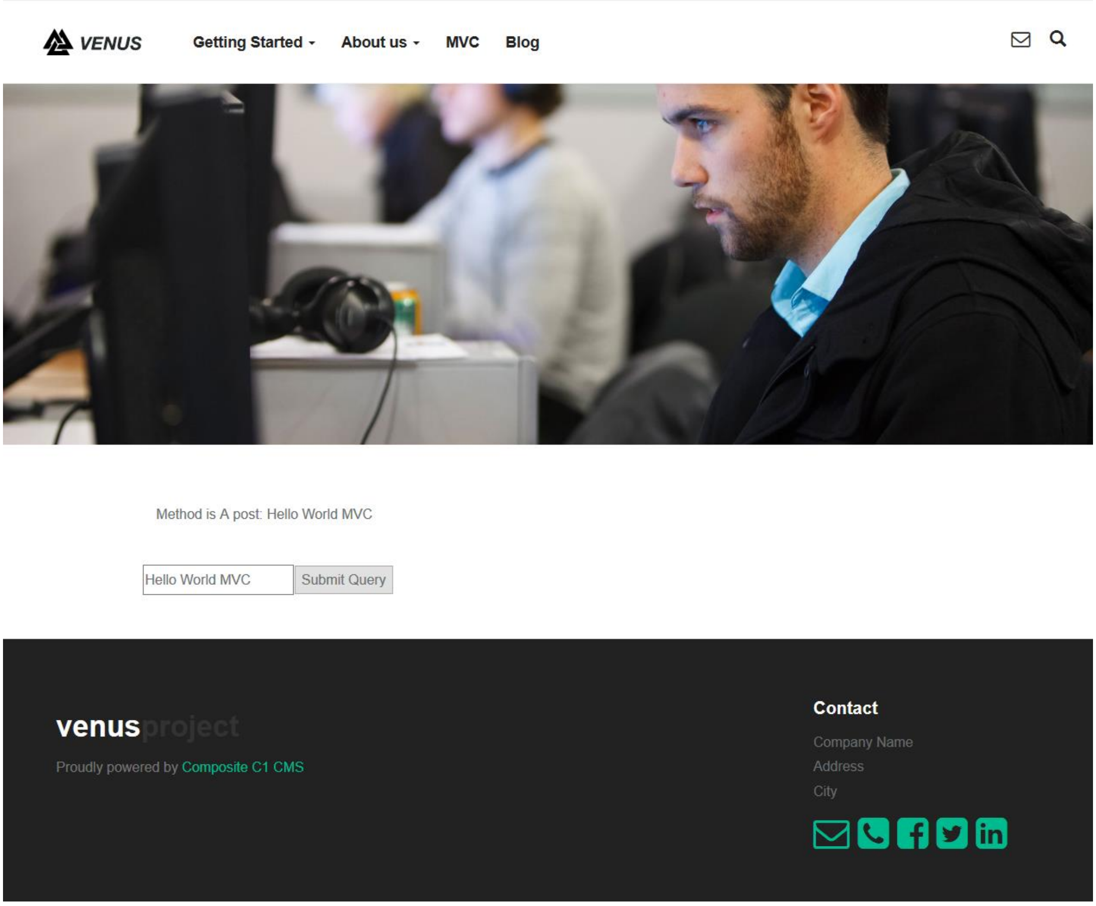


Method is Normal get:

Figure 4: MVC function preview on an edited page

Finally:

1. Save and publish the page.
2. In Visual Studio, press F5 to launch the website in the browser and browse to the page. The page will show the contents of your simple MVC application.



Method is A post: Hello World MVC

Hello World MVC

Figure 5: The view of the MVC function on a published page