Connect to Couchbase REST API App Tutorial

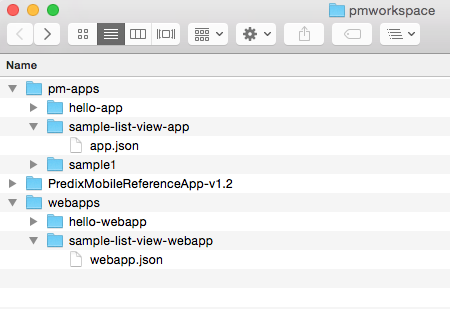
We are glad you made it this far! You’re well on your way to making an app on Predix Mobile. If you have experience in web development, then this tutorial will make you feel right at home. In this tutorial we will be building an application that displays a list of customers and the number of issues that belong to these customers, which in a use case would be alerts such as high voltage readings from a power plant. We will be using the [Foundation for Apps](http://foundation.zurb.com/apps.html) framework which helps you get up to speed quickly for web applications.

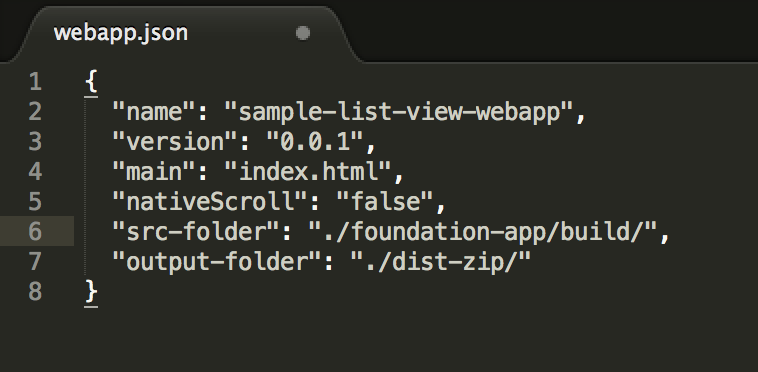
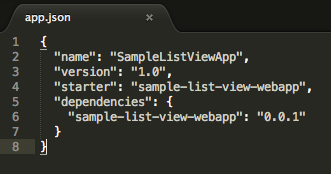
# Requirements

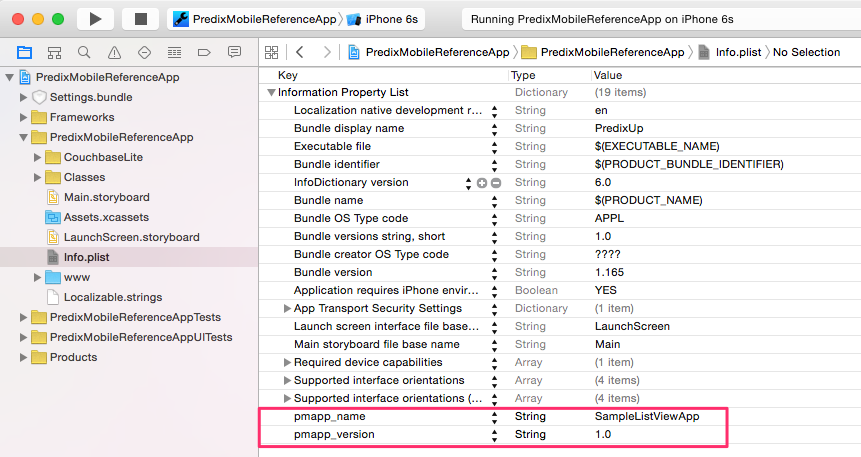
1. At this point, we expect you to have followed the [Getting Started for Predix Mobile Up](https://www.predix.io/docs#rae4EfJ6) and then the Mobile Hello World tutorial. Therefore, you should have everything needed to run your Predix Mobile Application. The only additional software you will need to download are web frameworks if you are using any (we will walk you through using Foundation for Apps).
2. Install [Foundation for Apps](http://foundation.zurb.com/apps.html)

# Setup your folder structure

1. We are creating a new app with a starter web app in the same way we did in the Hello World tutorial.
   1. Your resulting folder structure should look similar to this

NOTE: if you have not followed the previous tutorials, then you will only have the sample-list-view-app and sample-list-view-webapp in the pm-apps/ and webapps/ folder respectively.

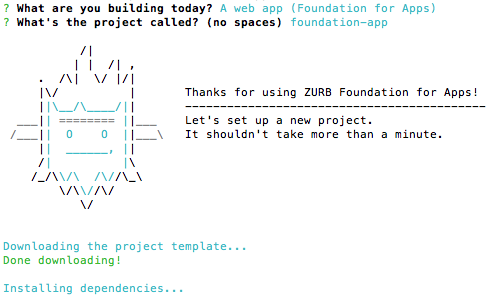
* 1. Your app.json file and webapp.json file should look like this:
  2. You can use the PredixMobileReferenceApp Xcode Project from before, we just need to change the pmapp\_name and pmapp\_version in info.plist to match our new app



# Create the app

This step can be done using whatever web frameworks you prefer, but we will be using Foundation for Apps and that will be the easiest to use when following along.

1. Follow the [Foundation for Apps Getting Started Guide](http://foundation.zurb.com/apps/getting-started.html). For the “Create a new project” step, run the following command in the webapp/sample-list-view-webapp folder.

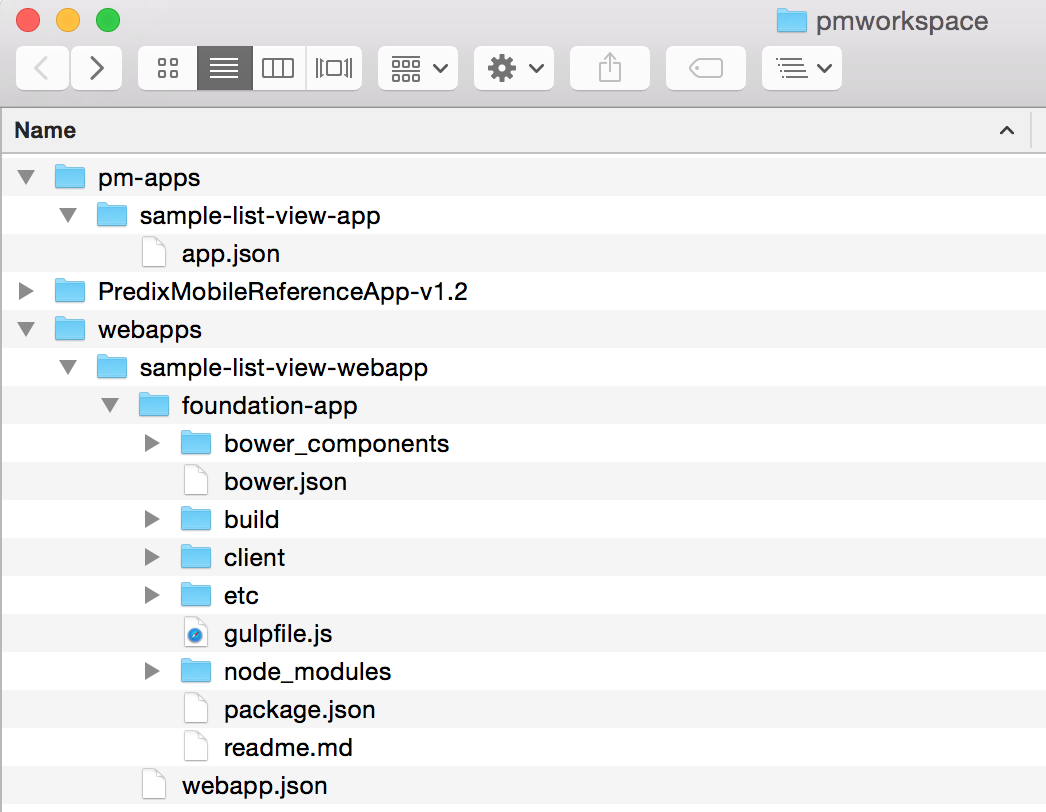
$ foundation new

1. Run the command *$npm start* in the folder containing package.json, which in this tutorial is the foundation-app folder.This will build your webapp, and open it in a browser. Kill (ctrl-c) this process when you are satisfied with the fact that you just created a webapp with one command.

NOTE: *$npm start* will run the start script in package.json which in turn runs a gulp file. These were all created and configured for you Foundation for Apps.

# Deploy the app

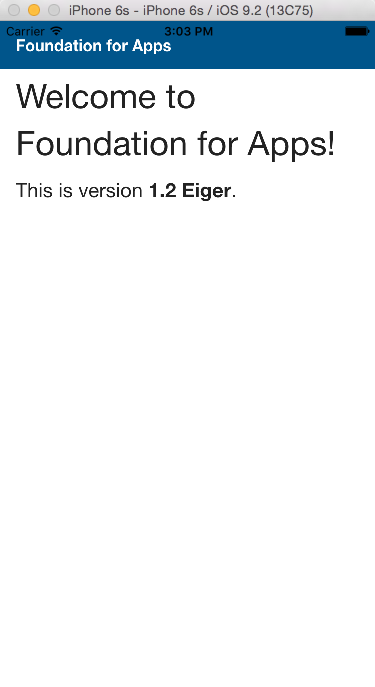
At this point you should have your webapp with the built code in a folder named foundation-app/build/ if you followed the last section “Create the app”. If you are using your own frameworks (or none at all), just ensure your final (built if needed) webapp code is in the directory specified by the src-folder in webapp.json. Following the tutorial your file structure should look similar to this:



1. At this time, we will publish the webapp
   1. $ cd <workspace>/webapps/sample-list-view-webapp
   2. $ pm publish

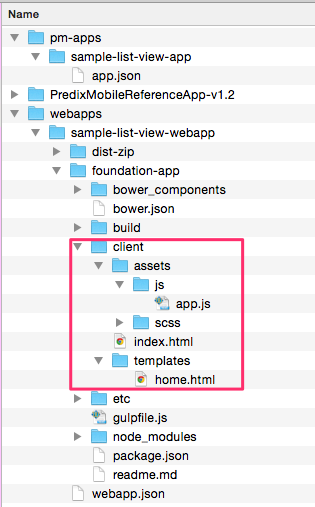
NOTE: If the pm commands give an authorized error, you will need to log in using $ pm auth <your username> <your password> in order to run the publish and define commands.

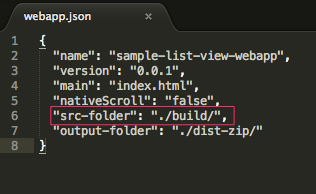
1. Now we need to *define* the app using the pm-tools CLI.
   1. $ cd <workspace>/pm-apps/sample-list-view-app
   2. $ pm define
   3. This is the only time we will need to define the app, unless we need to change anything in the app.json file (i.e. webapp version changes, add dependencies, etc.)
2. Run the app in Xcode (this will point to our new app since we already changed the values in the info.plist file in the PredixMobileReferenceApp). You should see your new app!

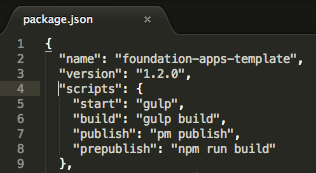


# Update the code in your app

To show you the steps you need to follow when you update your app’s code, you will make a small change and we’ll publish the app again.

The following sections will be dealing with mostly within the client folder indicated by the red box in the picture below. The client folder is where user code is added and changes are made.

1. Before we update any code, lets move webapp.json into the foundation-app/ folder so that it’s easier to include it in our scripts. Also, open webapp.json and change the “src-folder” key to “./build/” since we moved the file.
2. Change the menu bar text in index.html to say something different (e.g. “My first list view”).
3. Now we have to rebuild and publish our app. In your terminal:
   1. cd <workspace>/sample-list-view-webapp/foundation-app
   2. $npm run build
   3. $pm publish
4. Run the app in Xcode to see your change (Note: you will have to reauthenticate your user in the app). NOTE: Now you may be thinking, “This is crazy that I have to rebuild, rerun, and authenticate every time I want to update the code… These are web applications, I should just have to hit refresh in this day and age!” Live updating is possible, although kind of hacky right now (see this blog to learn how). We are working on making this process much simpler.
5. To make this process simpler for you, we’ll combine the build and publish step. In <your workspace>/sample-list-view-webapp/foundation-app/package.json, change the “scripts” like we have below.



1. Now you can run $ npm run publish to build and publish your app.

# Access the REST API in and display data in a list

Here we will access the REST API in the app in order to query and get information from the couchbase database. In this tutorial we will be providing you with an api endpoint but you can find more information on making other types of queries here: [Client Services API Reference](https://www.predix.io/docs#tMguUCx).

1. Open the home.html file within the client folder for editing and match it’s contents to the following code:

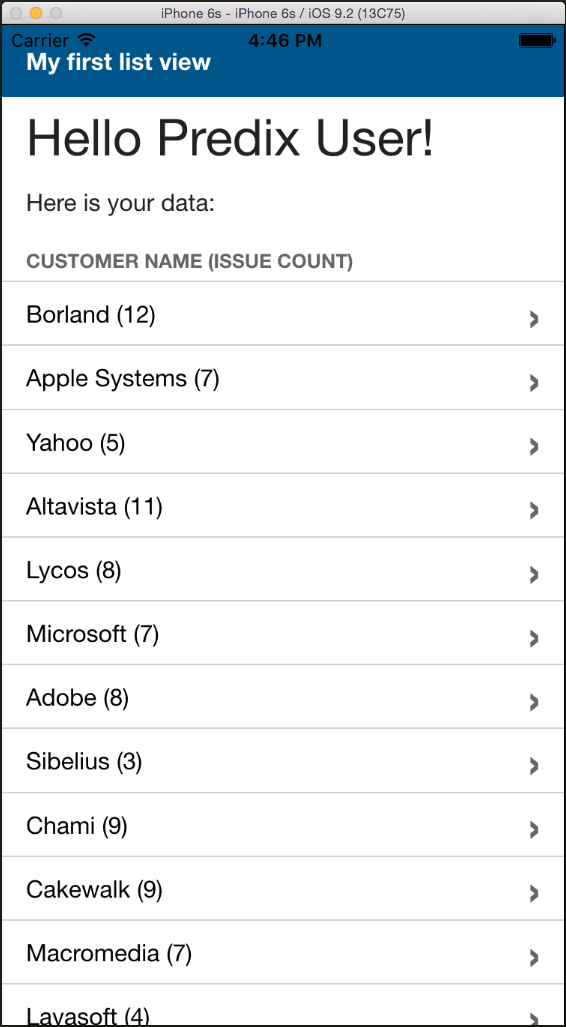
|  |
| --- |
| ---  name: home  url: /  controller: pmController  ---  <!-- The above section is used by foundation for routing (a small wrapper on AngularJS), and it specifies the AngularJS controller used in this file -->  <div class="grid-container">  <h1>Hello Predix User!</h1>  <p>Here is your data:</p>  <section class="block-list">  *<!-- Here is where the information from couchbase will be displayed. Simply match your home.html code to this. The ng-repeat is AngularJS used to repeat the list element for each customer. The title and count properties are the customer name and issue count respectively.-->*  <header>Customer Name (Issue Count)</header>  <ul>  <li ng-repeat='customer in customers' class="with-chevron">  <a href="#">{{customer.title}} ({{customer.count}}) </a>  </li>  </ul>  </section>  </div> |

2. Next we will be editing the app.js located, from the client folder, in asset/js. This is where the http request to couchbase will be made. The endpoint we will be using retrieves all documents (essentially JSON objects with arbitrary contents) and their contents from couchbase and is as follows: [**http://pmapi/cdb/pm/\_all\_docs?include\_docs=true**](http://pmapi/cdb/pm/_all_docs?include_docs=true)**.** Now open the app.js file for editing and match it’s contents to the following code:

|  |
| --- |
| (**function**() {  **'use strict'**;  **angular**.module(**'application'**, [  **'ui.router'**,  **'ngAnimate'**,  *// foundation*  **'foundation'**,  **'foundation.dynamicRouting'**,  **'foundation.dynamicRouting.animations'**  ]).controller(**'pmController'**, **function**($scope, $http, $filter) {  /\* The http function below is what makes the call to couchbase using the endpoint given  at the beginning of this step. It uses an AngularJs Function to make an http GET request.  \*/  $http.get(**"http://pmapi/cdb/pm/\_all\_docs?include\_docs=true"**).then(**function** (response) {  **var** customerList = response.**data**.rows;  customerList = $filter(**'filter'**)(customerList, {**doc**: {**dataType**: **"entity"**}}, **true**)  customerList = $filter(**'uniqueWithCount'**)(customerList,[**"doc"**,**"customer"**]);  $scope.**customers** = customerList;  });  /\* This is a custom filter function to only show unique items, but also shows the count of the number of times the item appeared in the original list. For this tutorial, the number of issues is in our original data, and we’re only showing unique customers, with the count of issues they have.  \*/  }).filter(**'uniqueWithCount'**, **function**() {  **return function**(input, filterKeys) {  **var** output = {};  **var** keys = [];  **var** outputArr = [];  **angular**.***forEach***(input, **function**(item){  **var** value = item;  **var** i;  *// input.doc.customer*  **for** (i=0;i<filterKeys.length;i++) {  value = value[filterKeys[i]];  }  *// Now value should be equal to the value we want to filter on*  **if** (value **in** output) {  *// Increment the count of the key in output*  output[value]++;  } **else** {  *// Add the count of the key to the output*  output[value] = 1;  }  });  **console**.log(output);  **for** (**var** item **in** output){  outputArr.push({**"title"**:item,**"count"**:output[item]});  };  **return** outputArr;  }  })  .config(*config*)  .run(*run*)  ;  // \*\*\* Nothing changed below here \*\*\*  *config*.**$inject** = [**'$urlRouterProvider'**, **'$locationProvider'**];  **function** *config*($urlProvider, $locationProvider) {  $urlProvider.otherwise(**'/'**);  $locationProvider.html5Mode({  **enabled**:**false**,  **requireBase**: **false**  });  $locationProvider.hashPrefix(**'!'**);  }  **function** *run*() {  **FastClick**.attach(**document**.**body**);  }  })(); |

3. Update your code using *npm run publish* as learned in the previous section, **Update the Code in your App**.

4. Run the application using xcode. After logging in you should see something similar to the following:



This is what you see then you have completed the this tutorial. The list in the app is parsed and displayed using data retrieved from couchbase. The app lists the customers that have issues and the total number of issues for that customer.