*“****Think: mHealth as personal health reform****”* Jane Sarasohn-Kahn

|  |  |
| --- | --- |
| *Concepts to cover*   * Implementing a mobile application   + -- * Things to know to add a mobile interface to web application | *To do* |

Having an accessible and programmable health record sets HealthVault apart. It enables a rich ecosystem of devices, mobile, and web applications. Chapter 3 focused on introducing the HealthVault application programming interface; Chapter 4 gave a good overview of HealthVault data types using a data-intensive “Quantified-Self” application. This chapter takes a closer look at building mobile applications for HealthVault.

We will look at an end-to-end example of building a mood tracking application on top mobile platforms. The chapter will covers element of mobile client programming using the code samples for Windows Phone 7 (C#). Similar interfaces are available for Android (Java) and iOS (Objective-C).

# The Mood Tracker Mobile Application

In Chapter 3, we built an end to end web-application that enables a user to track several kinds of data and use that data to help with self-experimentation. Many elements of self-tracking data, such as sleep, weight, and exercise have the capability to be measured through devices; however, it’s very hard to measure elements of happiness such as mood and stress automatically.

In recent years, we have seen a surge in mobile smartphone devices. Mobile devices offer a very effective tool for efficient data entry. Mobile devices are an ideal platform to build data collection tools. So our manual “mood tracking” need could be served by an application that makes it easy and engaging for a user to track mood using a smart phone. For purposes of our example, let built the application on Windows Phone 7 platform.

So, what should we build?

The application will allow the user to input their mood, well-being, and stress level; present a way to look at the history of the data; and add a bit of zest using an avatar, a mood plant. The mood plant summarizes the user’s emotional state over time. When the user is happy, stress-free, and fit for a long time, the plant thrives, showing a happy face (☺) and in case of depression and stress it shows the effects of bad health (☹).

Figure 5-1 is a sketch of what the app might look like.

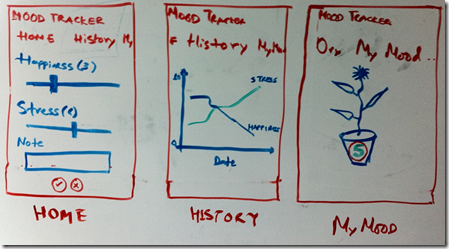
[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image1.png)

Fig 5-1. White board wire-frame of our mood tracking application.

## ****Choosing HealthVault Integration****

**The first question we need to answer is what kind of HealthVault connectivity does this application require?** We discussed several models of connecting with HealthVault in chapter 3. As this application is only for a client device, the model we will be using a client application, and will utilize the HealthVault Windows Phone 7 client library. Having a client application allows you to provide a rich interface and the potential capability to store the readings locally.

## ****Selecting Appropriate HealthVault data types****

**The next question we should solve is what HealthVault data types to use?** We discussed HealthVault data types in detail in chapter 4. Various data types could apply in this context, but browsing the [HealthVault data types](http://developer.healthvault.com/pages/types/types.aspx) reveals one relevant data type in particular: [Emotional State](http://developer.healthvault.com/pages/types/type.aspx?id=4b7971d6-e427-427d-bf2c-2fbcf76606b3).

On further analysis, it turns out that this type is almost perfect for our use. Mood, stress, and well-being are rated on a scale of 1-5. We do a further reading of associations for each of these values, and add appropriate textual elements for each of the values (mood, stress, and well-being).

# Getting Started

I assume you have Visual Studio installed with Window Phone 7 (WP7) tools. If not, you can get them from [here](http://create.msdn.com/en-us/home/getting_started).

Next, go over to [codeplex](http://healthvaultwp7.codeplex.com/) and download the HealthVault library with sample applications.

I extracted the library to my desktop and the folder structure looks like Fig 5-2. HvMobileRegular has the relevant C# code to abstract for working with the HealthVault web service, and HvMobilePhone uses the code in HvMobileRegular to build a library that works with Windows Phone 7 platform. TestRegular directory has a unit-test for HealthVault mobile Windows Phone 7 library. WegithTrackerDemo is a sample application that shows case uses of the library for a Weight Tracking application.

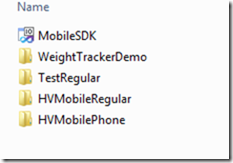
[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image5.png)

Fig 5-2. HealthVault WP7 library extracted.

If you open the MobileSDK solution in visual studio and press F5, the library compiles and the WeightTracker demo starts. Fig 5-3shows this application in action; we will use this application as a template for building ours.

[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image6.png)

Fig 5-3. Compiling and running HealthVault WP7 Sample Application

Without further ado, let’s create our new Silverlight for Windows phone project. We can create a solution for MoodTracker, and reference the HVMobilePhone library in that project. You can also use the existing project, MobileSDK, and associate a new application in it; in the source code associated with this chapter I created a new Project called MoodTracker (Figure 5-4).

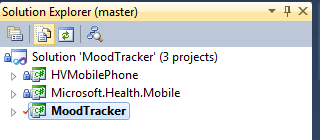


Fig 5-4. Beginnings of Mood Tracker

First things first, let’s set up the application to talk to HealthVault. In the App.xaml.cs class, add a reference to the HealthVault Service and HealthVault Shell. We also need to make sure we get a unique application ID in the developer environment of HealthVault. To do that, we head over to the [HealthVault Application Configuration Center](http://config.healthvault-ppe.com/), and create a new application by clicking on the “Create a new application” button (Figure 5-6). Note that in Chapter 3 we used the Application Manager utility to create a web application, in this chapter we use an alternative method which enables creating client as well as web applications.

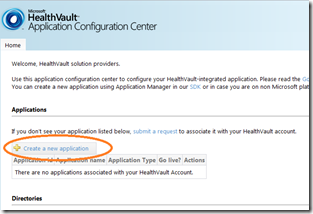
[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image8.png)

Fig5-6. Creating a new application in Application Configuration Center

We create an application of type Software on Device Auth (SODA), which is an authentication mechanism for client applications, and pick the name Mood Tracker for it, as shown in figure 5-7.

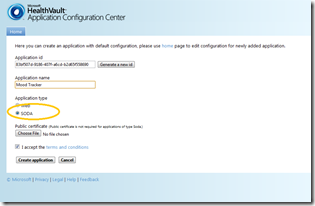
[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image9.png)

Fig 5-7. Creating Mood tracker as a SODA application

Once the application is created, we need to assign appropriate authorization rules for the data types that the application will access. To do that, click on the app’s link and assign appropriate data types for the application, as shown in Figure 5-8.

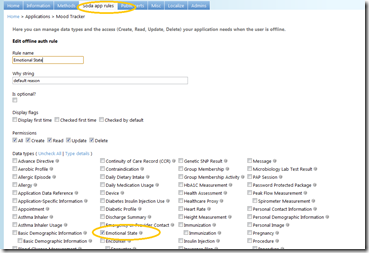
[](http://healthblog.vitraag.com/wp-content/uploads/2011/05/image10.png)

Fig 5-8. Adding Emotional State to our Mood tracker application.

Having created the client application and assigned data type authorization rules, we are all set! Now let’s configure the base page to work with the HealthVault pre-production environment (PPE). The PPE is the development environment publicly available for all HealthVault developers. The HealthVault platform in this environment is available at <https://platform.healthvault-ppe.com/platform/wildcat.ashx> and the HealthVault shell in this environment is available at <https://account.healthvault.com>. Chapter 6 will show how to deploy your app to the general public after you have developed and tested it.

Listing 5-1 shows the initial code for configuring the application. In Line 27, we assign the platformUrl, In Line 28, we assign the shellUrl, and Line 29 is the application identifier that we created using the application configuration center. The *HealthVaultService* object initialized the HealthVault Windows Phone 7 library with appropriate configuration variables. Using this object, we can make all the relevant HealthVault web service requests.

Listing 5-1 configuring the client application

We can make this project a startup project, press F5, and get to the first page of our application. We’re in business!

## Authenticating the Application and User with HealthVault

In order for the Mood Tracker application to work with HealthVault, we will get appropriate application creation credentials from the HealthVault Service. We must also set up a method by which the user can authorize the application using the HealthVault Shell.

1. To get the credentials from the HealthVault Service, the application contacts the HealthVault service to get application creation URL. The code for that is outlined in [MyMood.xaml.cs](https://github.com/vaibhavb/moodtracker/blob/master/MoodTracker/MyMood.xaml.cs) (Listing 5-2).

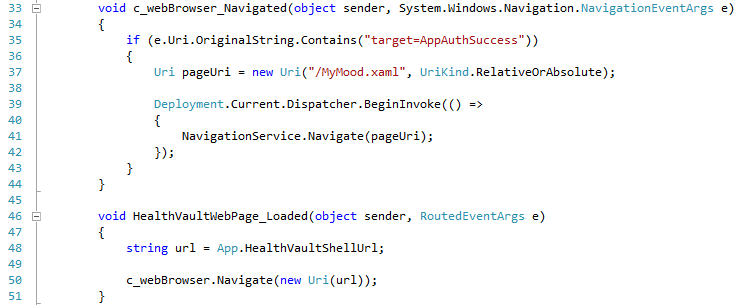


Listing 5-2 Authenticating the application with HealthVault Service

2. The application creation needs to be validated on behalf of the user.

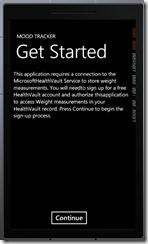
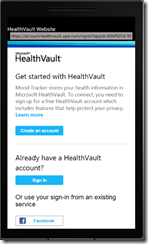
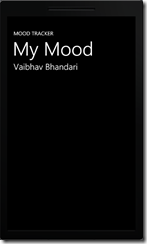
The best mechanism to achieve this is by having a page with a hosted browser that redirects appropriately to HealthVault and then closes the browser and navigates back to the application page after a successful authorization.

Listing 5-3 is the relevant code in HostedBrowserPage.xaml.

Listing 5-3 using a hosted browser to show HealthVault user authentication

Note that on success the Application is being redirected to MyMood.xaml, which is our application’s landing page.

Figure 5-9 shows the flow of how the authentication shown here works.

[](http://healthblog.vitraag.com/wp-content/uploads/2011/06/image.png)[](http://healthblog.vitraag.com/wp-content/uploads/2011/06/image1.png)[](http://healthblog.vitraag.com/wp-content/uploads/2011/06/image2.png)

Step1. Home page      Step2.Sign-in & Auth Step3. Mood tracker in action.

(Note to artist: please include the previous 3 steps as a legend in the figure.

Fig 5-9 Authentication Model with HealthVault

# Reading Data from HealthVault

The data type we settled on for our application was EmotionalState. Our first goal is to be able to read data for this type and display it in our application. To do this, we need test data for emotional state. Add test information in to the test or developer account for this application from [type samples](http://developer.healthvault.com/pages/types/types.aspx), associated with Emotional type in the HealthVault Developer Center, as shown in Fig 5-10. An important thing to note is that you need to be signed in to developer.healthvault.com to add the sample; otherwise this application gives an error.

[image](http://healthblog.vitraag.com/wp-content/uploads/2011/06/image3.png)

Fig 5-10. Adding Emotional State sample to HealthVault record.

We can verify that this sample is added to our record by viewing the information in the HealthVault PPE shell interface (<https://account.healthvault-ppe.com>), as shown in Fig 5-11.

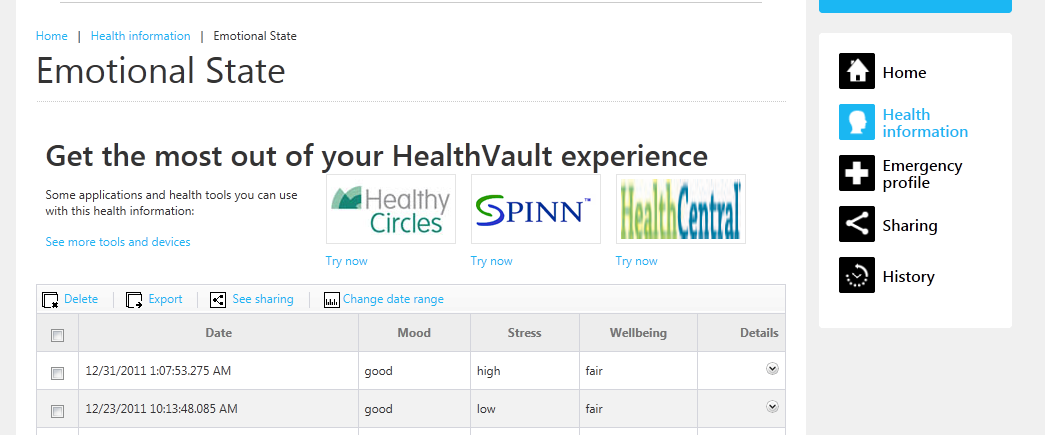


Fig 5-11. Emotional State Samples in developer account.

Chapter 2 explained the HealthVault GetThings method. GetThings enables an application to read data from the user’s health record. A read request for health data can be performed using various querying mechanisms. For the purposes of this application, we will retrieve the last item active for the user’s Emotional State data type.  
 

Listing 5-4 Using GetThings

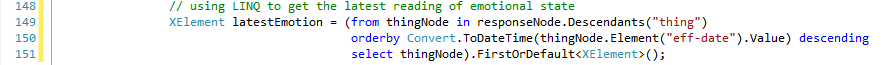
To make it easier to work with GetThings, I implemented a simple abstraction on the method in HealthVaultMethods class. Listing 5-5 shows code for this abstraction. It allows the construction of a GetThings query for one typeid, with maximum items returned and with appropriate minimum and maximum effective dates for these health items. Chapter 4 explains the XML query sent by the GetThings method in detail.



Listing 5-5 GetThings abstraction

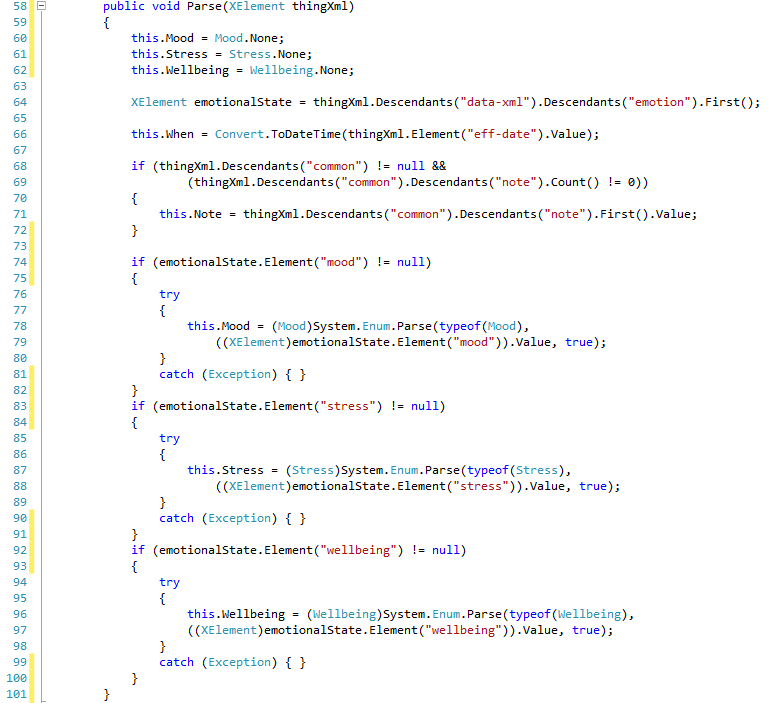
Now, once we can get emotional state things, we need to perform two actions things on the client side.

First, pick the *thing* item we are interested in from the *GetThings* response. To choose the appropriate item, LINQ to XML comes in very handy, offering a SQL-like select clause for XML data as shown in Listing 5-6. LINQ stands for Language Integrated Querying, and it allows for making queries natively from C#.



Listing 5-6 Choosing things from a Get-Thing response

Second, parse the items returned for mood, stress, and well-being data. We can achieve this by creating a model for Emotional State. This model is available to review in the EmotionalStateModel.cs file. The parse method on this model parses the appropriate elements in the *thing* XML. Chapter 4 details the format of this XML. Notice that in Line 68, we parse the common element to fetch the *note* data for emotional state type. In 66, we are setting the When date of the instance to the *eff-date* element. We have created enumerations for Mood, Stress and Wellbeing values, and we can parse the integers for those values using the *Enum.Parse* method.



Listing 5-7 parsing a thing for Emotion State data type

After retrieving the data in our emotional state model, we can use XAML to view it in our application. XAML is the user interface markup technology for Windows Phone 7. For the purposes of this book, we won’t go in detail of XAML. Fig 5-12 shows the display of the latest emotional state reading from HealthVault.

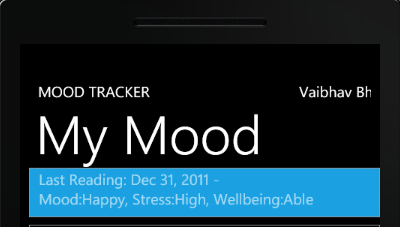


Fig 5-12. Latest Emotional State Reading in Mood Tracker!

# Writing Data to HealthVault

In the previous [section](http://healthblog.vitraag.com/2011/06/adding-a-healthvault-datatype-to-moodtracker-4/) we discussed how one can display the data retrieved from the HealthVault Emotional State data-type. Before we get to the topic of this section and discuss how we can put new items into HealthVault, Fig 5-13 shows a screen shot of how the application looks once we have enabled the put.

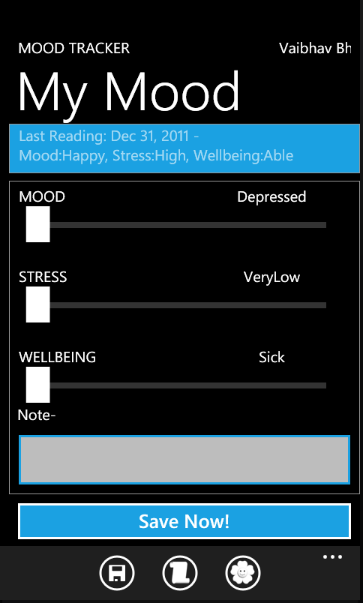
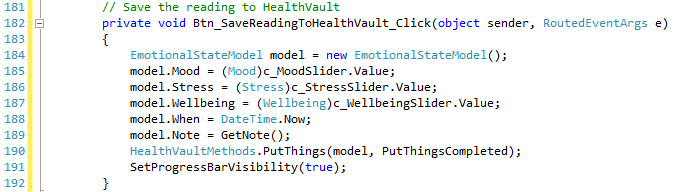


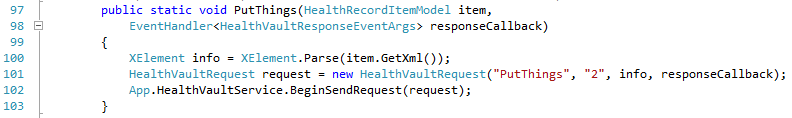
Fig 5-13. MoodTracker with put enabled

For each of the emotional states--mood, stress and well-being--we have a slider that lets the user capture their emotional state. They can also add a note pertaining to their mood using a text box. We want this information to be uploaded with the current time stamp once the user hits Save Now!

Listing 5-6 shows how the save button submits information to HealthVault. Note that in Line 190 we are calling an abstraction for PutThings method.  
 

Listing 5-6 Saving new data to HealthVault

In Chapter 3, we looked at the PutThings method in detail. PutThings enables an application to add or update health items in a user’s record. As Listing 5-7 line 97 shows, our abstraction fetches the relevant information from the base health record item object and submits that to HealthVault using the PutThings version 2 API. The response for this request is handled by the responseCallback function, which in turn can check for various return codes from the service.

  
Listing 5-7 PutThings abstraction

Now that we are able to write data to HealthVault, we have a mobile application which can read and update information to HealthVault!

# Graphing Mood

In the last section we [enabled](http://healthblog.vitraag.com/2011/06/entering-new-data-with-mood-tracker-5/) Mood Tracker to enter new data in to HealthVault.  We want to be able to discover patterns in mood, stress and well-being, and graphing them over time is a great mechanism by which to achieve this goal. Let’s start with a simplistic approach, showing the emotional state readings for mood, stress and well-being over a week. As Fig 5-14 shows, a user can browse mood reading based on a weekly margin and move forward or backward a week at time.

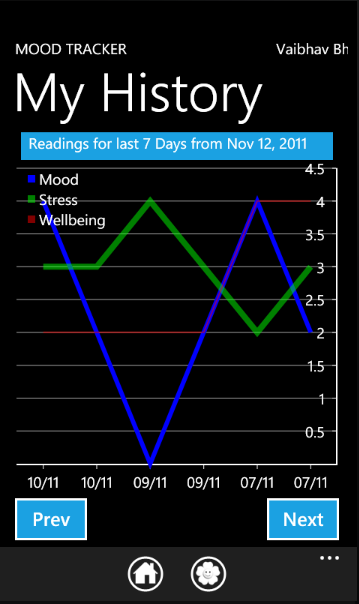
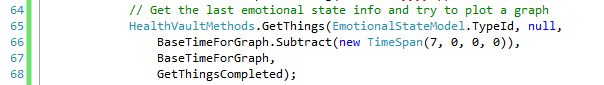


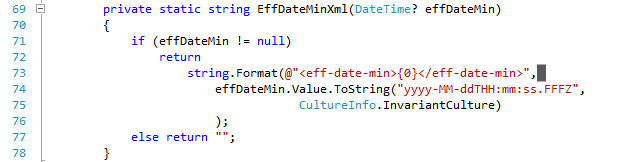
Fig 5-14 Graphing Emotional State over 7 days

In order to be able to get data from HealthVault for a specific time period, the [GetThings](https://github.com/vaibhavb/moodtracker/blob/master/MoodTracker/HealthVaultMethods.cs)[link] method needs to have the effective date filter enabled to look for appropriate readings. Line 66 in Listing 5-8 shows how the GetThings abstraction is configured to return elements for the last seven days only.



Listing 5-8 Fetching readings for last seven days

Note that the eff-date-min element, as implemented in GetThings class in HealthVautlMethods.cs, must be formatted in the ISO 8601 format. Line 73 in Listing 5-9 shows how we do the formatting.



Listing 5-9 Formatting eff-date-min for Get-Things request

Once we can selectively get information from HealthVault, we can use a graphing library to show the readings. In our case, I chose the open-source graphing library amCharts for its ease of use. In fact, I added it to the project with one click using the [NuGet Package manager](http://www.nuget.org/List/Packages/amChartsQuickCharts). Listing 5-10 shows a snippet of the configuration code showing how the graph is set up for mood, stress and well-being using a serial chart. Note that the series values are bound in Line 65 using a DataSource called EmotionList; it is a list of *observable* emotional states.



Listing 5-10 Graphing Emotional State

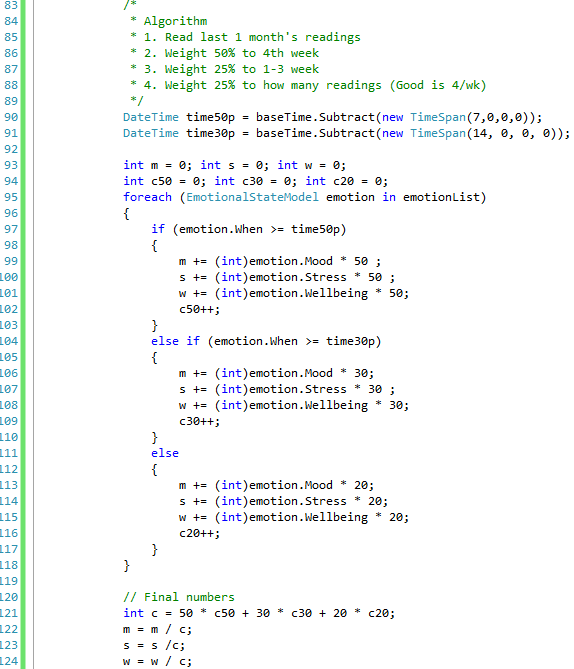
# Data Analysis – Mood Plant

We want a user to engage with the emotional state readings, and a good way to achieve this goal is by providing a zestful visualization for their emotional state. We use a mood plant as a mechanism to gauge a user’s emotional state over recent past.

The flower of the plant represents the average mood, the leaves represent average stress and the roots represent the average wellbeing. Individual flower, leaf and root ligatures map the values 0 through 5 to mood, stress and wellbeing. The final mood plant is a result of super-imposing these values. Fig 5-15 shows an instance of mood plant with mood 3, stress 3, and wellbeing 3.

  
Fig 5-15 Mood Plant

So how do we find average mood, stress, or well-being? Various correlations and algorithms can be used to express the average emotional state over time. We will start with a simple function that creates an average mood, stress, or well-being score based on a weighted average of the values. The function takes the readings for mood, stress, or well-being for the past month. It assigns 50% weight to the more recent week, 30% to the week 3, and 20% to weeks 1 and 2 of the month’s readings. It is left as an exercise to the reader to evaluate and try different functions. Listing 5-11, shows the code for the weighting algorithm

.

Listing 5-11 Algorithm for calculating mood, stress, and well-being over the past month

# What about Android and iOS?

In Chapter 3, we discussed various libraries available for HealthVault. In particular, there are libraries available for Android (Java) and iOS (Objective-C) that allow a developer to implement a mobile application on these platforms. These libraries are open source and commercial friendly.

We don’t show the code for a sample application these platforms, but the functionality available is very similar to the Windows Phone 7 library and the material covered in this chapter will be equally useful.

The Android library is available with the Java SDK on Codeplex [link], and the iOS library is available on GitHub [link]. It is left as an exercise to the reader to create solutions for these platforms.

# Mobile Web Applications

Earlier in the chapter, we discussed the architectural choice to develop a client application for our Mood Tracker. However, web applications do have a choice to use the Web for delivery. In fact, we might want to link the Mood tracker application to our Quantified Self web application so that a user can see other relevant data in a web browser.

It’s important to remember that a web application is a separate application entity from a native client application and that the user has to authorize it separately. The Quantified Self application, when launched from the Mood tracker, will ask for the user to sign in and authorize it.

Web applications can use standard browser detection techniques to present a mobile view of the content. HealthVault Shell does configure its view for mobile applications, and the calling application can always force a mobile view by adding ***mobile*=***true* to the URL parameters. It would be a good exercise for the reader to implement a mobile view of the quantified self-application.

# Contributing to the Mood Tracker Application

The source for MoodTracker windows phone application available at enablingquantifiedself.com, and we are inviting you, dear reader to extend this application and make it your own. Perhaps forks the git repository and contribute your code back or iOS, Android versions of it!