2017 OSIsoft TechCon

Developing Cross-Platform Mobile Apps Using Xamarin and PI Web API

OSIsoft, LLC
1600 Alvarado Street
San Leandro, CA 94577 USA
Tel: (01) 510-297-5800
Web: http://www.osisoft.com

Published: February 9, 2017

Table of Contents

Contents

Introduction

Several years ago, web mobile development was very popular among developers since web sites were visible by many different devices and browsers. However, web sites have some limitations, as they cannot be used offline, they cannot use the device capabilities of a smartphone and they were not easy to monetize. As a result, mobile applications (apps) have become popular. Apps have been able to overcome all the restrictions of the web experience for mobile users. Developing mobile apps can be expensive since each platform has its own native programming language.

Xamarin can reduce costs as it is a framework for cross-platform mobile app development in C#. Since the majority of the PI System developers are already familiar with C#, it is not difficult to get started developing mobile apps using this new technology.

What is Xamarin?

Xamarin is a product that brings .NET/C# to both Android and iOS. Xamarin is fully .NET while being able to produce true Android and iOS apps at the same time, as well as apps that are compliant with the distribution requirements of both Google Play and the iOS App Store.

How does Xamarin work?

The way it works is simple. You write one shared C# codebase with full access to all SDK possibilities and native UI creation mechanisms, and the result you get is a mobile app that looks and feels completely native.

Xamarin is based on an open-source implementation of .NET called Mono. This implementation includes its own C# compiler, development environment and main .NET libraries.

What do I need to get started with Xamarin?

If you are using Windows, Xamarin is an optional feature of the Visual Studio setup. If you are using Mac, you should download and install Xamarin Studio.

NOTE: To develop for iOS on Windows computers there must be a Mac computer accessible on the network, for remote compilation and debugging. This also works if you have Visual Studio running inside a Windows VM on a Mac computer

How do you develop a Xamarin app on top of the PI System?

If you are a PI System developer already familiar with C# and RESTful web services, you probably already know how to make HTTP requests against the PI Web API. As a result, you will find it very easy to create Xamarin apps which will interact with the PI System through PI Web API.

PI Weather Xamarin App

The Xamarin application of this lab is developed using Xamarin Forms, which is a cross-platform UI toolkit that allows developers to easily create native user interface layouts that can be shared across Android, iOS, and Windows Phone.

Since Azure Virtual Machines are used in this lab, it is not possible to run Android, iOS or Windows Phone emulators within this environment. This explains the fact that the app for this lab runs on the Windows platform. Nevertheless, if you download the source code package to your physical machine, you will be able to compile to Android, iOS and Windows Phone using the same codebase.

The **PI Weather Xamarin App** shows weather information of the 50 state capitals of the United States of America. On our first screen, the user selects a city out of the 50 available.

Figure 1 – First screen shows a list with the 50 state capitals of the United States of America

On the second screen, there is an image of the city and a small description from Wikipedia.

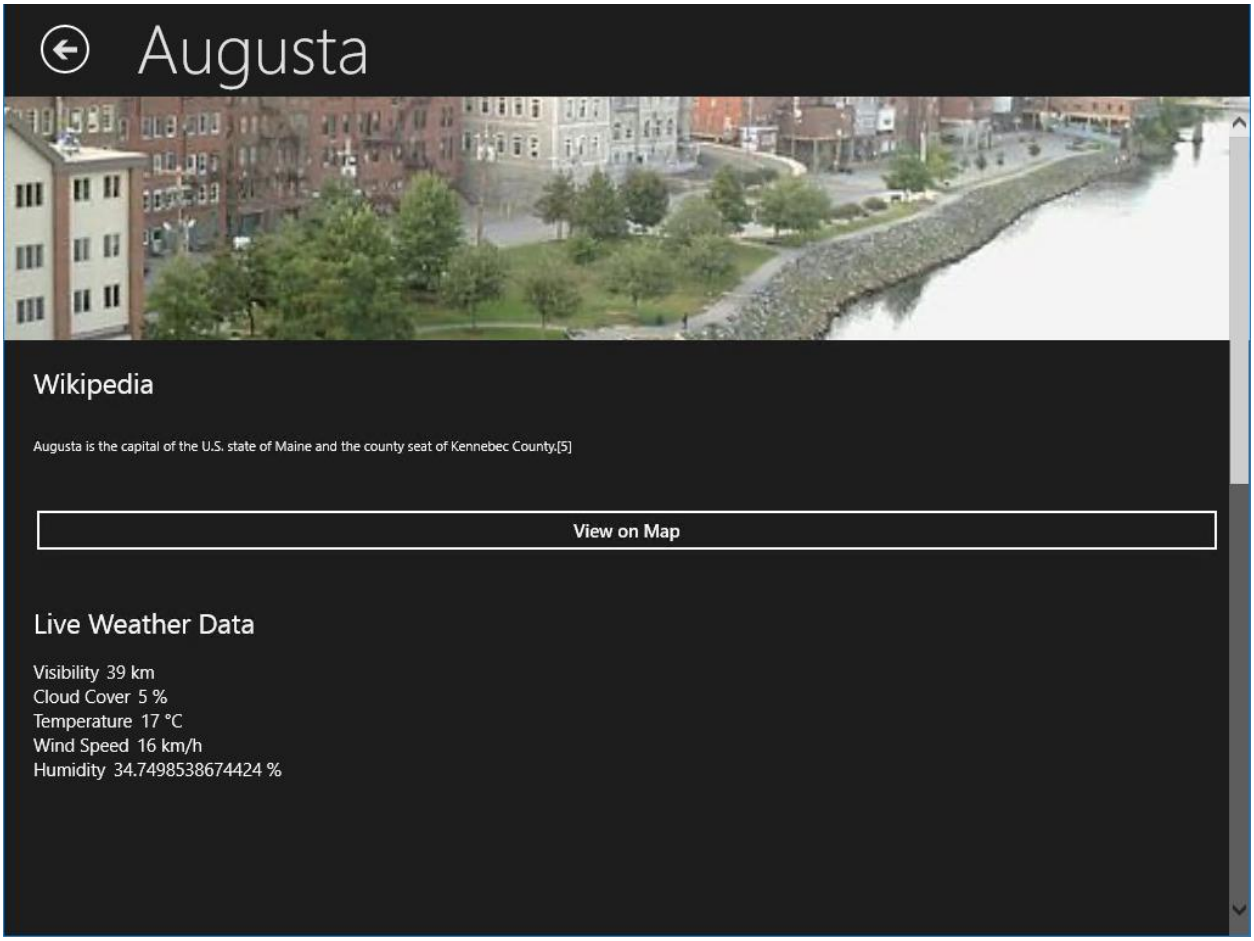

Figure 2 – Second screen shows a photo and the Wikipedia description of the selected city

There is a button to view the location of the city on the map.

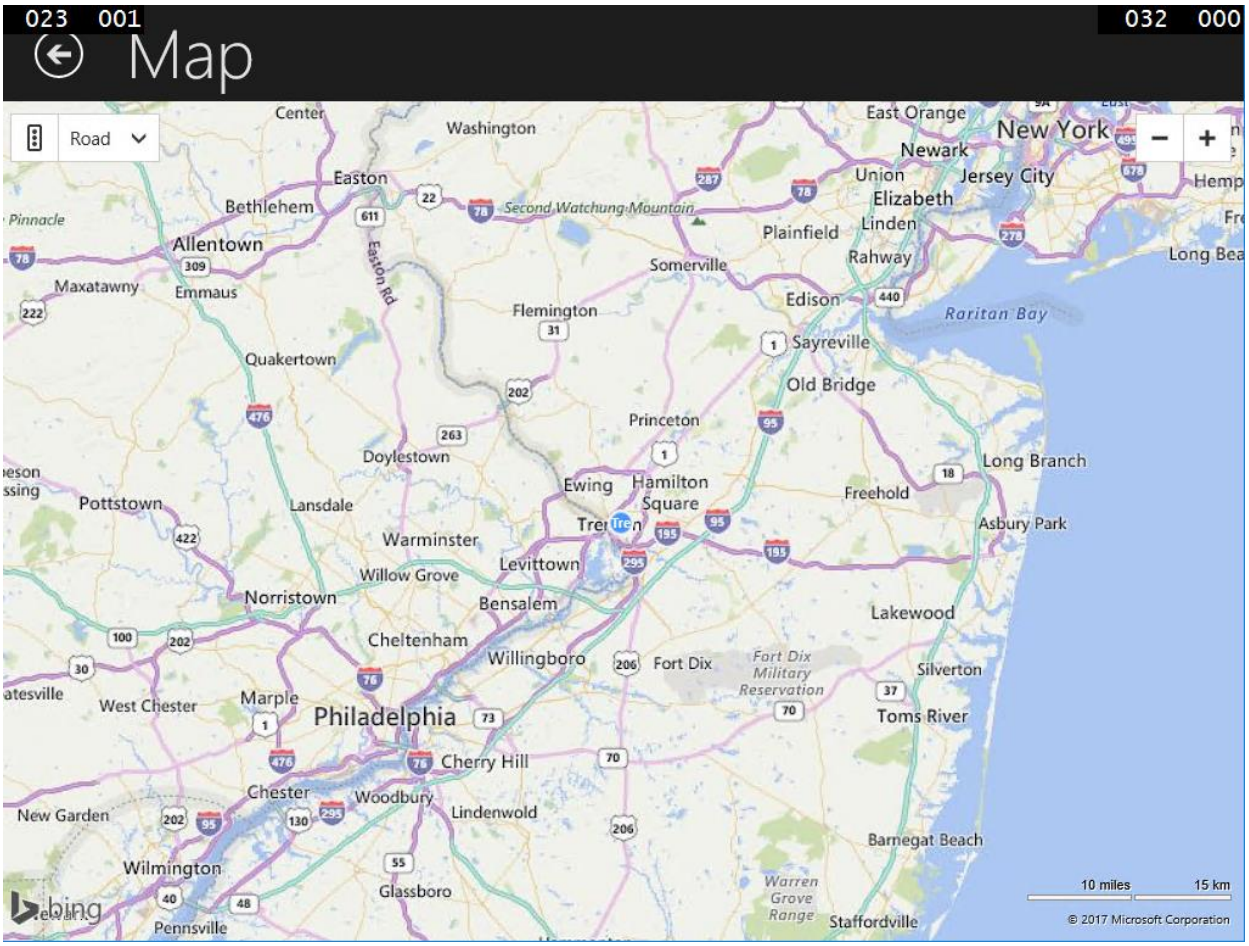

Figure 3– Third screen shows the city location on the map

Finally, there is a small table showing Live Weather Data. If the user clicks on one attribute, the app will show a PI Coresight display of the trend of the selected attribute.

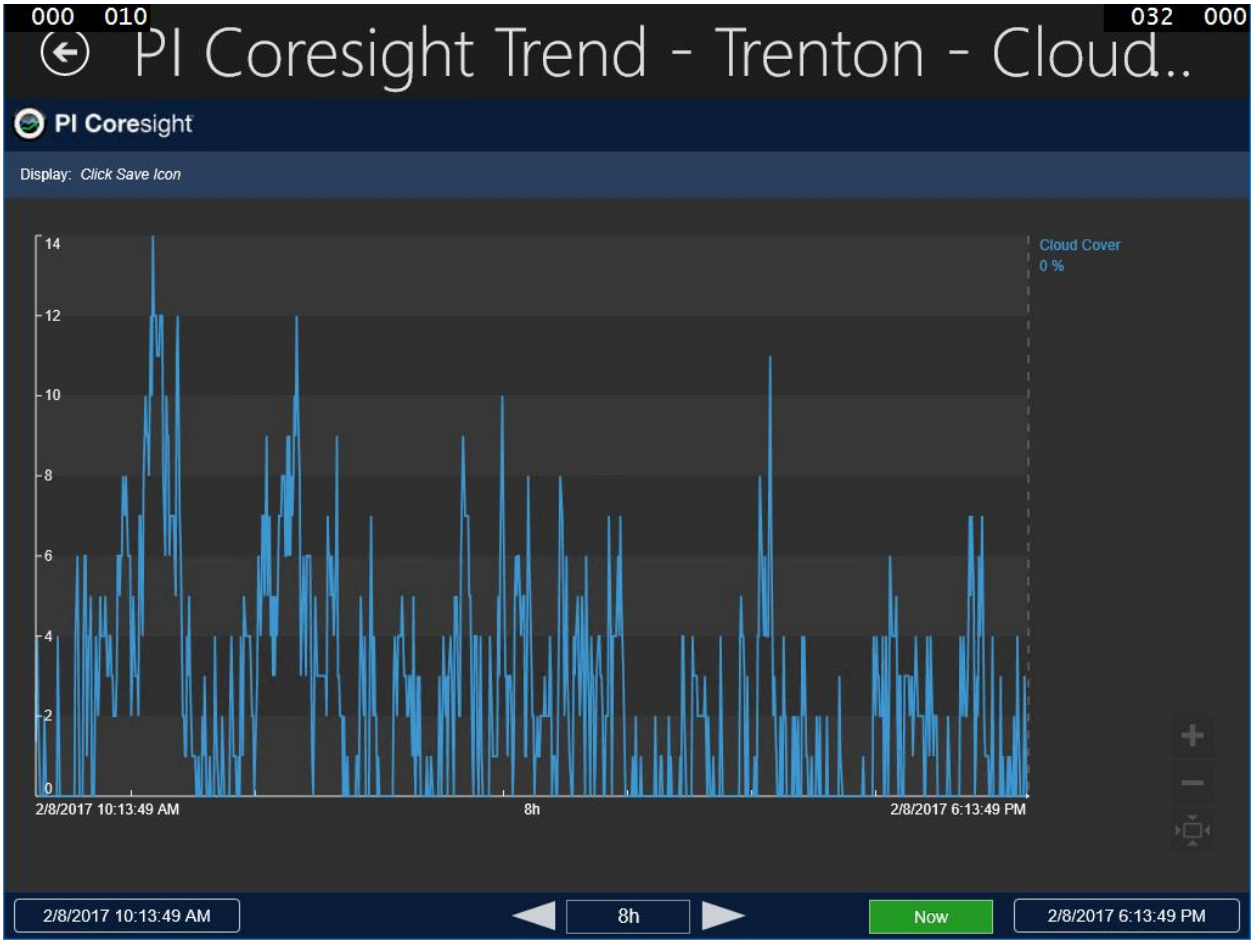

Figure 4 – Fourth screen shows the PI Coresight ad-hoc display of the selected attribute from the selected city

Introduction for the exercises

This lab has 5 exercises.

- Exercises 1 and 5 will focus on PI Web API calls.
- Exercise 3 will explain how integrate your app with PI Coresight.
- Exercises 2 and 4 are about Xamarin.Forms.

There are two Visual Studio solutions on the repository.

- Use **PIWeatherXamarinApp_Exercise** for the lab.
- Use **PIWeatherXamarinApp_Solution** if you want to view the solution of the exercises or test the complete lab solution.

Please use the following information to make HTTP calls against PI Web API:

- Public end-point: **https://pisrv01.pischool.int/piwebapi**
- AF Server name: PISRV01
- AF Database name: Weather
- Authentication method: Basic
- Domain: pischool.int (NetBIOS: pischool)
- Username: student01
- Password: student

The PI Coresight web site is: https://pisrv01.pischool.int/Coresight. The credentials are the same used for PI Web API.

The source code package of this lab can be downloaded from our GitHub repository:

https://github.com/osimloeff/Xamarin-TechCon2017

Once you open both **PIWeatherXamarinApp** projects, make sure that **PIWebatherXamarinApp.Windows** is the StartUp project and the platform target is x86. Since the projects PIWeatherXamarinApp.Droid, PIWeatherXamarin.Ios and PIWeatherXamarinApp.WinPhone won't be used in this lab, they can be unloaded from the solution in order to make it easier to manage.

If the app is opened and you want to go back to Visual Studio, make sure tha app is select and press Alt + Tab together.

Exercise 1: Getting data from PI Web API

Once the user starts the Xamarin app, the cities' data need to be retrieved from the PI System through PI Web API in order to be rendered on the screen.

Open the *CustomWebClient.cs* file on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. There is a method called *GetCitiesDataNoBatch.*

This method should return an object of the *Cities* class, which is a collection of the *City* object. Each *City* object should have 3 properties filled:

- Name → Name of the city.
- ElementWebId → The *webId* of the element which represents the city on the AF tree.
- ImageUrl → The URL of the image of the city.

Without hardcoding any *WebId* value within the code itself, complete this method in order to retrieve PI data and generate a *Cities* collection with 50 *City* objects with those 3 properties above properly filled. The maximum number of PI Web API calls allowed is 3.

Note that the Index search crawler was not set up. Do not call any action within the PI Web API Search controller.

Hints

You should make 3 HTTP requests against PI Web API. Here is a description of each one of them:

- Find the *WebId* of the Weather database (convert to *PIObject*).
- Find all the 50 attributes called *Wikipedia Thumbnail Url* (convert to *PIListObject*).
- Get the values of those 50 attributes by making a bulk call (convert to *PIListObject*).

You should use the *JsonConvert.DeserializeObject* from JSON.NET, which has already been added to the project in order to convert the JSON string into a C# object.

For the bulk call, use the method *GetItemsWebIds* from the *PIListObject,* which will return a *List<string>* with all the requested *WebIds*.

There is a method called *DownloadWebData* to retrieve the JSON string response from PI Web API.

Finally, once you've finished, start the **PIWeatherXamarinApp** and make sure the first screen shows 50 state capitals.

Solution

To view the solution, open the *CustomWebClient.cs* file on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Solution** folder and review the code of the *GetCitiesDataNoBatch()* method.

Exercise 2: Xamarin Forms Maps

When the user selects a city from the list, a second screen will show specific information about the city. There is a button on the screen to show where the city is located on the map. Nevertheless, if you click on this button, nothing happens. This is what this exercise is about.

Open the *MapPage.cs* file on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. There is a constructor for the *MapPage* class. Add some lines of code to the constructor in order to show the city location on the map using Xamarin Forms Map.

Hints

Read the hints below before starting to work on Visual Studio:

- If you search for "*Xamarin Forms Map*" on Google, the first pages found will be very useful for you to complete this exercise.
- Remember that you need to use Xamarin Forms Map on Windows 10.
- The *Xamarin.Forms.Map* library was already added to the **PIWeatherXamarinApp** and **PIWeatherXamarinApp.Windows** projects.
- The *Xamarin.FormsMaps.Init* call was already added to the *MainPage.xaml.cs* file from the **PIWeather.XamarinApp.Windows** project.
- Use the *Latitude* and *Longitude* properties from the *City* class, which is the input from the constructor of the *MapPage* class, to centralize the map on the selected city location.
- Don't forget to call *Navigation.PushAsync* within *OnViewMapBtnClicked* method from the *CityDetailsPage* class the in order to switch to the map display.

Once you've finished, start the **PIWeatherXamarinApp,** select a city from the list and make sure you can see the city location on the map after clicking on the "*View on Map*" button.

Solution

To view the solution, open the *MapPage.cs* and *CityDetailsPage.xaml.cs* files on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Solution** folder and review the code from both files.

Exercise 3: Learning how to use ListView

On this exercise, a table will be added to the screen showing specific information about the city. This table will contain live weather data of the city including the following attributes:

- Visibility
- Cloud Cover
- Temperature
- Wind Speed
- Humidity

Open the *CityDetailsPage.xaml* file from the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. There is a label whose text is "*Live Weather Data*". Add the table just below this label using *ListView* class from Xamarin.Forms on the XAML.

Hints

Read the hints below before starting to work on the Visual Studio:

- If you search for "*ListView Xamarin.Forms*" on Google, the first pages found will be very useful for you to complete this exercise.
- The current data is available within the local variable of the constructor of the *CityDetailsPage* class named *data*. The type of this variable is *List<WeatherData>.*

Once you've finished, start the **PIWeatherXamarinApp,** select a city from the list and make sure the live weather data is shown.

Solution

To view the solution, open the *CityDetailsPage.xaml* and *CityDetailspage.xaml.cs* files on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Solution** folder and review those files.

Exercise 4: Integrating your app with PI Coresight

In the previous exercise, a table showing the city live weather information was added to the screen. But if the user clicks on any row of the table, nothing happens.

In this exercise, we will add functionality such that when the user clicks on a row of the table, a PI Coresight display will appear showing the trends of the selected city attribute.

First of all, open the *CityDetailsPage.xaml.cs* file from the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. You need call *Navigation.PushAsync* when the user clicks on an item of the *ListView*. This method will show the *PITrendsPage* with the PI Coresight display.

Then, open the *PITrendsPage.xaml.cs* file from the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. Add some lines to the *PITrendsPage* constructor in order to show the PI Coresight trend on the *WebView.*

Nevertheless, you should implement different URLs for Android and Windows. Since Android accepts writing the username and password in the URL, this will be implemented to avoid asking the customer to type it.

Hints

Read the hints below before starting to work on this exercise:

- If you open the *PITrendsPage.xaml,* you will find the *WebView* object.
- The URL of the *WebView* should be set on the *Source* property. The URL of the PI Coresight display could be returned by the *GetCoresightUrl* method from the *WeatherData* class. This method has an input to return the URL with the credentials or not.
- Use *Device.OnPlatform* to call different methods for different mobile platforms.

Once you've finished, start the **PIWeatherXamarinApp**, select a city from the list, click on a row of the live weather data and make sure a PI Coresight Trend is shown on the screen.

Solution

To view the solution, open the *PITrendsPage.xaml.cs*, *CityDetailsPage.xaml.cs* and *WeatherData.cs* files on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Solution** folder and review the code.

Exercise 5: Using PI Web API Batch

Exercise 1 was about retrieving PI data from the PI System through PI Web API in order to render the 50 cities on the screen with their images. It needed 3 HTTP requests to retrieve all information required. On this exercise, the same data will be retrieved from PI Web API with only 1 call (instead of 3), using batch.

Open the *CustomWebClient.cs* file on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Exercise** folder. There is a method called *GetCitiesDataWithBatch*.

This method should return an object of the *Cities* class, which is a collection of the *City* object. Each *City* object should have 3 properties filled:

- Name → Name of the city.
- ElementWebId → The webId of the element which represents the city on the AF tree.
- ImageUrl → The URL of the image of the city.

Without hardcoding any *WebId* value within the code itself, complete this method in order to retrieve PI data and generate a *Cities* collection with 50 *City* objects with those 3 properties above properly filled using PI Web API Batch. Remember that the maximum number of PI Web API calls allowed is 1.

Do not forget to change from *webClient.GetCitiesData()* to *webClient.GetCities(true)* on the *MainPage.xaml.cs* file in order to retrieve PI data through batch.

Hints

You should make 1 PI Web API batch request against PI Web API with 3 internal requests written on the body requests. Here is a small description of each internal request:

1. Find the *WebId* of the *Weather* database.
2. Find all the 50 attributes called *Wikipedia Thumbnail Url*
3. Get the values of those 50 attributes.

For the last internal request, the url itself won't be the same when compared to the one used on exercise 1. A different strategy should be used in order to make 1 HTTP request against PI Web API. Consider using the *RequestTemplate* field for the third request.

The method *PostWebData* should be used in order to make the HTTP POST request.

Open the Windows application and make sure the list of the 50 cities still appear on the screen.

Solution

To view the solution, open the *CustomWebClient.cs* file on the **PIWeatherXamarinApp** project located under the **PIWeatherXamarinApp_Solution** folder and review the code of the *GetCitiesDataWithBatch()* method.