PI World 2018 Lab

Developing Modern Web Apps Using Angular 5 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: April 10, 2018

Table of Contents

Introduction

What is Angular?

Angular 2, 4 and 5 are conceived as a complete rewrite of AngularJS (Angular 1.x) in order to fulfill the expectations of modern developers who demand speed and responsiveness from their web applications. By offering faster initial loads, improved changed detection and improved rendering, Angular 5 is up to five times faster than AngularJS. Angular 5 is sleeker and composes a more simplified API.

Many of the original constructs have been removed in favor of easier to remember patterns for writing our code. By copying the huge success of the React framework, it decided to put web components at the core of Angular 5. As a result, Angular apps are now much more responsive to changes in state. The latest version of JavaScript wasn’t around when AngularJS was first created. By taking advantage of some of the new features like classes, modules and decorators, Angular 5 takes building sophisticated high performing applications to a whole new level. Angular 5 is written in Typescript, which is a superset of JavaScript. Typescript gives all the enhancements of ES2015 including built-in strong typing.

All in all, Angular 5 is a great framework to create modular, maintainable and testable applications.

How do you develop an Angular web app on top of the PI System?

If you are a PI System developer already familiar with Typescript 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 easy to create Angular web apps which will interact with the PI System through PI Web API.

What do I need to get started with Angular?

By looking at the previous answer, it is clear that the server side of our web application will be handled by PI Web API. Therefore, only the client side needs to be developed. Given this scenario, the following components, browser and IDE are going to be used:

- Node.js/npm
- Typescript
- Angular CLI
- Visual Studio Code
- Google Chrome

PI Web API client library for Angular 5 (2017 R2)

Overview

This chapter was copied from the README.MD file from the PI Web API client library for Angular 5 GitHub repository, which was developed on top of the PI Web API 2017 R2 Swagger specification.

Requirements

PI Web API 2017 R2 installed within your domain using Kerberos or Basic Authentication. If you are using an older version, some methods may not work.

- Angular 5.0+

Installation

To install this library, run:

```
$ npm install angular-piwebapi –save
```

Consuming the PI Web API client library

Update your Angular `AppModule` by adding the PIWebAPIService as a provider:

```
import { PIWebAPIService } from 'angular-piwebapi';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [PIWebAPIService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Documentation

All classes and methods from this client library are described on the DOCUMENTATION.

Notes

- It is highly recommended to turn debug mode on in case you are using PI Web API 2017 R2+. By default, PI Web API sends a generic error message which is not useful to troubleshoot development issues. In order to receive more detailed exception errors, please turn the debug mode on by creating or editing the DebugMode attribute's value to TRUE from the System Configuration element.
- The X-Requested-With header is added automatically by the library to work with CSRF defences.

Examples

Please check the sample_main.txt from this repository. Below there are also code snippets written in Angular 5 for you to get started using this library:

Set up the instance of the PI Web API top level object

If you want to use basic authentication instead of Kerberos, set useKerberos to false and set the username and password as shown below:

Basic authentication

```
this.piWebApiHttpService.configureInstance("https://devdata.osisoft.com/piweb
api/", false, "username", "password");
```

Kerberos authentication

```
this.piWebApiHttpService.configureInstance("https://devdata.osisoft.com/piweb
api/", true);
```

If you want to test if it connects, just execute the code below:

```
    this.piWebApiHttpService.home.get().subscribe(res => {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

Get the PI Data Archive WebId

```
this.piWebApiHttpService.dataServer.getByPath('\\\\SERVERNAME').subscribe(res
=> {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

Create a new PI Point

```
    var pointName = "SINUSOID_TEST74" + Math.trunc(100000*Math.random());
    var newPoint = new PIPoint(null, null, pointName, null, "Test PI Point
for Angular PI Web API Client", "classic", "float32", null, null, null,
false);
```

```
    this.piWebApiHttpService.dataServer.createPoint(res.WebId,
newPoint).subscribe(res => {
                console.log(res);
        }, error => {
                console.log(error.json());
        });
```

Get PI Point WebId

```
    this.piWebApiHttpService.point.getByPath("\\\\MARC-
PI2016\\sinusoid").subscribe(res => {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

Get recorded values in bulk using the StreamSet/GetRecordedAdHoc

```
    var point1webId =
"P0QuorgJ0MskeiLb6TmEmH5gAQAAAATUFSQy1QSTIwMTZcU0lOVVNPSUQ";
    var point2webId =
"P0QuorgJ0MskeiLb6TmEmH5gAgAAAATUFSQy1QSTIwMTZcU0lOVVNPSURV";
    var point3webId =
"P0QuorgJ0MskeiLb6TmEmH5g9AQAAAATUFSQy1QSTIwMTZcQ0RUMTU4";


    var webIds = []
    webIds.push(point1webId);
    webIds.push(point2webId);
    webIds.push(point3webId);

    this.piWebApiHttpService.streamSet.getRecordedAdHoc(webIds, null, "*",
null, true, 1000, null, "*-3d", null).subscribe(res => {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

Send values in bulk using the StreamSet/UpdateValuesAdHoc

```
    let streamValuesItems : PIItemsStreamValues = new PIItemsStreamValues()
    let streamValue1 : PIStreamValues = new PIStreamValues()
    let streamValue2 : PIStreamValues = new PIStreamValues()
    let streamValue3 : PIStreamValues = new PIStreamValues()

    let value1 : PITimedValue = new PITimedValue()
    let value2 : PITimedValue = new PITimedValue()
    let value3 : PITimedValue = new PITimedValue()
    let value4 : PITimedValue = new PITimedValue()
    let value5 : PITimedValue = new PITimedValue()
    let value6 : PITimedValue = new PITimedValue()

    value1.Value = 2
    value1.Timestamp = "*-1d"
    value2.Value = 3
    value2.Timestamp = "*-2d"
    value3.Value = 4
    value3.Timestamp = "*-1d"
    value4.Value = 5
    value4.Timestamp = "*-2d"
    value5.Value = 6
    value5.Timestamp = "*-1d"
    value6.Value = 7
    value6.Timestamp = "*-2d"

    streamValue1.WebId = point1webId
    streamValue2.WebId = point2webId
    streamValue3.WebId = point3webId

    var values1 = [];
    values1.push(value1)
    values1.push(value2)
    streamValue1.Items = values1

    var values2 = [];
    values2.push(value3)
    values2.push(value4)
    streamValue2.Items = values2

    var values3 = [];
    values3.push(value5)
    values3.push(value6)
    streamValue3.Items = values3

    var streamValues = []
    streamValues.push(streamValue1)
    streamValues.push(streamValue2)
    streamValues.push(streamValue3)
    this.piWebApiHttpService.streamSet.updateValuesAdHoc(streamValues, null,
null).subscribe(res => {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

Using PI Web API Batch

```
    var pirequest = {};
    pirequest["4"] = {
        "Method": "GET",
        "Resource": "https://marc-web-
sql.marc.net/piwebapi/points?path=\\\\MARC-PI2016\\sinusoid",
        "Headers": {
            "Cache-Control": "no-cache"
        }
    };
    pirequest["5"] = {
        "Method": "GET",
        "Resource": "https://marc-web-
sql.marc.net/piwebapi/points?path=\\\\MARC-PI2016\\cdt158",
        "Headers": {
           "Cache-Control": "no-cache"
        }
    };
    pirequest["6"] = {
        "Method": "GET",
        "Resource": "https://marc-web-
sql.marc.net/piwebapi/streamsets/value?webid={0}&webid={1}",
        "Parameters": [
            "$.4.Content.WebId",
            "$.5.Content.WebId"
        ],
        "ParentIds": [
            "4",
            "5"
        ]
    };
    this.piWebApiHttpService.batch.execute(pirequest).subscribe(res => {
        console.log(res);
    }, error => {
        console.log(error.json());
    });
```

If you know the name of the action and controller you want to call, the syntax to do so is as follows:

PIWebAPIService.{controllerName}.{actionName}(inputs);

Please read the PI Web API programming reference to find the names of the actions, controllers and inputs.

If you want to check out the details about how to use each method available on the library, please refer to DOCUMENTATION.

PI Weather Angular 5 Sample App

The sample application of this lab is developed using Angular 5, which is a platform that makes it easy to build applications with the web.

The **PI Weather Angular 5 Sample 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.

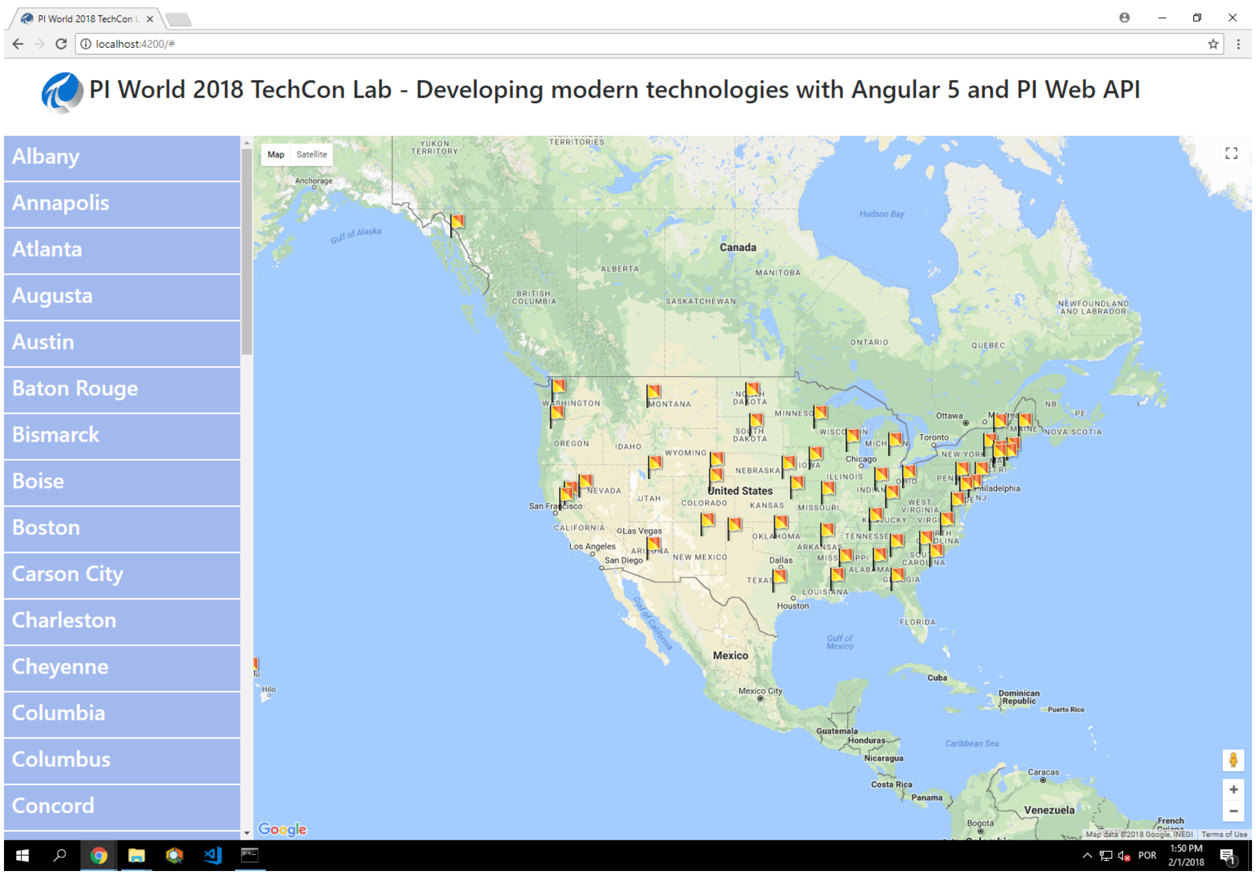
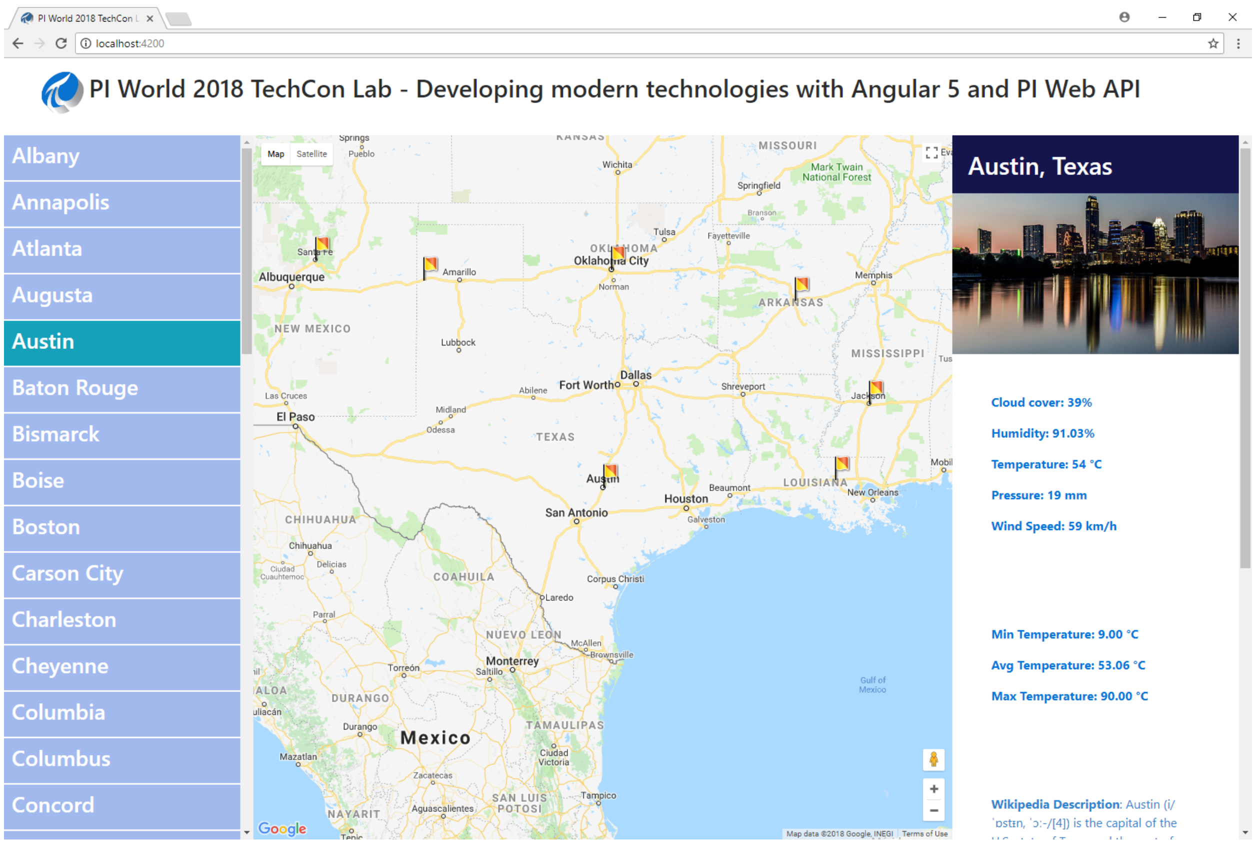


Figure 1 – First screen shows a list with the 50 state capitals of the United States of America on the left pane and a map on the right pane

On the second screen, which appear when the user selects a city, there is an image of the city, a small description from Wikipedia and weather data.



PI World 2018 TechCon Lab - Developing modern technologies with Angular 5 and PI Web API

Austin, Texas

Cloud cover: 39%

Humidity: 91.03%

Temperature: 54 °C

Pressure: 19 mm

Wind Speed: 59 km/h

Min Temperature: 9.00 °C

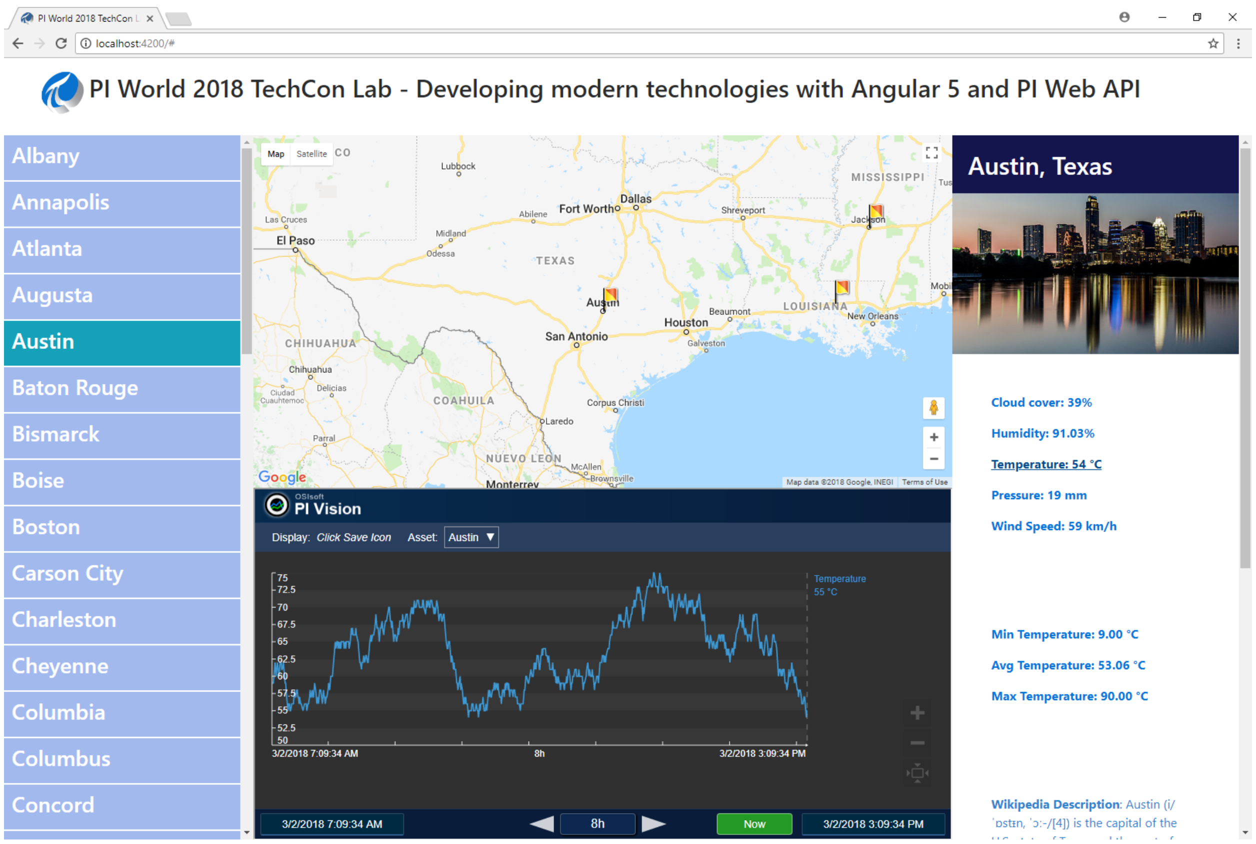
Avg Temperature: 53.06 °C

Max Temperature: 90.00 °C

Wikipedia Description: Austin (/ˈɒstɪn, ˈɔː-/[4]) is the capital of the

Figure 2 – Second screen shows a photo and the Wikipedia description of the selected city, live weather information and the city location on the map

Finally, if the user clicks on one attribute, the app will show a PI Vision display of the trend of the selected attribute.



PI World 2018 TechCon Lab - Developing modern technologies with Angular 5 and PI Web API

- Albany
- Annapolis
- Atlanta
- Augusta
- Austin
- Baton Rouge
- Bismarck
- Boise
- Boston
- Carson City
- Charleston
- Cheyenne
- Columbia
- Columbus
- Concord

Austin, Texas

Cloud cover: 39%

Humidity: 91.03%

Temperature: 54 °C

Pressure: 19 mm

Wind Speed: 59 km/h

Min Temperature: 9.00 °C

Avg Temperature: 53.06 °C

Max Temperature: 90.00 °C

Wikipedia Description: Austin (/ˈɒstɪn, ˈɔː-/[4]) is the capital of the

Figure 3 – Third screen shows the PI Vision ad-hoc display of the selected attribute from the selected city embedded on the web app.

Introduction for the exercises

This lab has 5 exercises.

- Exercises 1 and 2 are about retrieving the cities geolocation.
- Exercises 3 is about getting live and performing calculations.
- Exercise 4 will explain how integrate your app with PI Vision 3.
- Exercise 5 is about PI Web API Batch

There are two Visual Studio solutions on the repository.

- Use **Exercise** for the lab.
- Use **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: **Kerberos**
- Domain: pischool.int **(NetBIOS: pischool)**
- Username: **student01**
- Password: **student**

The PI Vision 3 web site is: https://pisrv01.pischool.int/PIVision. 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/Modern-Web-Technologies-TechCon2018

Once you open any project from the repository (Exercise or Solution), make sure to open the command prompt and run “ng serve” to start the web server. Once the web app is started you need to open a web browser to access to web content. It is recommended to open Visual Studio Code, go to the Debug pane and click on “Start Debugging”. Google Chrome will open up your application. The advantage of opening it through VS Code is that you can use this IDE to debug the app instead of Google Chrome Developer Tools.

For this lab, an AF database will be used called Weather with 50 elements derived from the City element template. Each city represents a United States capital and it has many attributes related to not only Weather such as temperatures, precipitation and humidity but also to geolocation including latitude and longitude.

Exercise 1: Getting data from PI Web API

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

Open the *app.component.ts* file under the **src/app** folder of the **Exercise** project folder. There is an empty method called *getDataNoBatch().*

This method should retrieve information about the 50 elements as well as their latitude and longitude Web IDs. This step is required before retrieving the latitude and longitude values which will be done on exercise 2.

Without hardcoding any *WebId* value within the code itself, complete this method to retrieve the requested PI data. Your code needs to call the *processResponses()* method whose unique input is generated using the responses of the PI Web API requests. The maximum number of PI Web API calls allowed is 3. Try to reduce the size of the response as much as possible.

Hints

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

1. Get the 50 elements, each one represents a city.
2. Find all the 50 attributes Web IDs called *Latitude*
3. Find all the 50 attributes Web IDs called *Longitude*

You need the *WebId* of the AF database in other to make the 3 HTTP requests (2, 3 and 4). The *WebId* of the AF database should be generated using the PI Web API client library for Angular, without making an HTTP request under the hood. Please refer to the *piwebapi.webIdHelper.generateWebIdByPath()* method. The last 3 requests can be called in parallel by using the *Observable.forkJoin()* method, available on the Angular framework, in order to improve the performance.

You don't need to get the latitude and longitude values since this will be done in exercise 2.

In the end, call *this.processResponses(res)* respecting the type of the input of the method. This method will process the responses from the requests above and add the cities in the map.

The *selectedFields* input might be used to reduce the size of the HTTP responses.

Finally, once you've finished, start the **Exercise** project and make sure that the 50 state capitals are shown on the left pane. At this point, we don't expect the cities to be added to the map.

Solution

To view the solution, open the *app.component.ts* file under the **src/app** folder located under the **Solution** project and review the code of the *getDataNoBatch()* method.

Exercise 2: Getting geolocation information

The web application should show the location of the 50 cities on a map. If you run “ng serve” on the command prompt from the **Exercise** project folder, you will realize that this is still not taking place because in order to add a city to a map, the latitude and longitude values need to be known.

Open the *app.component.ts* file under the **src/app** folder from the **Exercise** project folder. If you review the *processReponses()* method, you will notice that each city stores its latitude and longitude Web IDs as properties. Add some lines of code within the *processResponses()* method in order to get all cities geolocation (latitude and longitude) using the *city.latitudeWebId* and *city.longitudeWebId* properties as inputs.

Hints

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

1. Review the methods from the **StreamSet** controller in order to find the most suitable for this scenario.
2. Store the latitude and longitude values on the *city.latitude* and *city.longitude* properties.
3. The variable *this.cities* shall be updated with the cities latitude and longitude values.
4. Review the function getGeolocationValue() and call it when appropriate.
5. Use the online documentation from the PI Web API client library for Angular. Then, navigate to the StreamSet controller documentation.

Once you’ve finished, start the **Exercise** project and make sure you can see a map with the 50 Unites States capitals.

Solution

To view the solution, open the *app.component.ts* file under the **src/app** folder under the **Solution** folder and review the code.

Exercise 3: Showing weather live and performing calculations

On this exercise, two tables will be added to the detailed pane showing specific information about the city. They will contain live and calculated weather data of the selected city including the following attributes:

- Live Visibility
- Live Cloud Cover
- Live Temperature
- Live Wind Speed
- Live Humidity
- Maximum Temperature
- Minimum Temperature
- Average Temperature

Open the *details-pane.component.html* file under the **src/app/details-pane** folder under the **Exercise** project folder. There is a HTML code snippet commented. The first step is to uncomment the code which shows both tables on the detail pane.

Then, edit the *applySelectCity()* method from the *right-pane.component.ts* under the **src/app/right-pane** folder. Write some lines of code in order to not only get all live values from a given element but also the maximum, minimum and average temperature from the last 24 hours.

Create two methods on the *DetailsPaneComponent* that are going to be called by the *applySelectCity()*:

- *showValues()* → this function will update the *this.selectedCity* with updated live values and the temperature WebId, which is going to be used on the following request.
- *showTemperatureRange()* → this function will update the *this.selectedCity* with calculated values from the last 24 hours of the city temperature.

Make sure that the live data and temperature values are shown on the tables of the *details-pane.component.html* by looking at the properties name of the *selectedCity* variable.

Hints

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

- You can uncomment a code snippet by pressing Ctrl + K then Ctrl + U.
- The *showValues()* method has two inputs: selectedCity (class City) and cityValues (class PIItemsStreamValue).
- Use the *getValues()* method from the *StreamSet* controller of the PI Web API client library and the *getSummary()* method from the *Stream* controller within the *applySelectCity()* method.
- The *showTemperatureRange()* method has one input values (class PIItemsSummaryValue).
- Import the *PIItemsSummary* value on the top of the *details-pane-components.ts* file.

Once you've finished, wait for the Angular CLI restart the web app and then select a city from the left pane list and make sure the live and calculated weather data is shown.

Solution

To view the solution, open the *right-pane.component.html*, *details-pane.component.html* and *details-pane.component.ts* files under the the **src\app\right-pane** and **src\app\details-pane** folders located under the **Solution** project and review the code.

Exercise 4: Integrating your app with PI Vision 3

In the previous exercise, tables showing the city weather information were added to the details pane. But if the user clicks on any row of the top table, nothing happens.

In this exercise, we will add a new feature so that when the user clicks on any row of the top table, a PI Vision display will appear below the map showing the PI Vision trend of the selected city attribute.

First of all, open the *map-pane.component.html* file from the **src/app/map-pane** folder under the **Exercise** project folder and add an iframe under div whose id is “trend”.

Then, edit the *openVision()* method from the *details-pane.component.ts* to build the PI Vision URL that will display the trend of the selected attribute. This method will show the trend with the PI Vision display. You might want to download the PI Vision User Guide from the OSIsoft TechSupport Download Center to help you with this task.

Finally, open the *details-pane-component.html* and edit the links properties in order to call the *openVision()* method once the link is clicked using the Angular 5 syntax.

Hints

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

- The iframe needs to show the content of the URL stored under the variable *this.piVisionUrl*.
- The link below will help you get started with Angular events binding:
    - https://angular.io/guide/user-input

Once you’ve finished, select a city from the left pane, click on a row of the live weather data on the right pane and then make sure a PI Vision trend is shown below the map on the screen.

Solution

To view the solution, open the *map-pane.component.html*, *details-pane.component.html* and *details-pane.component.ts* files under the the **src\app\map-pane** and **src\app\details-pane** folders located under the **Solution** project and review the code.

Exercise 5: Using PI Web API Batch

Exercises 1 and 2 were about retrieving PI data from the PI System through PI Web API in order to render the 50 cities on the left pane and on the map. It needs 5 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 5), using PI Web API Batch.

Open the *app.component.ts* file under the **src/app** folder of the **Exercise** project. This exercise is about editing the method called *getDataWithBatch().* Nevertheless, within the *ngOnInit()* method,

- Comment the *this.getDataNoBatch()* line and
- Uncomment the *this.getDataWithBatch()* line

The variable *globalRequest* is the body message of the PI Web API POST Batch request.

Without hardcoding any *WebId* value within the code itself, complete this method in order to retrieve the 50 cities information including their geolocation by modifying the *globalRequest* variable content. Remember that the maximum number of PI Web API calls allowed is 1.

Hints

Below there is a suggestion to make 1 PI Web API Batch request against PI Web API with 6 internal requests written on the body request:

1. Find the *WebId* of the Weather database. Since it is a batch request, you can generate the *WebId* on the server-side.
2. Get the 50 elements, each one represents a city (use the response of the request 1).
3. Find all the 50 attributes Web IDs called *Latitude* (use the response of the request 1).
4. Find all the 50 attributes Web IDs called *Longitude* (use the response of the request 1).
5. Get the 50 *Latitude* values (use the response of the request 3).
6. Get the 50 *Longitude* values (use the response of the request 4).

The requests 5 and 6 use the *RequestTemplate* field.

Review the content of the *processResponseWithBatch* method. The response of the batch is the input of this method. As result make sure they are compatible and modify your batch request accordingly.

Solution

To view the solution, open the *app.component.ts* file under the **src/app** folder located under the **Solution** project and review the code of the *getDataWithBatch()* method.

Have an idea how to improve our products?
OSIsoft wants to hear from you!
https://feedback.osisoft.com/

Save the Date!

OSIsoft PI World Users Conference in Barcelona. September 24-27, 2018.

Register your interest now to receive updates and notification early bird registration opening.

http://pages.osisoft.com/UC-CORP-Q3-18-EMEAUsersConference_RegisterYourInterest2018.html