

## Quick Start Guide to HealthVault SDK

Getting started with HealthVault is fairly easy and there are a few things to be aware of before starting. To get things started with the SDK you will require Visual Studio 2010 but after the initial setup, we'll be using Visual Studio 2012 RC.

### MSDN Documentation

We will not be covering HealthVault from an architecture perspective and all technical aspect of HealthVault, but I do recommend you get familiar with it by reading the MSDN documentation. Here are a list of links I found useful during development of Asthma Journal

1. [Getting Started With Microsoft HealthVault SDK](#)
2. [Microsoft HealthVault How To Guides](#)
3. [Microsoft HealthVault Platform Overview](#)
4. [Microsoft HealthVault Application Integration Recommendations](#)

### URLs for HealthVault

There are also some URLs that you should be aware of as HealthVault has a production environment and a Pre-production environment (PPE). For a complete list see [HealthVault documentation](#) but here is a list of PPE URLs (note: the US PPEs will work outside of US)

Configuration	Value	Description
Shell PPE URL	<a href="https://account.healthvault-ppe.com/">https://account.healthvault-ppe.com/</a>	Used by developers to create HealthVault accounts for developing and testing applications in U.S. PPE.
Platform PPE URL	<a href="https://platform.healthvault-ppe.com/platform">https://platform.healthvault-ppe.com/platform</a>	The HealthVault platform service URL for U.S. PPE.
Platform PPE IP Range	65.55.202.96/27	The IP range for U.S. HealthVault platform PPE.
Application Configuration Center (ACC)	<a href="https://config.healthvault-ppe.com">https://config.healthvault-ppe.com</a>	The ACC URL for creating and managing apps in the U.S. HealthVault instance.

The following is a 'quick start' guide to getting started and will get you up and running relatively quickly.

1. Make sure you have **Visual Studio 2010** installed as the SDK tools depend on this

2. You will need a Windows Live Id to use HealthVault. I recommend you create a new one for testing purposes and not use your existing one. Go to [account.live.com](http://account.live.com) to create a new one or manage your existing one.
3. Install the Microsoft HealthVault SDK at <http://www.microsoft.com/en-us/download/details.aspx?id=3418>

## HealthVault Application Manager

The purpose of HealthVault App Manager is to quickly create a template application that will access HealthVault. It also has a direct link to Application Configuration Center which is another step required to get your HealthVault app up and running.

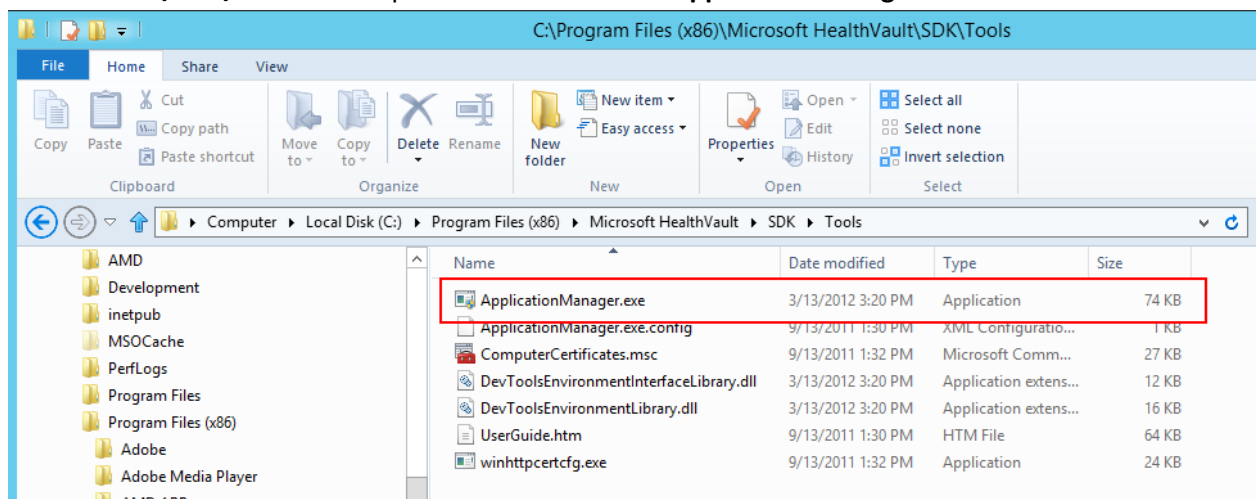
One of the drawbacks of the App Manager is it creates a ASP.NET WebForms template. You can use WebForms, but we will be converting it to ASP.NET MVC 4 web application instead of using WebForms.

## Creating Your App

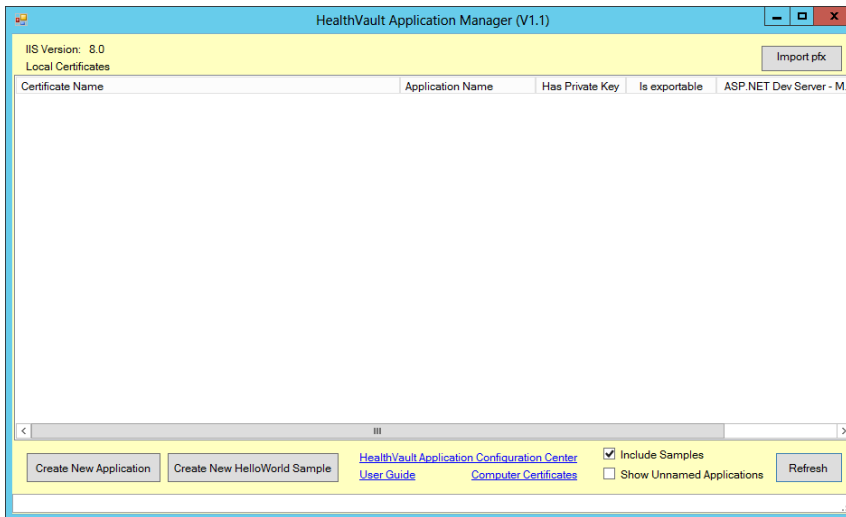
HealthVault App Manager will create a 'starter' app to get you up and running and will also create a certificate for you that will be required to upload to HealthVault PPE Server.

To create your HealthVault App follow these steps

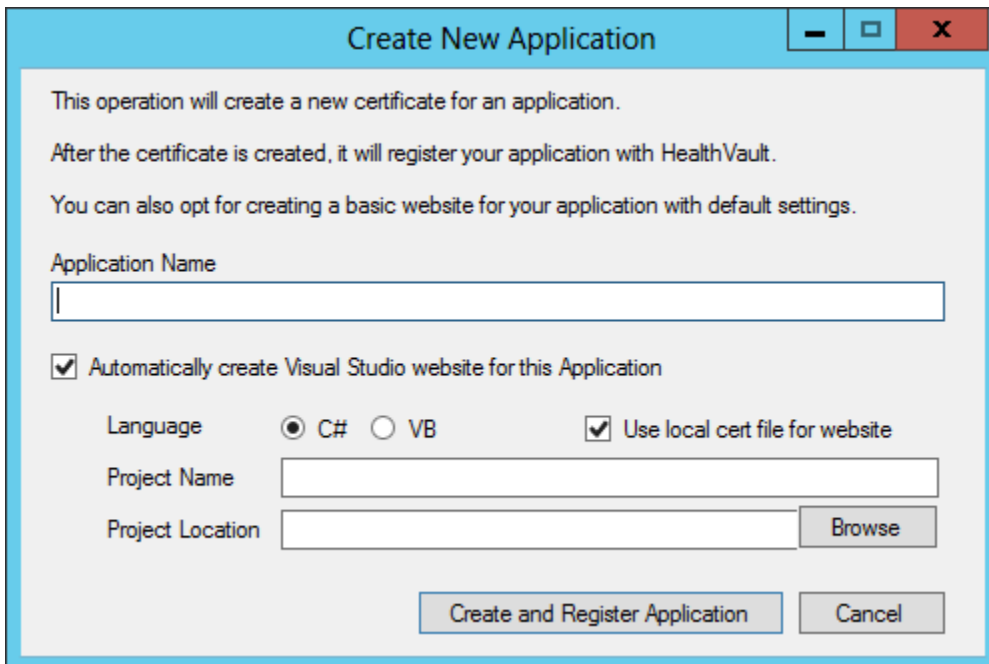
1. Once the installation is complete, open the following directory **%program files%\Microsoft HealthVault\SDK\Tools** in an explorer window and run **ApplicationManager.exe**



2. **HealthVault Application Manager v1.1** will start up which will leave you with a screen as follows



3. Click on **Create New Application**
4. A **Create New Application** dialog will appear as follows



5. Type in the name of your application, select the language and make sure **Use Local cert file for website** is checked
6. [Optional] Change the **Project Name** if required
7. Select a **Project Location** to save the files to
8. Click on **Create and Register App**. This will create a Visual Studio 2010 project and start off a browser session so you can register the application.

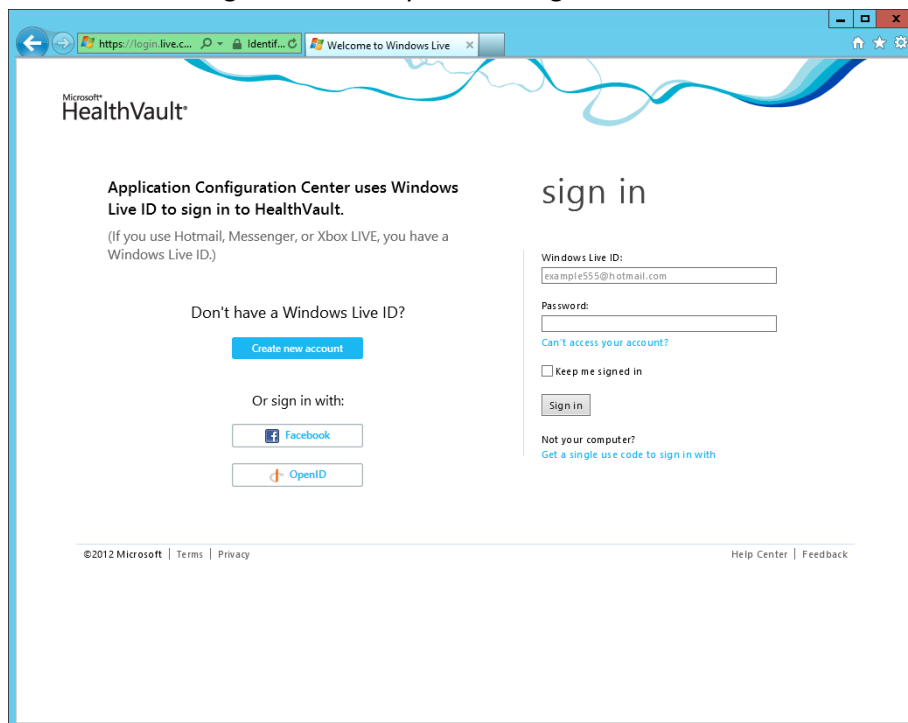
Next we will continue the setup of the application and register the application with HealthVault and walkthrough the Visual Studio project generated

## Registering your App with HealthVault

When we register our application with HealthVault, we are essentially giving HealthVault servers our certificate generated in the previous section so it can authenticate our requests. The certificate is used by the HealthVault SDK to encrypt any requests to the HealthVault server, and the HealthVault server can decrypt the data.

The following is a walkthrough of registering your app with HealthVault PPE environment

1. In the browser session that was opened in **Step 6** of creating your app enter your Live Id Credentials. If you do not have one, create a new one. You can also use OpenId or Facebook, but the walkthrough will assume you are using LiveId



2. Once signed in you will be taken to a 'Terms and Conditions' page for HealthVault which you must accept to create the application. Check the **I accept the terms and conditions** then click

the **Create application** button.

The screenshot shows a web browser window with the address bar displaying 'https://config.h.' and a tab titled 'Create Application'. The page header for 'Microsoft HealthVault Application Configuration Center' includes a welcome message for 'RedBit Test Account' and links for 'Sign out' and 'Help'. The main content area is titled 'Please read and accept the terms and conditions to create a new application.' and contains the 'Microsoft HealthVault Developer Site Terms of Use'. The terms are dated 'May, 2008' and include sections for 'Acceptance of Terms' and 'Description of Services'. At the bottom of the terms, there is a checkbox for 'I accept the terms and conditions' and a 'Create application' button. The footer contains copyright information for Microsoft and links to 'Privacy', 'Legal', 'Help', and 'Feedback'.

Microsoft HealthVault  
Application Configuration Center

Welcome, RedBit Test Account | [Sign out](#) | [Help](#)

Please read and accept the terms and conditions to create a new application.

## Microsoft HealthVault Developer Site Terms of Use

Updated: May, 2008

- Acceptance of Terms**

This is a contract between you and Microsoft Corporation. Your use of the Microsoft HealthVault Developer Site and related application programming interfaces ("service") is subject to the following terms ("agreement"). Microsoft reserves the right to update this agreement at any time on notice as described in section 18. Please note that we do not provide warranties for the service. The contract also limits our liability. These terms are in sections 11 and 12 and we ask you to read them carefully.
- Description of Services**

This service is offered for the sole purpose of designing, developing and testing your applications for compatibility with Microsoft HealthVault as described in the HealthVault SDK and any related resources Microsoft provides. Except for your developer account email address and Windows Live ID, you may not use, save or store any personally identifiable information or personal health

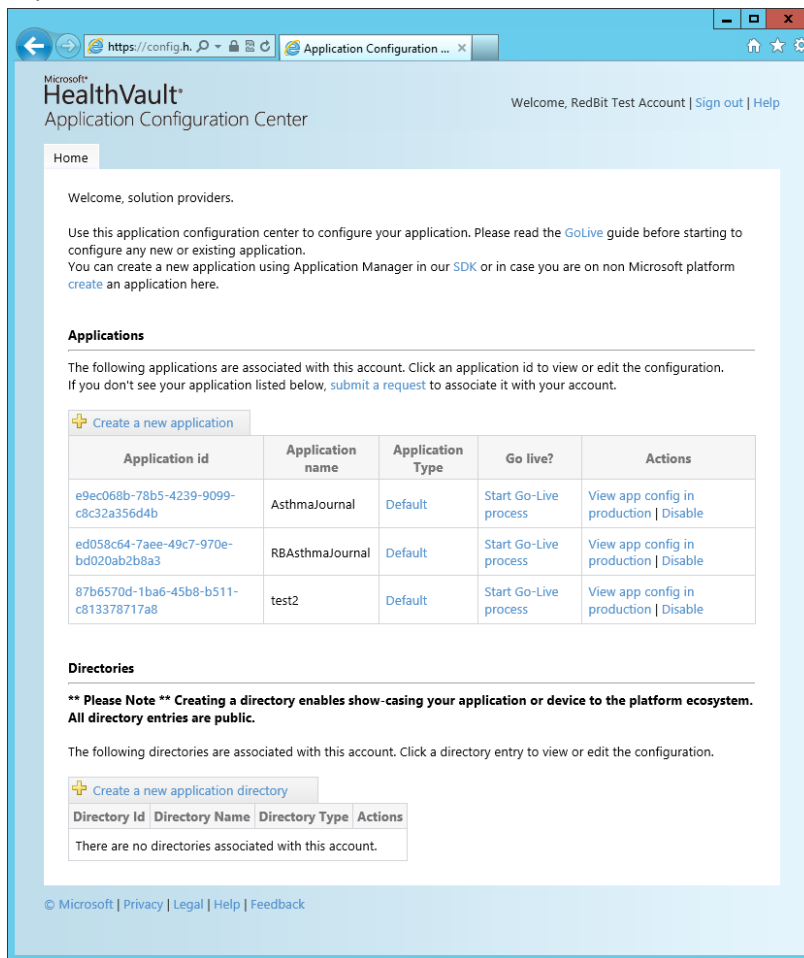
☐ I accept the terms and conditions

[Create application](#)

© Microsoft | [Privacy](#) | [Legal](#) | [Help](#) | [Feedback](#)

- The Application Configuration Center page will be shown with your newly created app displayed.  
NOTE: You can also manually create the app and upload cert but I will leave that to the reader to

explore.



## Configuring HealthVault App Rules

The HealthVault app requires rules for online and offline data access. In the next steps we will be configuring **Online Rules**

1. Click on your newly created app. This will take you into the configuration page.
2. Click on the **Online Rules** tab and click **Edit** for rule named **default name**.
3. [Optional] Change the **Rule Name**

### Edit online auth rule

Rule name

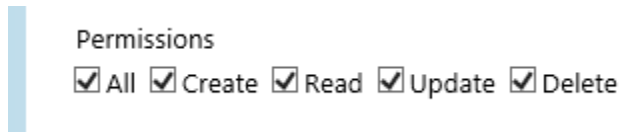
default name

4. [Optional] Set the **Why String**

Why string

To submit asthma journal entries to HealthVault and access account features such as images.

5. For **Permissions** select All



Permissions

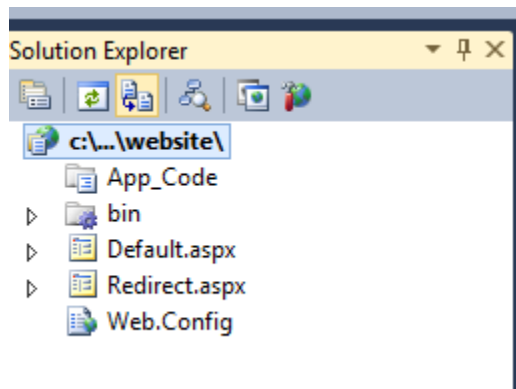
☒ All ☒ Create ☒ Read ☒ Update ☒ Delete

6. For Asthma Journal we will need the following data types. Select the ones you need for your specific purposes.
  - a. Application Data Reference
  - b. Application-Specific Information
  - c. Basic Demographic Information
  - d. Personal Demographic Information
  - e. Personal Image
7. [Optional] The following are option when in development but should be set for production
  - a. **Information Tab** - set the various values as required
  - b. **Localize Tab** – set more application specific information

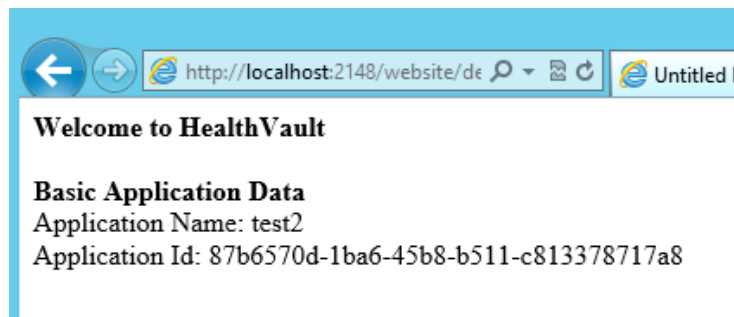
## HealthVault Sample Application

Now that our HealthVault app is setup with the HealthVault pre-production environment, we can finally look at the source code that was generated by the HealthVault Application Manager Utility.

When you created the app with HealthVault Application Manager, Visual Studio 2010 instance was launched with the project loaded. The project contains minimal files



Running the project, will require you to log into HealthVault with your Live Id and will display some of the app properties from HealthVault.



The following sections will cover some key items to be aware of in the project and integrating with HealthVault

## Web.Config Settings

Web.Config contains various settings that the HealthVault SDK uses to allow you to communicate with the HealthVault servers.

```
<appSettings>
  <add key="ApplicationId" value="87b6570d-1ba6-45b8-b511-c813378717a8" />
  <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="Redirect.aspx" />
  <!-- 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="c:\temp\test2\cert\WildcatApp-87b6570d-1ba6-45b8-b511-c813378717a8.pfx" />
</appSettings>
```

The following are a brief explanation of the keys

1. **ApplicationId** – The GUID that was setup in the Application Configuration Center
2. **ShellUrl** – Defines the location where the user will log in. Needs to be changed when going production
3. **HealthServiceUrl** – Defines the location where all requests to HealthVault will go through. Needs to be changed when going production
4. **NonProductionActionUrlRedirectOverride** – page to re-direct to in non-production environments
  - a. **WCPage\_ActionHome, WCPage\_ActionAppAuthSuccess, WCPage\_ActionSignOut** – variables used for redirection depending on the response from the shell url request (usually a login request)
5. **ApplicationCertificateFileName** – the name of the file to use to encrypt the requests to HealthVault. This should be the same file that was uploaded to HealthVault when the app was created with the HealthVault Application Configuration Manager

## HealthVaultServicePage Class

When using ASP.NET Webforms, you can have your pages inherit from Microsoft.Health.Web.HealthVaultServicePage instead of System.Web.UI.Page. The HealthVaultServicePage handles a lot of the communication handling with HealthVault via the SDK.

## Code Walkthrough

The code in the sample application is very straight forward and just shows the application name and the application ID from HealthVault. Here is the relevant HTML code

```
<b>Welcome to HealthVault</b> <br /><br />

<b>Basic Application Data</b> <br />
<asp:Label ID="AppName" runat="server" Text="Application Name: " /><br />
<asp:Label ID="AppId" runat="server" Text="Application Id: " /><br />
```



And the C# codebehind is as follows

```
ApplicationInfo info = ApplicationConnection.GetApplicationInfo();
AppName.Text += info.Name;
AppId.Text += info.Id.ToString();
```

The HealthServicePage.ApplicationConnection object will be one of the main properties you use to get information in and out of HealthVault which we will look at in the next section. In the code sample above, we are just getting the app name and the app Id that was setup in HealthVault.

Running the web app produces the following results



### Adding Some Code

To make the page a bit more interesting , we'll go ahead and add some code to accomplish the following

1. Display the signed in person's name
2. Display the signed in person's birth year

### Get the Person Details

Add the following to Default.aspx under the AppId asp:label

```
<asp:Label ID="name" runat="server" Text="Name: " /><br />
<asp:Label ID="bday" runat="server" Text="DOB: " /><br />
```

Add the following C# method to Default.aspx.cs

```
/// <summary>
/// Generic method to get data from healthvault depending on the TypeId
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="typeID"></param>
/// <returns></returns>
T GetSingleValue<T>(Guid typeID) where T : class
{
    // create a searcher to get data from healthvault
    HealthRecordSearcher searcher = PersonInfo.SelectedRecord.CreateSearcher();

    // create a filter to add to the search
    HealthRecordFilter filter = new HealthRecordFilter(typeID);
```

```

searcher.Filters.Add(filter);

// make the request to find the data
HealthRecordItemCollection items = searcher.GetMatchingItems()[0];

// return the data if available
if (items != null && items.Count > 0)
{
    return items[0] as T;
}
else
{
    return null;
}
}

```

The code is commented so will not walk through line by line. In a nutshell, this method will allow us to get a value from HealthVault depending on the typeId being passed. More on typeId in the next part, but first add the following under the line where AppId gets set in Default.aspx.cs

```

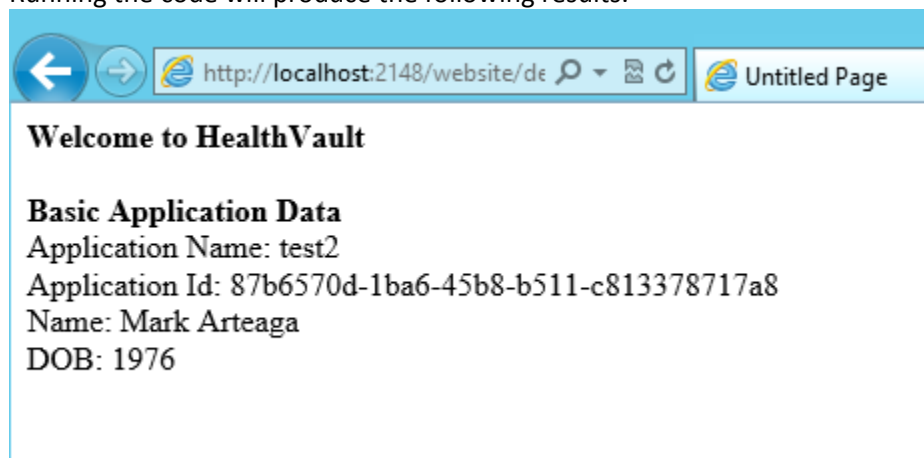
// set the persons name
name.Text += PersonInfo.Name;

// attempt to get the birthdate of the user
Basic basic = GetSingleValue<Basic>(Basic.TypeId);
if (basic != null && basic.BirthYear.HasValue)
    bday.Text += (basic.BirthYear.ToString());

```

Again, code is commented so I'll only focus on the GetSingleValue method as promised. You will notice that we pass a Basic.TypeId which is a GUID. HealthVault defines some standard data types that you can use to get or add data to like [Allergy](#) or [BloodPressure](#). [Basic](#) type, contains the information of the person signed in such as Birthdate and 'is not considered personally identifiable information' according to the MSDN Documentation. Every [HealthVault.ItemTypes](#) contains a GUID that identifies the type in the system, and this GUID is used to perform the search to retrieve the data.

Running the code will produce the following results.



## HealthVault Source Code

Microsoft HealthVault SDK does come with source code and can be found under %ProgramFiles%\Microsoft HealthVault\SDK\Source\HealthVaultDLLSource.zip.

In the development of Asthma Journal using MVC, having the source code was instrumental in integrating HealthVault with ASP.NET MVC. We will cover that topic in more detail in the next article.

## Conclusion

In this article, we covered getting started developing with HealthVault and using HealthVault Application Manager to get a sample application up and running. We also walked through setting up access rights for the application in HealthVault Application Configuration Center portal to allow us access to the appropriate data within HealthVault. In the next article, we'll cover getting HealthVault SDK integrated with an ASP.NET MVC web application.