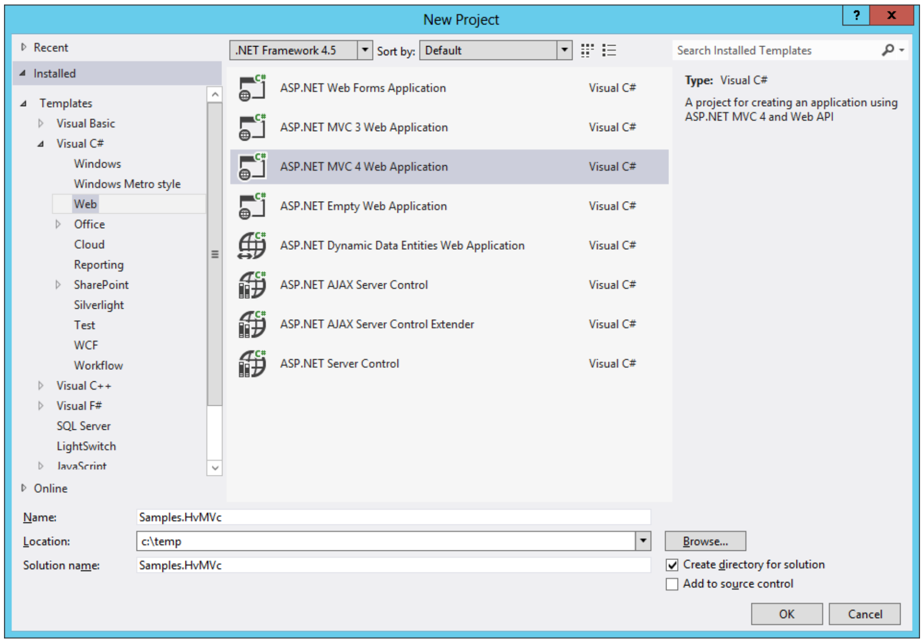
Integrating HealthVault SDK with APS.NET MVC

Author: Mark Arteaga
Date: July 31, 2012

In the previous articles we took a look at how to get a custom application that connects to HealthVault up and running. In this article we'll look at how to get the HealthVault SDK up and running with an ASP.NET MVC website.

We will be using Visual Studio 2012 RC, but the steps should be similar for Visual Studio 2010.

1. Open Visual Studio 2012, if you have not installed you can download the latest VS2012 Release Candidate. You can also download the appropriate Express Version of Visual Studio which are free versions of Visual Studio
2. Create a New ASP.NET 4 Web Application by clicking on **File -> New Project**



New Project

- Recent
- Installed
  - Templates
    - Visual Basic
    - Visual C#
      - Windows
      - Windows Metro style
      - Web
      - Office
      - Cloud
      - Reporting
      - SharePoint
      - Silverlight
      - Test
      - WCF
      - Workflow
    - Visual C++
    - Visual F#
    - SQL Server
    - LightSwitch
    - JavaScript
- Online

.NET Framework 4.5 Sort by: Default

ASP.NET Web Forms Application	Visual C#
ASP.NET MVC 3 Web Application	Visual C#
ASP.NET MVC 4 Web Application	Visual C#
ASP.NET Empty Web Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
ASP.NET AJAX Server Control	Visual C#
ASP.NET AJAX Server Control Extender	Visual C#
ASP.NET Server Control	Visual C#

Search Installed Templates

Type: Visual C#

A project for creating an application using ASP.NET MVC 4 and Web API

Name: Samples.HvMVc

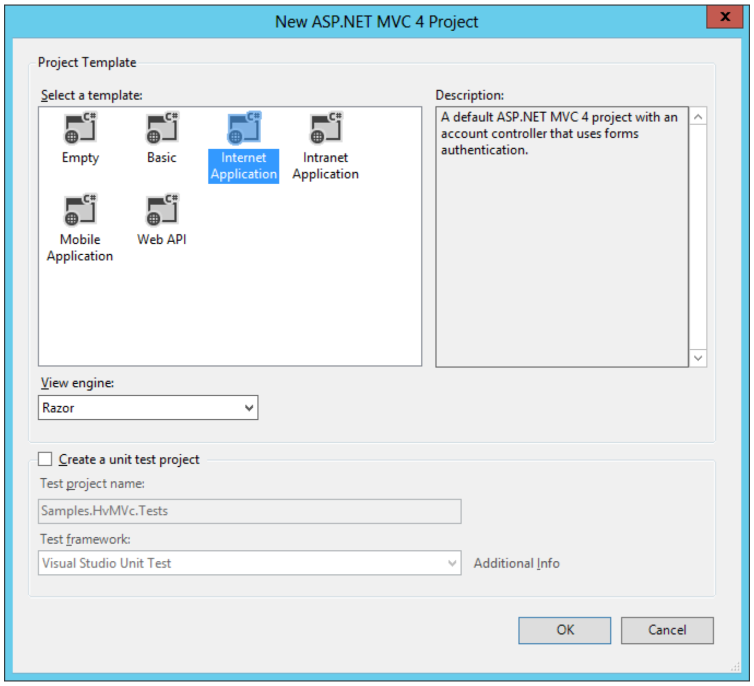
Location: c:\temp

Solution name: Samples.HvMVc

[✓] Create directory for solution

[ ] Add to source control

3. In the **New ASP.NET MVC 4 Project** dialog select **Internet Application** and View Engine with **Razor** selected then click OK



New ASP.NET MVC 4 Project

Project Template

Select a template:

Description:

A default ASP.NET MVC 4 project with an account controller that uses forms authentication.

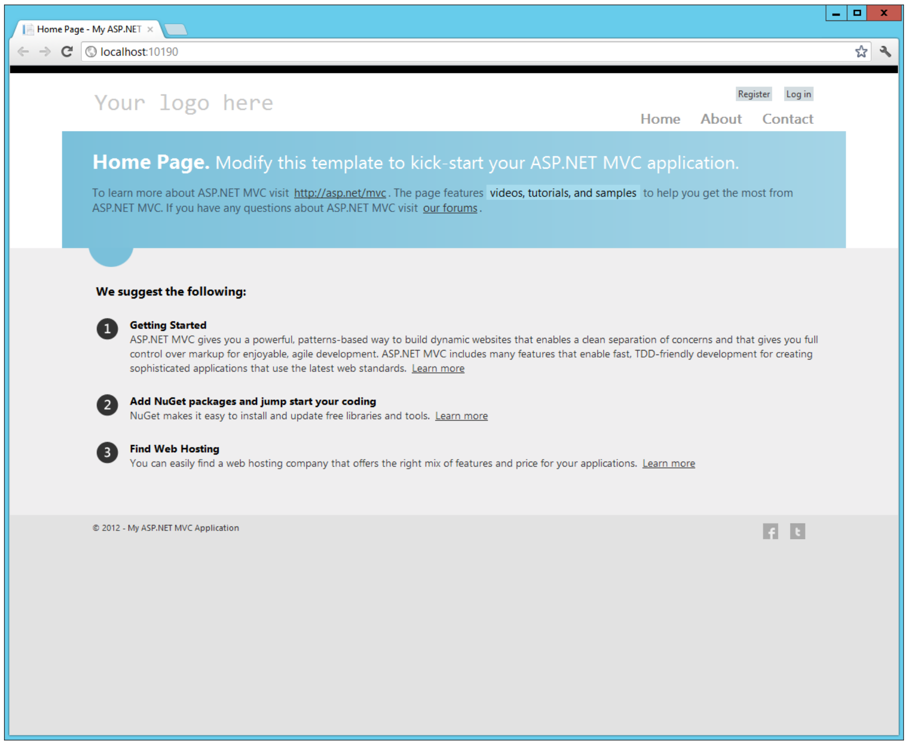
View engine: Razor

Create a unit test project

Test project name: Samples.HvMVc.Tests

Test framework: Visual Studio Unit Test

4. After the project is created run the project and you will get the standard MVC4 template in your browser.



Home Page. Modify this template to kick-start your ASP.NET MVC application.

To learn more about ASP.NET MVC visit http://asp.net/mvc . The page features videos, tutorials, and samples to help you get the most from ASP.NET MVC. If you have any questions about ASP.NET MVC visit our forums .

We suggest the following:

1. **Getting Started**
   ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards. Learn more
2. **Add NuGet packages and jump start your coding**
   NuGet makes it easy to install and update free libraries and tools. Learn more
3. **Find Web Hosting**
   You can easily find a web hosting company that offers the right mix of features and price for your applications. Learn more

© 2012 - My ASP.NET MVC Application

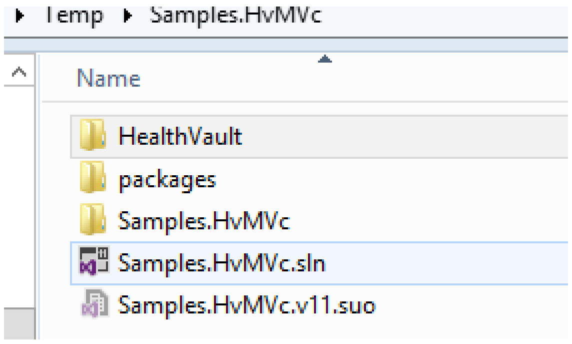
Now that the standard project is up and running, the next few steps will be to get our New ASP.NET MVC Project working with HealthVault SDK

Upgrading HealthVault SDK Projects

This step is optional, but it is a good idea to add the HealthVault SDK source code when you are debugging your application to help figure out what is going on under the covers. You could easily add the assembly references, but my recommendation is to add the source files.

First step we have to take is upgrading the HealthVault projects to VS 2012 RC. The following steps will guide you through this.

1. Unzip the HealthVault source code located under %ProgramFiles%\Microsoft HealthVault\SDK\Source\HealthVaultDLLSource.zip
2. Where your MVC project is located, create a directory called HealthVault
3. Copy the HealthVault source code to the directory where your project is located, in my case C:\Temp\Samples.HvMVc and my directory looks like this

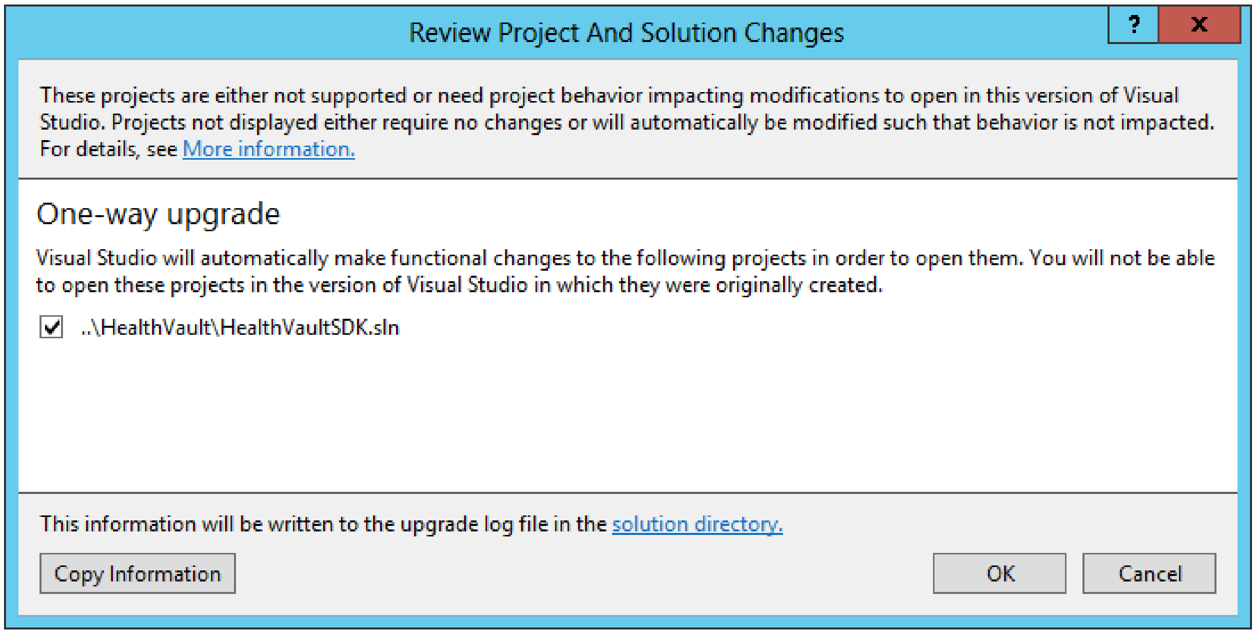


Temp ▸ Samples.HvMVc

Name

- HealthVault
- packages
- Samples.HvMVc
- Samples.HvMVc.sln
- Samples.HvMVc.v11.suo

4. Open up a new instance of VS 2012 RC and open HealthVaultSDK.sln under the HealthVault directory
5. If using VS2012 RC you will be asked to go through a one way upgrade. Read through the information if you like, then click OK to continue



Review Project And Solution Changes

These projects are either not supported or need project behavior impacting modifications to open in this version of Visual Studio. Projects not displayed either require no changes or will automatically be modified such that behavior is not impacted. For details, see More information.

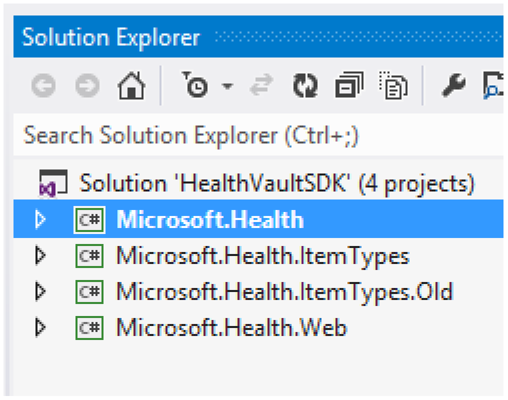
One-way upgrade

Visual Studio will automatically make functional changes to the following projects in order to open them. You will not be able to open these projects in the version of Visual Studio in which they were originally created.

☑ ..\HealthVault\HealthVaultSDK.sln

This information will be written to the upgrade log file in the solution directory.

6. When done the project will open up in VS2012 RC and your project structure should look as follows



Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'HealthVaultSDK' (4 projects)
- Microsoft.Health
- Microsoft.Health.ItemTypes
- Microsoft.Health.ItemTypes.Old
- Microsoft.Health.Web

7. Remove the **Microsoft.Health.ItemTypes.Old** as these are not required for our purposes and are there for legacy reasons.
8. Attempt to Build the project to make sure it compiles.
9. If you successfully compile, shut down this instance of VS2012

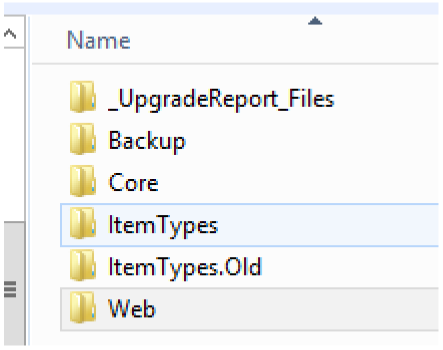
Note that if you are using VS2010, the steps in upgrading will be very similar but I'll leave it up to the reader to explorer that scenario.

Now that we have upgraded the projects, we can add them to our ASP.NET MVC project.

Adding HealthVault SDK Source

In this section we will add the newly converted HealthVault SDK projects to your ASP.NET MVC Project.

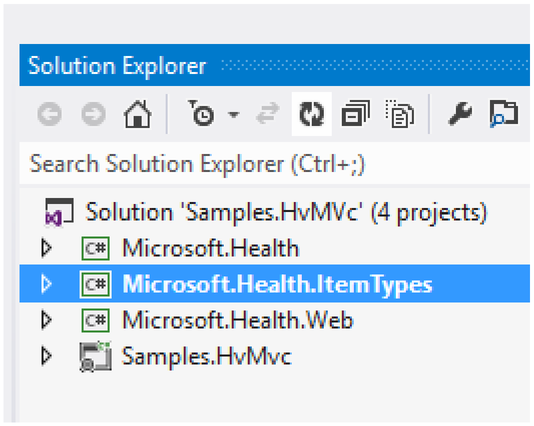
1. In your MVC project, click on **File -> Add -> Existing Project**
2. Navigate to the HealthVault folder you created, should look something like this



Name
- _UpgradeReport_Files
- Backup
- Core
- ItemTypes
- ItemTypes.Old
- Web

3. Add the **Microsoft.Health.csproj** in the **Core** directory
4. Add the **Microsoft.Health.ItemTypes.csproj** in the **ItemTypes** directory
5. Add the **Microsoft.Health.Web.csproj** in the **Web** directory

6. Your project should look similar to this now



Solution 'Samples.HvMVc' (4 projects)
- Microsoft.Health
- Microsoft.Health.ItemTypes
- Microsoft.Health.Web
- Samples.HvMvc

7. Compile the project to make sure it still builds

Setting up Web.Config

In the previous article I went through some of the web.config values generated by the HealthVault Application Manager. We need to use these values in our MVC project so the HealthVault SDK will work as it depends on these key/value pairs.

Add the following to your appSettings in web.config

```
<!-- SPECIFIC VALUES TO HAVE THE HAVE THE HEALTH VAULT SDK WORK-->
<add key="ApplicationId" value="ed058c64-7aee-49c7-970e-bd020ab2b8a3" />
<add key="ShellUrl" value="https://account.healthvault-ppe.com/" />
<add key="HealthServiceUrl" value="https://platform.healthvault-ppe.com/platform/" />
<!-- when we call the SignOut() method on HealthServicePage, it redirects us to the page
below -->
<add key="NonProductionActionUrlRedirectOverride" value="Account/Login" />
<!-- The redirect page (specified above) uses these keys below to redirect to different
      pages based on the response from the shell -->
<add key="WCPage_ActionHome" value="default.aspx" />
<add key="WCPage_ActionAppAuthSuccess" value="default.aspx" />
<add key="WCPage_ActionSignOut" value="SignedOut.aspx" />
<add key="ApplicationCertificateFileName" value="~\App_Data\WildcatApp-ed058c64-7aee-
49c7-970e-bd020ab2b8a3.pfx" />
```

You will also need to add your certificate generated by the HealthVault Application Manager to your project. Follow these steps to add to your project

1. In an Explorer window navigate to the sample project created by HealthVault Application Manager and you will find two directories called **cert** and **website.**
2. Open the **cert** directory
3. Copy the **pfx** file into your MVC project **App_Data** directory.

You will notice the key for the certificate is set in the web.config, but unfortunately, the HealthVault SDK cannot read that directory format structure so we need to change it. Add the following code to your **Global.asax.cs** in the **Application_Start()** method

```
// set the certification file name so the HealthVault Sdk can find it
ConfigurationManager.AppSettings["ApplicationCertificateFileName"] =
Server.MapPath(ConfigurationManager.AppSettings["ApplicationCertificateFileName"]);
```

This will allow us to change the value in web.config and properly map the directory on the server. Once added your **Application_Start()** method should look like this.

```
protected void Application_Start()
{
    // set the certification file name so the HealthVault Sdk can find it
    ConfigurationManager.AppSettings["ApplicationCertificateFileName"] =
Server.MapPath(ConfigurationManager.AppSettings["ApplicationCertificateFileName"]);

    AreaRegistration.RegisterAllAreas();

    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

This will allow us to use the HealthVault SDK which we will be predominantly doing in the next part of this article.

Conclusion

In this first part of this article, we prepared our ASP.NET MVC project to be integrated with HealthVault. We also upgraded the HealthVault SDK source code to VS2012 projects and added them to our MVC project. In part II of this article, we'll start adding code to authenticate with HealthVault, get user information and integrate with SqlServerMembershipProvider.