Integrating Windows 8 WinJS Metro App with HealthVault Part II

Author: Mark Arteaga
Date: July 31, 2012

In this second part of this article we will look at how to access HealthVault data from a Windows 8 application using WinJS (HTML/CSS/Javascript). We will continue from Part I were we setup the initial infrastructure to have the Win8 app connect

We will be taking key learning's from Asthma Journal available for download on GitHub.

Saving HealthVault Users

First thing we are going to have to do is save a list of HealthVault users accessing the system and save the authorization token.

1. In your models directory, create a new class called **HealthVaultUser.cs** and implement as follows

```
/// <summary>
/// Represents a healthvault user
/// </summary>
public class HealthVaultUser
{
    /// <summary>
    /// The Id for the record
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// The user id associated with this healthvault record
    /// </summary>
    public Guid UserId { get; set; }

    /// <summary>
    /// The name of the healthvault user since the username in asp.net membership is a
guid
    /// </summary>
    public string Name { get; set; }

    /// <summary>
    /// The HealthVault person id
    /// </summary>
    public Guid PersonId { get; set; }

    /// <summary>
    /// The record id for the health person record, not the same as personId
    /// </summary>
    public Guid RecordId { get; set; }

    /// <summary>
    /// The HealthVault Selected Record ID
```

```
    /// </summary>
    public Guid AuthorizedRecordId { get; set; }

    /// <summary>
    /// The serialized person info object
    /// </summary>
    public string PersonInfoObject { get; set; }

    /// <summary>
    /// The token returned by health vault.
    /// </summary>
    public string WCToken { get; set; }

    /// <summary>
    /// Determins whether there is access to the record
    /// </summary>
    public string HealthRecordState { get; set; }

}
```

2. Open up **HVDbContext.cs** and add the following property

```
/// <summary>
/// HealthVault users
/// </summary>
public DbSet<HealthVaultUser> HealthVaultUsers { get; set; }
```

3. In the Controllers directory, open **AccountController.cs** and locate the **Login** method
4. Add the following line before the `HVUserImageHelper.Default.SaveImageToBlobStorage` call

```
// save the user to the local table
SaveUser(personInfo, authToken);
```

5. Implement the SaveUser method as follows

```
private void SaveUser(PersonInfo personInfo, string token)
{
    // create a new context
    var context = new HVDbContext();

    // check if the user already exists
    var user = (from t in context.HealthVaultUsers
                where t.PersonId.ToString().Equals(personInfo.PersonId.ToString())
                select t).FirstOrDefault();

    if (user == null)
    {
        // add user to collection
        context.HealthVaultUsers.Add(new HealthVaultUser()
        {
            HealthRecordState = personInfo.SelectedRecord.State.ToString(),
            Name = personInfo.SelectedRecord.Name,
            PersonId = personInfo.PersonId,
            WCToken = token,
            PersonInfoObject = personInfo.GetXml()
        });
```

```
    }
    else
    {
        // update the user
        user.HealthRecordState = personInfo.SelectedRecord.State.ToString();
        user.Name = personInfo.SelectedRecord.Name;
        user.PersonId = personInfo.PersonId;
        user.WCToken = token;
        user.PersonInfoObject = personInfo.GetXml();
    }

    // save the record
    context.SaveChanges();
}
```

Updating the database

We will need to update our database in Sql Azure and running the following query will add the appropriate table

```
CREATE TABLE [dbo].[HealthVaultUsers]
(
       [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(MAX) NOT NULL,
    [PersonId] UNIQUEIDENTIFIER NOT NULL,
       [RecordId] UNIQUEIDENTIFIER NOT NULL,
    [PersonInfoObject] NVARCHAR(MAX) NOT NULL,
    [WCToken] NVARCHAR(MAX) NOT NULL,
    [HealthRecordState] NVARCHAR(50) NOT NULL
)
```

Once you have added the table, compile and run the project. Attempt to sign in and a record will be created in the table.

Note, when saving the WCToken, you should encrypt this value somehow. In production, I would not keep this token in a database unencrypted.

Building Services

Now that we have our main MVC App updated, we can build out the APIs to return the appropriate data. The next few sections will cover building these services up.

Sending List of Users

First method we want to create in our API is a method to send back all the users registered in the system. Open up **Areas\v1\Controllers\DoctorAccountController.cs** and add the following method

```
/// <summary>
/// Gets a list of users in the system
/// </summary>
/// <returns></returns>
[AuthorizeRole(Roles="Doctor")]
public ActionResult GetUserList()
{
    var ret = new { status = "ok" };
```

```
    // get a list of users
    var context = new HVDbContext();
    var users = (from t in context.HealthVaultUsers
                 select new
                 {
                     t.Id,
                     t.RecordId,
                     t.Name,
                 }).ToList();

    // compose the response
    return Json(new
    {
        status = ret.status,
        users = users.Select(a => new { a.Id, a.Name, imageUrl =
HVUserImageHelper.Default.GetImageUrl(a.RecordId) }).ToList(),
    }, JsonRequestBehavior.AllowGet);
}
```

Run the application, navigate to http://localhost/v1/doctoraccount/getuserlist and you should get the following


```
{
   status: "ok",
  - users: [
     - {
          Id: 1,
          Name: "Mark Arteaga",
          imageUrl: "http://sampleshvmvc.blob.core.windows.net/userimages/8043c301-23ab-4a4c-9633-8cca1d437087.jpg"
       }
   ]
}
```


Sending User Data

Now that we can get a list of users, we can go ahead and create our method to return data about the user. Add the following method to the **DoctorAccountController.cs**

```
[HandleError]
public ActionResult GetUserData(int userId = -1)
{

    // just do a basic check
    if (userId == -1)
        return Json(new { status = "error", msg = "userId not sent" },
JsonRequestBehavior.AllowGet);

    // try to find the user
    var context = new HVDbContext();
    var user = (from t in context.HealthVaultUsers
                where t.Id == userId
                select t).FirstOrDefault();

    // if no user is found return error
    if (user == null)
        return Json(new { status = "error", msg = "userId not found" },
JsonRequestBehavior.AllowGet);
```

```
    // extract the token and make the request to health vault for all the data
    var authToken = user.WCToken;

    // register the type in the HV SDK
    ItemTypeManager.RegisterTypeHandler(HVJournalEntry.TypeId, typeof(HVJournalEntry),
true);


    // create the appropriate objects for health vault
    var appId = HealthApplicationConfiguration.Current.ApplicationId;
    WebApplicationCredential cred = new WebApplicationCredential(
        appId,
        authToken,
        HealthApplicationConfiguration.Current.ApplicationCertificate);

    // setup the user
    WebApplicationConnection connection = new WebApplicationConnection(appId, cred);
    PersonInfo personInfo = null;
    try
    {
        personInfo = HealthVaultPlatform.GetPersonInfo(connection);
    }
    catch
    {
        return Json(new { status = "error", msg = "Unable to connect to HealthVault
service" }, JsonRequestBehavior.AllowGet);
    }

    // get the selected record
    var authRecord = personInfo.SelectedRecord;

    // make sure there is a record returned
    if (authRecord == null)
        return Json(new { status = "error", msg = "cannot get selected record" },
JsonRequestBehavior.AllowGet);

    // before we add make sure we still have permission to read
    var result = authRecord.QueryPermissionsByTypes(new List<Guid>() {
HVJournalEntry.TypeId }).FirstOrDefault();
    if (!result.Value.OnlineAccessPermissions.HasFlag(HealthRecordItemPermissions.Read))
        return Json(new { status = "error", msg = "unable to create record as no
permission is given from health vault" }, JsonRequestBehavior.AllowGet);

    // search hv for the records
    HealthRecordSearcher searcher = authRecord.CreateSearcher();
    HealthRecordFilter filter = new HealthRecordFilter(HVJournalEntry.TypeId);
    searcher.Filters.Add(filter);
    HealthRecordItemCollection entries = searcher.GetMatchingItems()[0];
    var ret = entries.Cast<HVJournalEntry>().ToList().Select(t => t.JournalEntry);

    return Json(ret, JsonRequestBehavior.AllowGet);
}
```

Run the application, navigate to http://localhost/v1/doctoraccount/getuserdata?userid=1 and you should get the following if you have submitted data

```
[
  - {
        Id: 0,
        UserId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        RecordId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        MissedWorkSchool: false,
        SawDoctor: false,
        SawEmergency: false,
        PeakFlowReading: 0,
        IsDay: false,
        Latitude: -1,
        Longitude: -1,
        Created: "/Date(1343364356000)/",
        EntryDate: "/Date(1343361600000)/",
        HvId: null
    },
  - {
        Id: 0,
        UserId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        RecordId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        MissedWorkSchool: false,
        SawDoctor: false,
        SawEmergency: false,
        PeakFlowReading: 0,
        IsDay: false,
        Latitude: -1,
        Longitude: -1,
        Created: "/Date(1343151478000)/",
        EntryDate: "/Date(1343102400000)/",
        HvId: null
    },
  - {
        Id: 0,
        UserId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        RecordId: "19b2b02e-c31d-4a23-ac3f-fc64504a6c53",
        MissedWorkSchool: false,
        SawDoctor: false,
        SawEmergency: false,
        PeakFlowReading: 0,
        IsDay: false,
        Latitude: -1,
        Longitude: -1,
        Created: "/Date(1343151469000)/",
        EntryDate: "/Date(1343102400000)/",
        HvId: null
```

Signing out Users

For completeness, we will want user to have the ability to sign out so add the following method in your **DoctorAccountController.cs**

```
/// <summary>
/// Signs out the currently signed in user
/// </summary>
/// <returns></returns>
public ActionResult SignOut()
{
    FormsAuthentication.SignOut();
    return Json(new { status = "ok" }, JsonRequestBehavior.AllowGet);
}
```

Accessing Services from Win8

Now that our services are ready, we can go ahead and update our WinJS app to retrieve this data after a user has logged in. Our app will be very simple and just display a list of users, and when a user is clicked, show the data for that user.

Updating LoginManager

To support our new features, we will need to add some extra functionality to our app, primarily in the signing in and out process.

1. In your loginManager.js file, add the following two lines which will be method handlers for when signout is successful or fails

```
// method to call when signout occurs
onSignOut: null,

// method to call if signout fails for some reason
onSignOutFailed: null,
```

2. Next add the following function which will essentially call our **SignOut** method in our **DOctorAccountController**

```
// signs out the current session
signOut: function () {
    // make a request
    WinJS.xhr({
        url: this._baseUrl + 'SignOut',
    }).then(
    function (result) {
        if (result.status === 200) {
            // we are good see if the status code is ok
            var res = JSON.parse(result.responseText);
            if (res.status === 'ok') {
                // we are ok so callback
                if (self.onSignOut)
                    self.onSignOut(result);
            }
            else {
                // tell the user we can't signout
```

```
                if (self.onSignOutFailed)
                    self.onSignOutFailed(result);
            }
        }
        else {
            // there was a wrong reposne from server
            if (self.onSignOutFailed)
                self.onSignOutFailed(result);
        }
    },
    function (result) {
        // there was an error so report back to user
        if (self.onSignOutFailed)
            self.onSignOutFailed(result);
    });
}
```

Getting List of Users and Data

When the user is signed in, we want to show a list of users that have registered with the system. To facilitate accessing data, we will create a new file called **hvData.js** to access all data from the server.

In the **js** folder create a new javascript file called **hvData.js** and implement it as follows

```
(function () {
    "use strict";

    var self;
    WinJS.Namespace.define("Application", {
        HealthVaultData: WinJS.Class.define(
            function HealthVaultData() {
                // save a ref
                self = this;

                // store this object globally
                Application.healthVaultData = this;
            },
            {
                _baseUrl: 'http://localhost:10190/api/v1/DoctorAccount/',

                // Gets a list of users in the system
                getUsers: function (success, fail) {
                    WinJS.xhr({
                        url: this._baseUrl + 'getuserlist'
                    }).then(
                        function (result) {
                            try {
                                var res = JSON.parse(result.responseText);
                                if (res.status === 'ok') {
                                    // we are ok
                                    if (success)
                                        success(res.users);
                                }
                                else {
                                    // we are not user id must not exist
                                    if (fail)
                                        fail(result);
```

```
                        }
                    }
                    catch (e) {
                        // just assume there is no cookie saved
                        if (fail)
                            fail()
                    }

                },
                function (result) {
                    // there was an error so let the caller know
                    if (fail)
                        fail(result);
                });
        },

        // Gets a list of a users entries
        getUserEntries: function (userId, success, fail) {
            WinJS.xhr({
                url: this._baseUrl + 'getuserdata?userId=' + userId
            }).then(
                function (result) {
                    try {
                        var res = JSON.parse(result.responseText);
                        if (res.status === 'ok') {
                            // we are ok
                            if (success)
                                success(res);
                        }
                        else {
                            // we are not user id must not exist
                            if (fail)
                                fail(result);
                        }
                    }
                    catch (e) {
                        // just assume there is no cookie saved
                        if (fail)
                            fail()
                    }

                },
                function (result) {
                    // there was an error so let the caller know
                    if (fail)
                        fail(result);
                });
        },
    }
)
});
})();
```

This will essentially call the server and retrieve user data or a user list and return to the caller.

Open up **default.html** and add a reference to this new file as follows

```
<script src="/js/hvData.js"></script>
```

Updating our UI

We will be showing the list of users in a ListView and databinding to that view. We will not go in detail on this but if you want more information see QuickStart: Adding a ListView (Metro style apps using Javascript and HTML). First we will need to update our **home.html** to add the extra sign in/out functionality we added in the previous section

1. Open up **home.html** and add the following in the `section` tag

```
<button>Sign In</button>
<div id="userList" data-win-control="WinJS.UI.ListView"></div>
```

2. Open **home.js** and replace the **ready function** with the following implementation. This new implementation basically adds functionality to change the UI depending on if the user is signed in or not.

```
var self = this;
hvData = new Application.HealthVaultData();

// handle the on sign out method
Application.loginManager.onSignOut = function () {
    self.status.innerText = "need to sign in";
    self.btnSignOut.innerText = "Sign In";
    userList.winControl.itemDataSource = new WinJS.Binding.List([]).dataSource;
}

// handle not being able to sign out
Application.loginManager.onSignOutFailed = function () {
    // show a message to user and ask if they want to try again
    var msg = new Windows.UI.Popups.MessageDialog("Unable to sign out.");
    msg.commands.append(new Windows.UI.Popups.UICommand("Try again", function (command) {
        Application.loginManager.signOut();
    }));
    msg.commands.append(new Windows.UI.Popups.UICommand("Close"));
    msg.defaultCommandIndex = 0;
    msg.cancelCommandIndex = 1;
    msg.showAsync();
}

// wire up the completed events
Application.loginManager.onLoginComplete = function (result) {
    // we are good so attempt to get data
    console.log('sign in successful');
    self.status.innerText = "sign in successful";
    self.btnSignOut.innerText = "Sign Out";
    self.bindUsers();
};

// wire up the failed event
Application.loginManager.onLoginFailed = function (result) {
    // show a message to user and ask if they want to try again
    var msg = new Windows.UI.Popups.MessageDialog("Unable to sign in.");
    msg.commands.append(new Windows.UI.Popups.UICommand("Try again", function (command) {
```

```
        self.showSettings();
    }));
    msg.commands.append(new Windows.UI.Popups.UICommand("Close"));
    msg.defaultCommandIndex = 0;
    msg.cancelCommandIndex = 1;
    msg.showAsync();
};

// attempt to make the request
Application.loginManager.ping(function () {
    // we are good
    self.status.innerText = "already logged in and cookie set";
    self.btnSignOut.innerText = "Sign Out";
    self.bindUsers();
},
function (result) {
    self.status.innerText = "need to sign in";
    self.btnSignOut.innerText = "Sign In";
    self.showSettings();
});

// wire up the sign out button
self.btnSignOut.addEventListener('click', function () {
    if (this.innerText === "Sign In")
        self.showSettings();
    else
        Application.loginManager.signOut();
});
```

3. Change the **showSettings** method implementation to the following since our log in/out handlers are moved to the ready function

```
WinJS.UI.SettingsFlyout.showSettings("login", "/pages/login/login.html");
```

4. Now add the following helper functions to get elements in the page

```
status: {
    get:function(){ return document.querySelector('section[role=main] p');}
},

btnSignOut: {
    get: function () { return document.querySelector('section[role=main] button'); }
}
```

5. Add the following function which will call hvdata.js to get a list of users

```
// binds the list to the listview
bindUsers: function () {
    var self = this;
    hvData.getUsers(function (users) {
        // we are good so bind
        userList.winControl.itemDataSource = new WinJS.Binding.List(users).dataSource;
    },
    function () {
        // something went wrongshow a message to user and ask if they want to try again
        var msg = new Windows.UI.Popups.MessageDialog("Unable to get user list.");
```

```
        msg.commands.append(new Windows.UI.Popups.UICommand("Try again", function
(command) {
            WinJS.Promise.timeout(300).then(function () {
                self.bindUsers();
            });
        }));
        msg.commands.append(new Windows.UI.Popups.UICommand("Close"));
        msg.defaultCommandIndex = 0;
        msg.cancelCommandIndex = 1;
        msg.showAsync();
    });
},
```

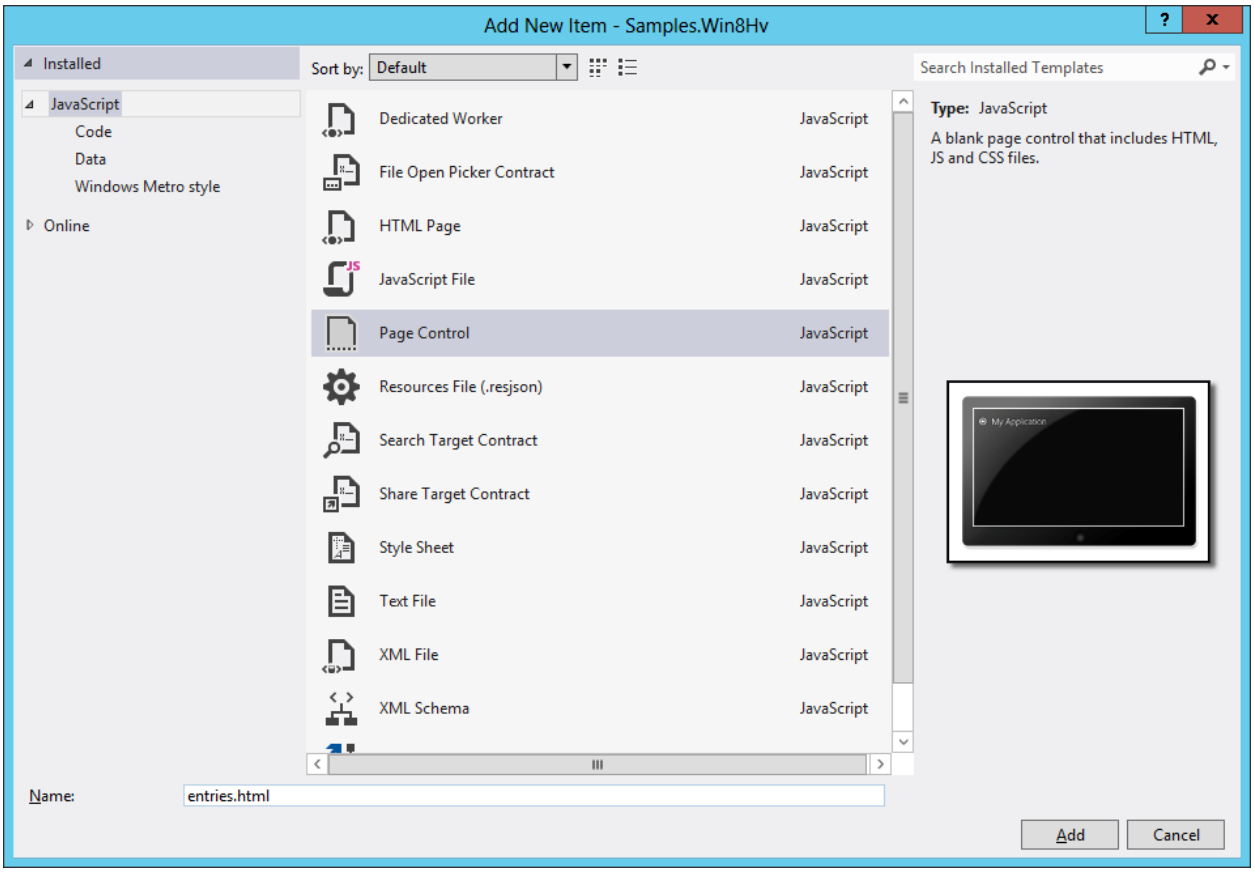
Compile and run the app. If you have not signed in before, you will be shown the login flyout. If you have you will be shown a sign out button. If you have signed in before and your cookie is still active, you will be shown the sign out button. At the appropriate times, the list will be bound to what is returned from the services.

Showing User Entries

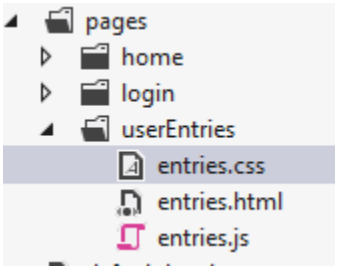
Now that we have a list of users displayed when selected we want to go ahead and display the entries for that user.

1. Right click on **pages** and click **Add -> New Folder**
2. Name the folder **userEntries**
3. Right click on the new folder and click **Add -> New Item**

4. In the **Add New Item** dialog box select **Page Control** and name it **entries.html**



5. Your folder structure should look like the following



Now that we have our structure for showing entries, we need a way to navigate to the page. Open **home.js** and add the following in the **ready** function

```
// wire up the item invoked event
userList.addEventListener('iteminvoked', function (e) {
    var elem = document.getElementById("contenthost");

    // get the item that is selected
    var item = userList.winControl.itemDataSource.list.getItem(e.detail.itemIndex).data;

    // set the item globally so we can grab from entries page
    Application.selectedUser= item;

    // exit the content and when done navigate and enter the content
    WinJS.UI.Animation.exitContent(elem, null).then(function () {
```

```
        // now navigate
        WinJS.Navigation.navigate("/pages/userEntries/entries.html").then(function () {
            WinJS.UI.Animation.enterPage(elem, null);
        });
    });
});
```

Open up **entries.html** and replace the section child node with the following to add ListView

```
<div id="entries"  data-win-control="WinJS.UI.ListView">
```

Open **entries.js** and add the following function

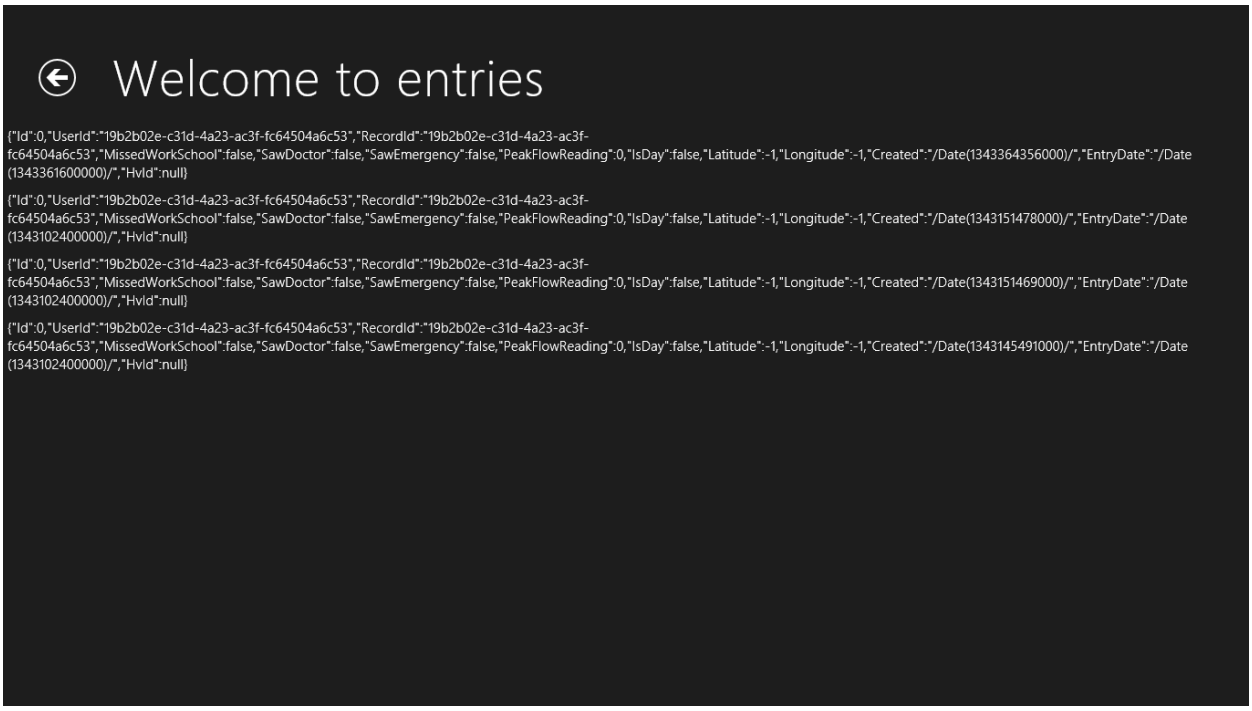
```
// gets the user data from the system
getUserData: function(){
    Application.healthVaultData.getUserEntries(Application.selectedUser.Id, function
(data) {
        // we are good
        entries.winControl.itemDataSource = new WinJS.Binding.List(data).dataSource;
    },
    function () {
        // something went wrongshow a message to user and ask if they want to try again
        var msg = new Windows.UI.Popups.MessageDialog("Unable to get user data.");
        msg.commands.append(new Windows.UI.Popups.UICommand("Try again", function
(command) {
            WinJS.Promise.timeout(300).then(function () {
                self.getUserData();
            });
        }));
        msg.commands.append(new Windows.UI.Popups.UICommand("Close"));
        msg.defaultCommandIndex = 0;
        msg.cancelCommandIndex = 1;
        msg.showAsync();
    });
},
```

In the **ready** function, add the following to call the newly created method

```
this.getUserData();
```

Compile and run the project. When you have successfully signed in, you will get a list of users in the system,

And when you click on a user entry, it will navigate you to the entries page and show the entries associated with that user.



Final Cleanup

We have gone through and successfully retrieved data from the HealthVault server and displayed within an HealthVault app. A few things that are still required that I will leave up to the user to implement are

1. Progress on login/logout – show a progress bar when the user is logging in or out
2. Progress on getting data – show a progress when we are grabbing data from our server to display
3. Styling of data – need to style the user information as well as the entries for the user

Conclusion

In this second part of this article, we looked at updating our MVC application to save HealthVault user information to allow us to access data while the user is not logged in. We also updated the services to return user information and data associated with that user. And finally, we updated the Win8 WinJS application to display users and data for selected users.