Introduction

In this blog entry I would like to give an introduction to the NEW Script Portlet using AngularJS. I will also show you how to develop a sample weather application using the Script Portlet using AngularJS.

This sample is only for demo purposes, and is by no means a completed application. In addition, this blog entry does not cover all the features of AngularJS or the Script Portlet.

What is Script Portlet?

The Script Portlet is IBM’s version of http://jsfiddle.net. It allows developers to write an application using JavaScript, HTML and CSS technologies. You only need a browser to develop and run these technologies.

Installing Script Portlet

You can download Script Portlet from Greenhouse. This Portlet is constantly being updated with new features. At the time of this writing the Portlet is at version 1, update 3.

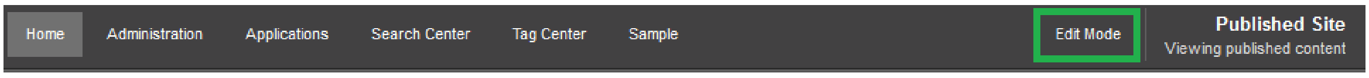
Your portal server should be at least at V8.0.0.1 CF 11 to install version 1.3 of Script Portlet.

Once your Portal is updated, follow the steps in the “readme.pdf” to install the Portlet. This file is located in the .zip file you downloaded from Greenhouse.

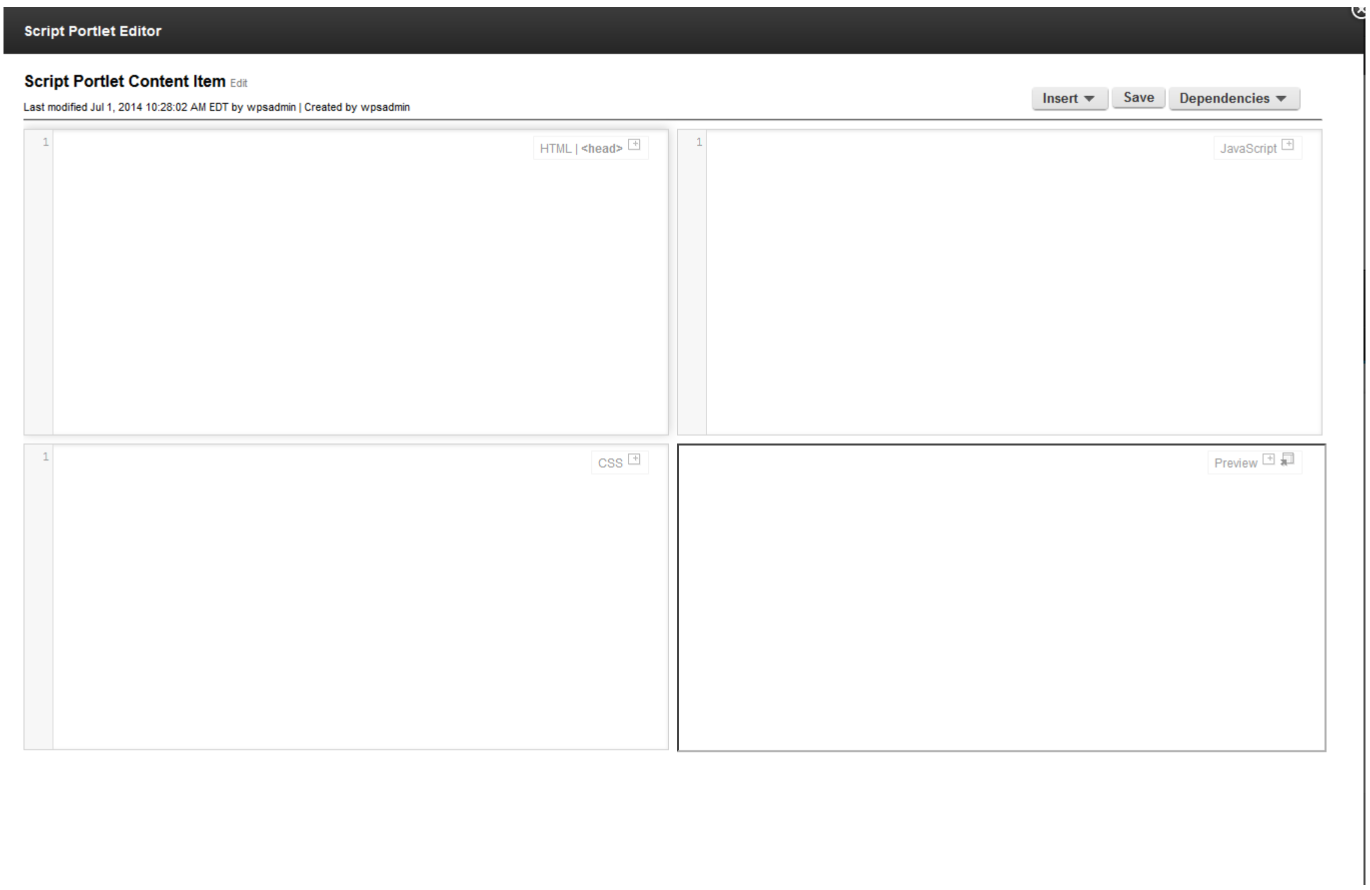
Hello World

So now that you have your Script Portlet, let’s do a simple “Hello World”

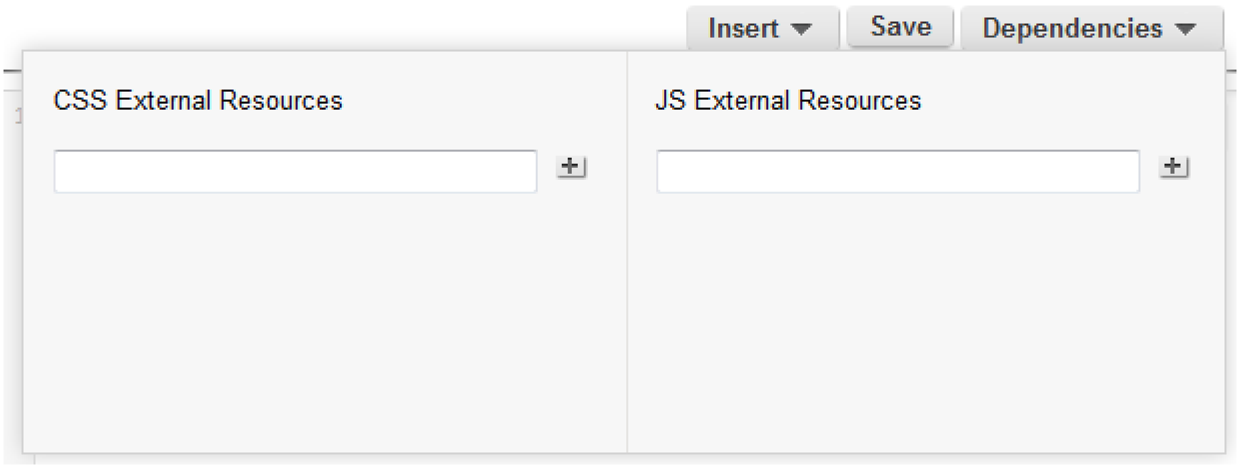
1. Place the Script Portlet on a portal page.
2. Click “Edit Mode” on the top right of the page. This will enable the “Edit” link on your Portlet.



3. Click the “Edit” link on the Portlet. This will launch four panels.

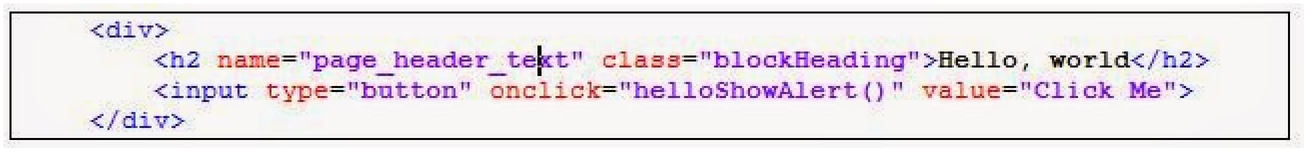

Script Portlet Editor
Script Portlet Content Item Edit
Last modified Jul 1, 2014 10:28:02 AM EDT by wpsadmin | Created by wpsadmin
Insert
Save
Dependencies
HTML | <head>
JavaScript
CSS
Preview

HTML/<head>: In this panel you can write your HTML code. Click on the <head> link to open the Head panel. Here you can write any code that you want to place in the <head> section of your Portal page. Note: If you want to include any JS or CSS files you can place them under the "Depedencies" section, explained later.
CSS: In this section put any CSS classes you want to define.
JavaScript: In this section place any JavaScript code you may have.
Preview: Once you click the "Save" button, your application can be previewed in this panel. For all these panels you have an icon that looks like this: . Click on this icon to expand the panel. In the Preview panel you can also preview the application in a new browser window, just click the new browser icon.
Dependencies: You can define any external JS or CSS dependencies here. Just give the URL and click the "+" button to add the external resource to your application. Any dependencies you define here are automatically added to <head> section of the page.
Rename the Portlet: You can rename the Portlet title. Click "Edit" link next to "Script Portlet Content Item". Rename it to "Hello World".



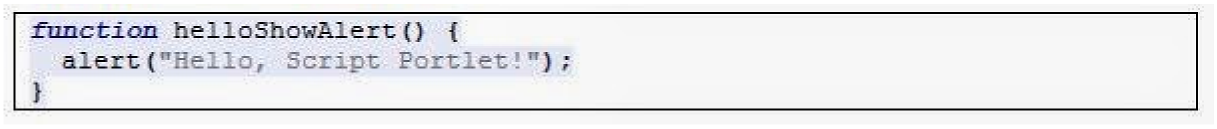
4. Now that we know the interface, let’s continue with our Hello World app. Place the below code in their respective panels.

HTML:



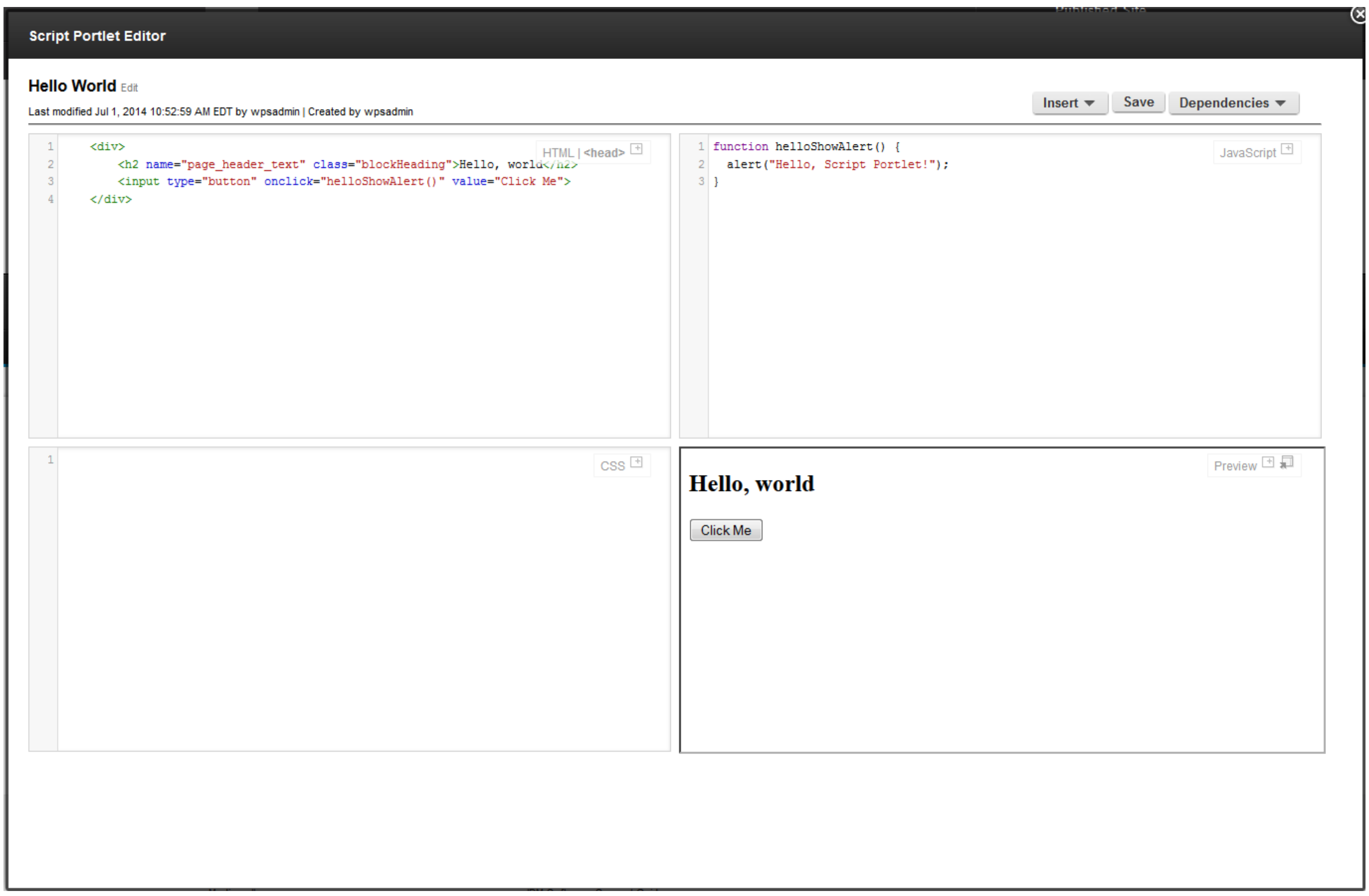
```
    <div>
        <h2 name="page_header_text" class="blockHeading">Hello, world</h2>
        <input type="button" onclick="helloShowAlert()" value="Click Me">
    </div>
```

JavaScript:



```
function helloShowAlert() {
  alert("Hello, Script Portlet!");
}
```

5. Click the Save button. This will render the application in the Preview panel.



Hello World Edit

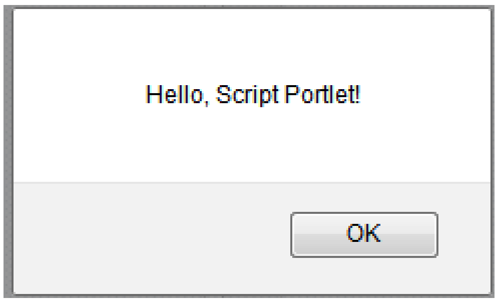
Last modified Jul 1, 2014 10:52:59 AM EDT by wpsadmin | Created by wpsadmin

```html
    <div>
        <h2 name="page_header_text" class="blockHeading">Hello, world</h2>
        <input type="button" onclick="helloShowAlert()" value="Click Me">
    </div>
```

```javascript
function helloShowAlert() {
  alert("Hello, Script Portlet!");
}
```

Hello, world

6. Click on the “Click Me” button. It will display an alert that says “Hello, Script Portlet”.

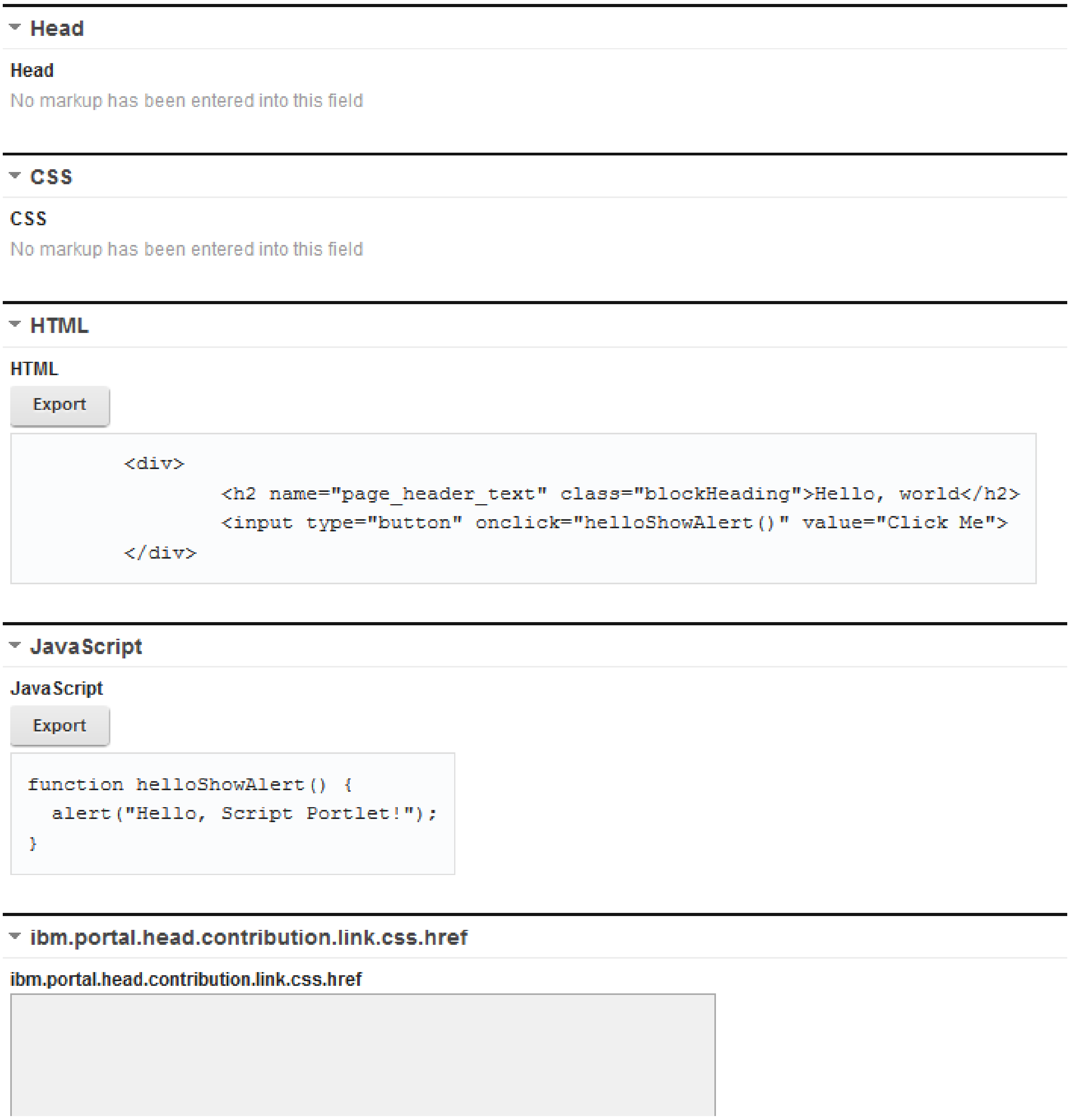


Hello, Script Portlet!

7. Great, we finished our first application.

Some More Details About Script Portlet

All the code you develop in script Portlet is stored within IBM Web Content Management found on the Portal server.



No markup has been entered into this field

No markup has been entered into this field

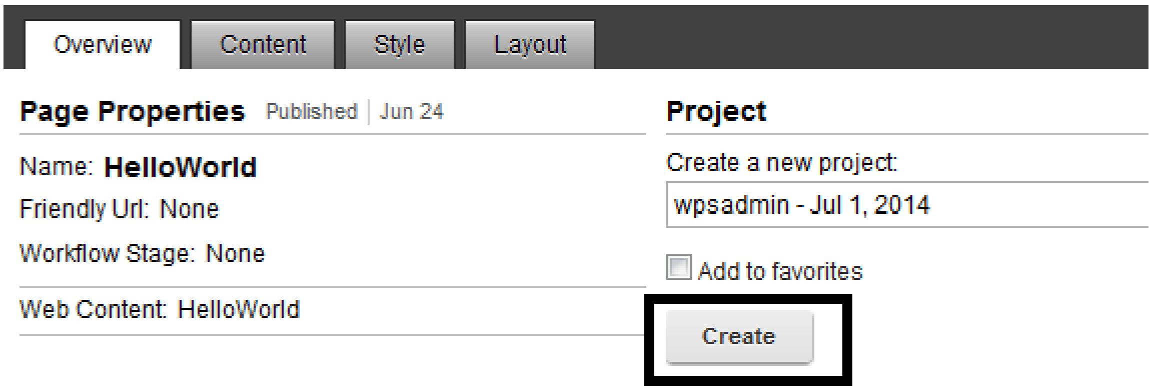
```
        <div>
                <h2 name="page_header_text" class="blockHeading">Hello, world</h2>
                <input type="button" onclick="helloShowAlert()" value="Click Me">
        </div>
```

JavaScript
JavaScript
Export

```
function helloShowAlert() {
  alert("Hello, Script Portlet!");
}
```

ibm.portal.head.contribution.link.css.href
ibm.portal.head.contribution.link.css.href

You could create your own WCM project using managed pages and create a Portlet using Script Portlet. This project can then use WCM workflows to approve the page and Portlet for release and change management control.



Page Properties Published | Jun 24

Name: HelloWorld
Friendly Url: None
Workflow Stage: None
Web Content: HelloWorld

Project

Create a new project: wpsadmin - Jul 1, 2014

Add to favorites

For more details on Managed pages please refer to the IBM Wiki.

What is AngularJS?

AngularJS is like any other JavaScript framework (such as jQuery, Dojo, Prototype.js etc.), but with one key difference. AngularJS is based on a MVC framework. With growing demand to make web applications work more like desktop applications, you need to carefully design your front end views. If your JavaScript is all over the place with <script> tags then it will be impossible to maintain. With other frameworks like jQuery and Dojo, you as a developer should put extra effort to design your application properly. But with AngularJS (much like Spring and Struts) it provides guidelines of how to develop an MVC application.

Say Hello to AngularJS

Let’s develop a simple application before we develop a bigger application.
1. Place the script Portlet on the portal page and open the editor.
2. Add https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-beta.13/angular.min.js as the

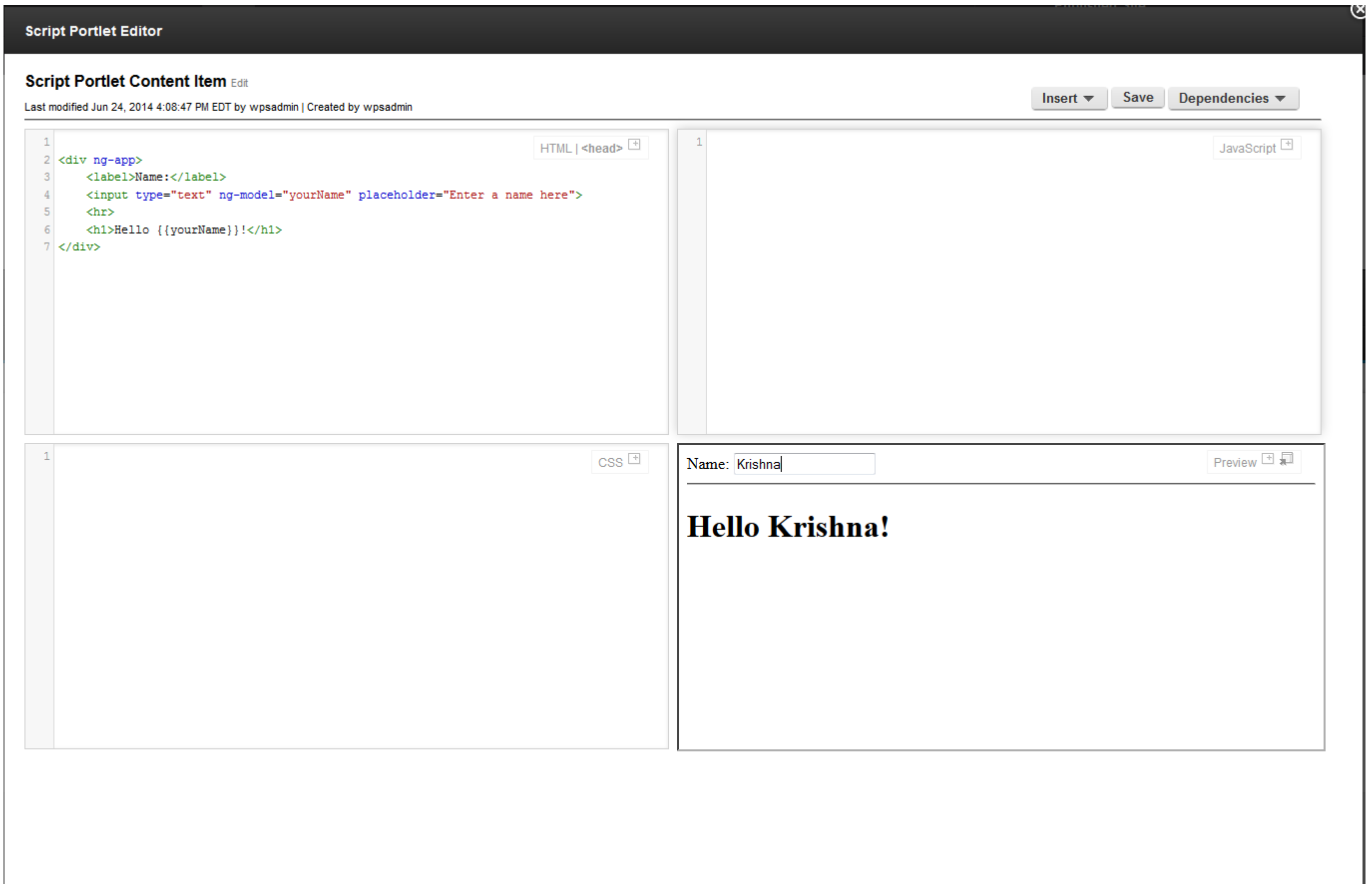
3. Directives are notations letting AngularJS know what that block means. For example “ng-app” lets the framework consider that block as an application. This “ng-app” directive makes AngularJS take care of some important tasks on your behalf. The “ng-app” directive makes sure it initializes the models (ng-model), controllers (ng-controller), views (ng-view), etc. as the application is loaded.

We use {{someModelReference}} format to reference a model, typically meaning a data element. This information is enough to complete our first AngularJS application.

4. Copy the below code in HTML panel.

```
<div ng-app>
    <label>Name:</label>
    <input type="text" ng-model="yourName" placeholder="Enter a name
here">
    <hr>
    <h1>Hello {{yourName}}!</h1>
</div>
```

5. Click the “Save” button. The application is opened in Preview panel. Type your name in the text box.



Last modified Jun 24, 2014 4:08:47 PM EDT by wpsadmin | Created by wpsadmin

```
<div ng-app>
    <label>Name:</label>
    <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <hr>
    <h1>Hello {{yourName}}!</h1>
</div>
```

Name: Krishna

Hello Krishna!

AngularJS Building Blocks

As seen in the above example, with some simple directives we made our code simple and clean, but at the same time we achieved some functionality without any JavaScript code. Now our code looks more readable. Let’s explore the building blocks of this framework.

Module

A module is like an application. You can have one or more modules/applications on a page. In the above application "ng-app" initializes an application. A module can be initialized as ng-app="MyModuleName". Modules contain models, controllers, views, factories, services, etc.

Model

Any framework model represents your data. In AngularJS you can define what a model is. It can be as simple as string or it could be an object. You use "ng-model" to bind your data to HTML controls. You can also reference your model using {{yourObject.attribute}}

Model data is properly scoped when you use "ng-app" directive. Models can be further scoped using "ng-controller". They can also be passed between controllers or views.

Controller

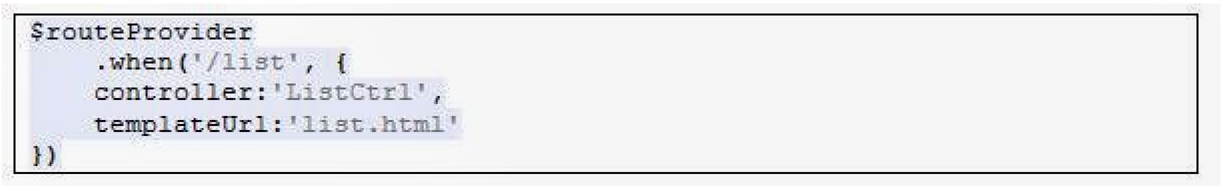
As in any MVC framework, your controller is the bridge between model and view. Controller can do custom logic, invoke a factory/service to call backend and construct the data model and hand it over to appropriate view.

View

View is the markup for the application. "ng-view" defines a view. There can be multiple views in an application. Each view is a HTML template that has place holders for model data and event handlers to hand over control to Controllers.

Routing

Routing is the configuration that defines which view to invoke and which control to be used. In the below example when "/list" is the path, it renders the "list.html" template and passes "ListCtrl" controller.



```
$routeProvider
    .when('/list', {
    controller:'ListCtrl',
    templateUrl:'list.html'
})
```

Factory/Service

Factory/Service is used to make a reusable service. Typically controllers invoke one or more services.

Weather App

Let's develop a sample weather application using some of the features of AngularJS.

Since we have only one view, we don't have to define the view explicitly or configure the routing. You can explore more on views and routing at https://angularjs.org/.

We use a free REST API to develop this application. You need a create a free account at http://www.wunderground.com/weather/api/d/docs?d=data/index&MR=1

This free account will generate a key, which should be used in our sample application.

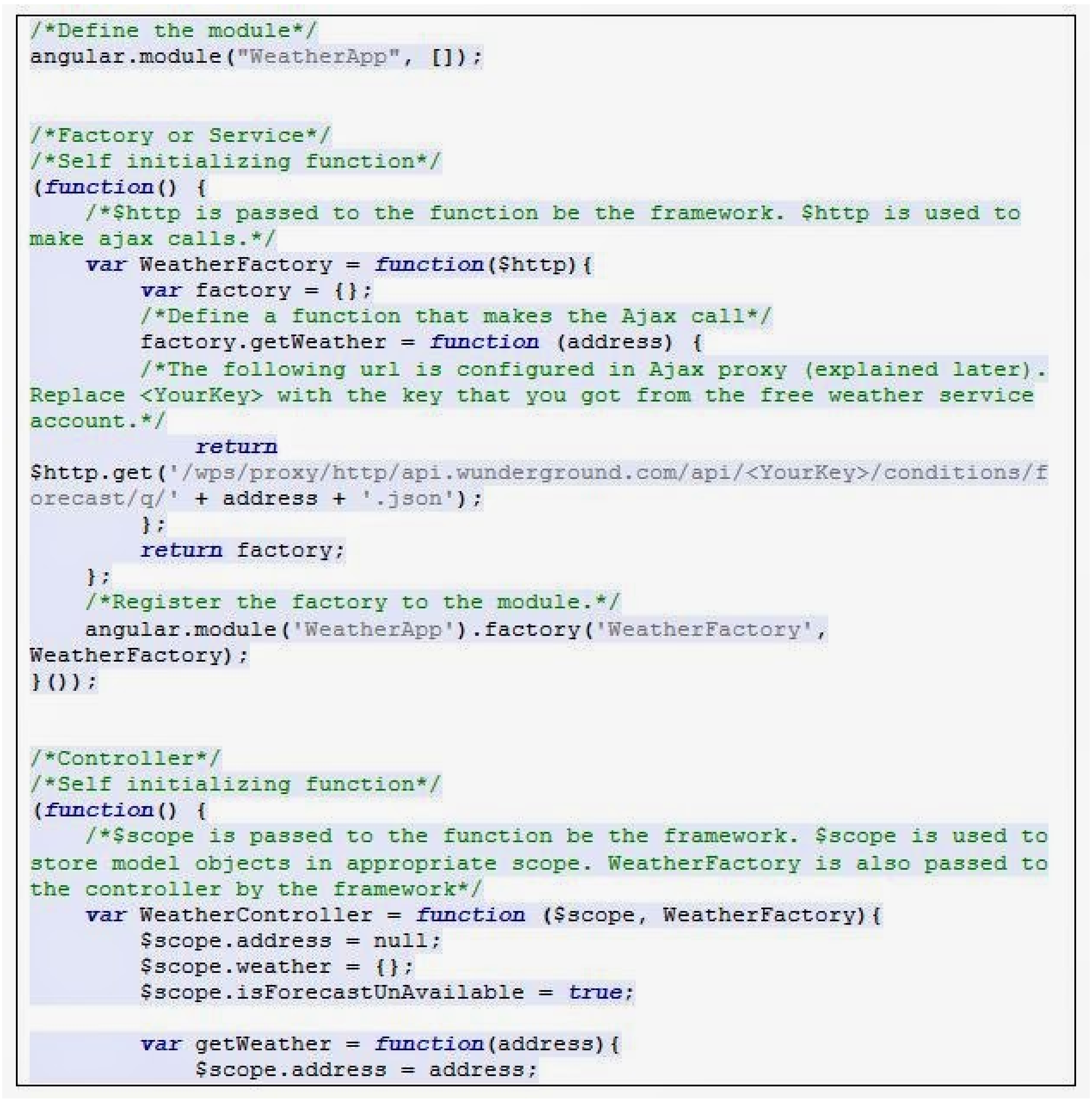
Dependencies

Define the following dependencies.
1. https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-beta.13/angular.min.js
2. http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css

We use bootstrap to control the look and feel. Bootstrap is an open source CSS utility to create responsive applications.

JavaScript

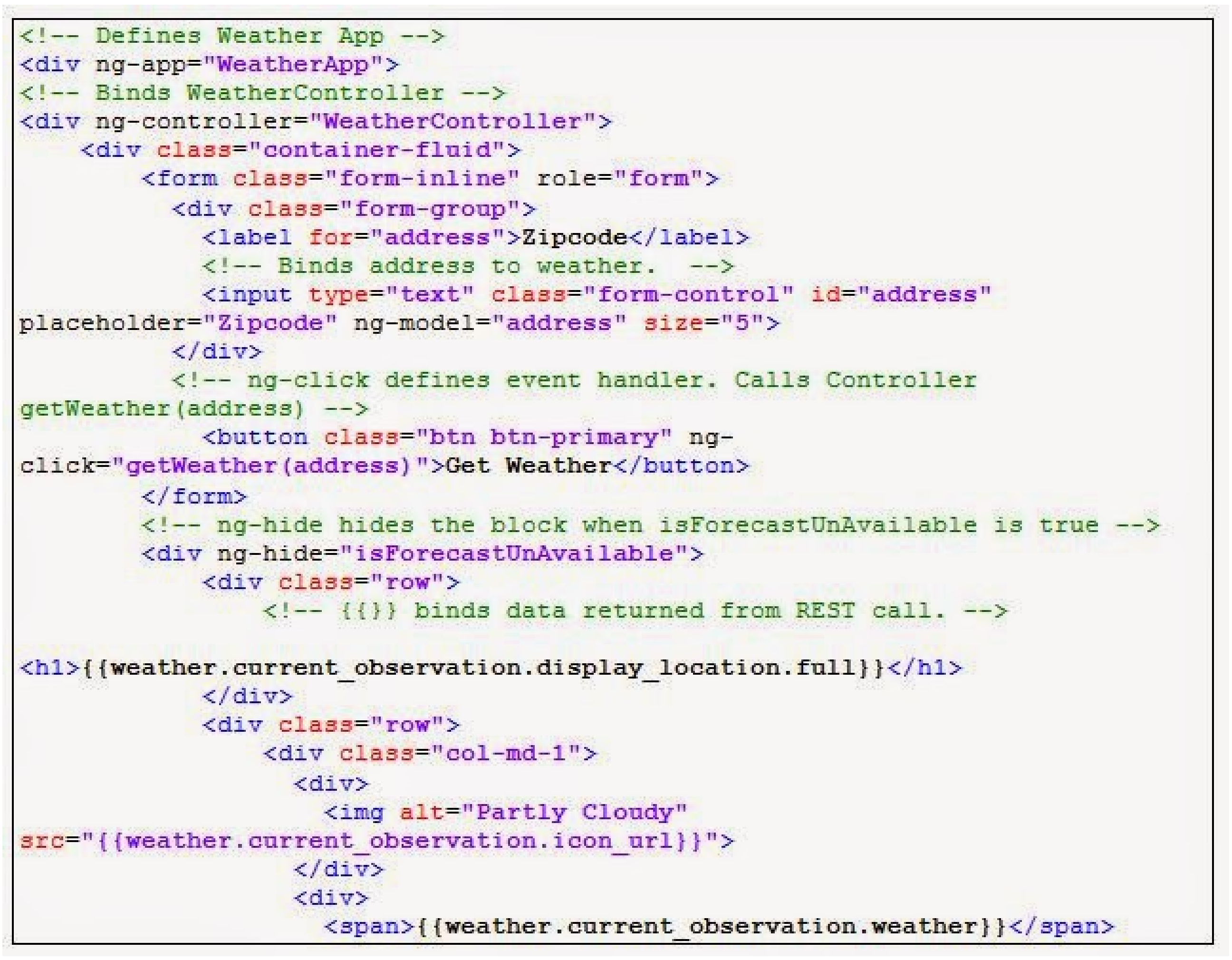
Place the following code in JavaScript panel. Code is explained with inline comments.



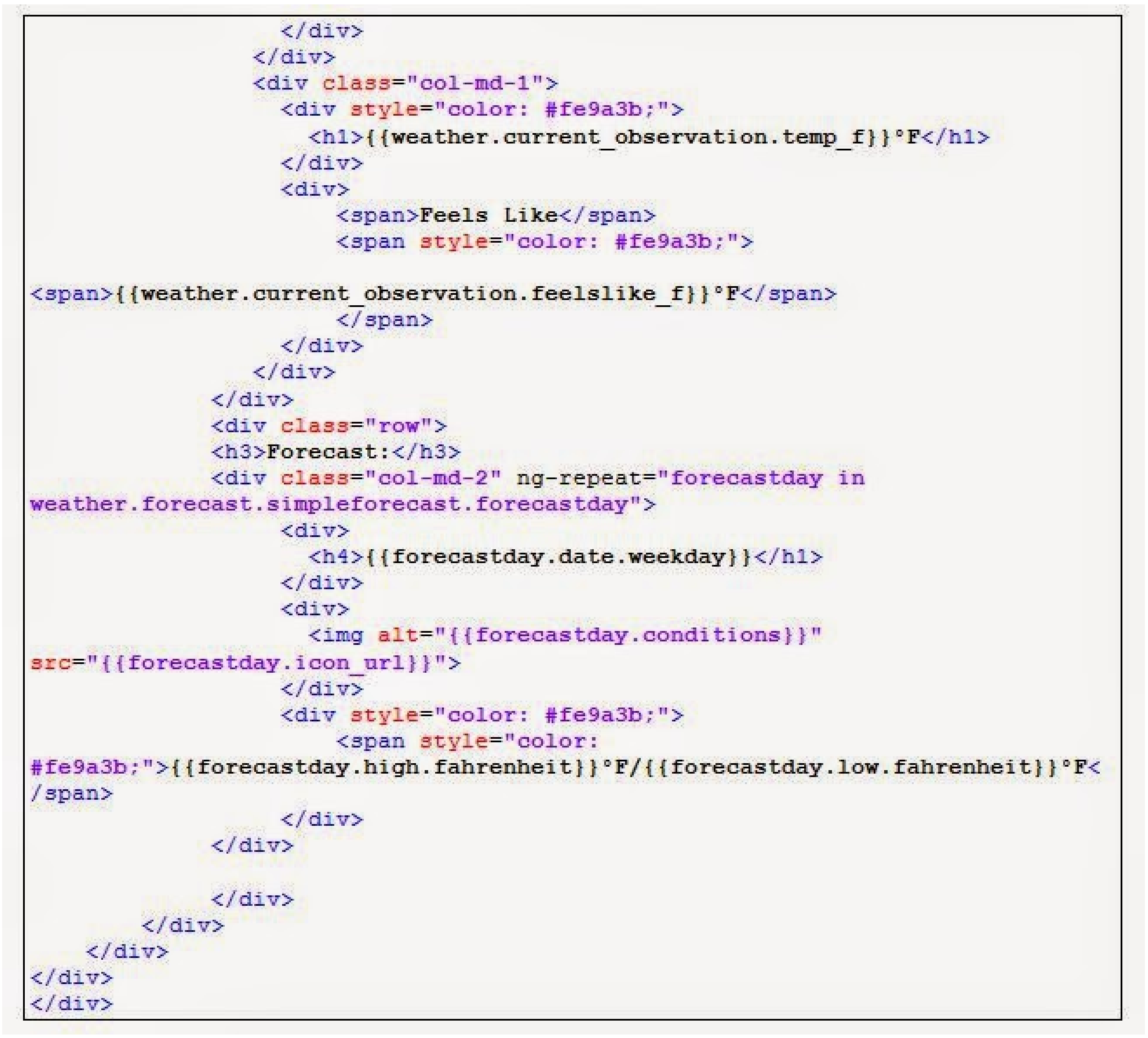
```
/*Define the module*/
angular.module("WeatherApp", []);


/*Factory or Service*/
/*Self initializing function*/
(function() {
    /*$http is passed to the function be the framework. $http is used to
make ajax calls.*/
    var WeatherFactory = function($http){
        var factory = {};
        /*Define a function that makes the Ajax call*/
        factory.getWeather = function (address) {
        /*The following url is configured in Ajax proxy (explained later).
Replace <YourKey> with the key that you got from the free weather service
account.*/
            return
$http.get('/wps/proxy/http/api.wunderground.com/api/<YourKey>/conditions/f
orecast/q/' + address + '.json');
        };
        return factory;
    };
    /*Register the factory to the module.*/
    angular.module('WeatherApp').factory('WeatherFactory',
WeatherFactory);
}());


/*Controller*/
/*Self initializing function*/
(function() {
    /*$scope is passed to the function be the framework. $scope is used to
store model objects in appropriate scope. WeatherFactory is also passed to
the controller by the framework*/
    var WeatherController = function ($scope, WeatherFactory){
        $scope.address = null;
        $scope.weather = {};
        $scope.isForecastUnAvailable = true;

        var getWeather = function(address){
            $scope.address = address;
```

HTML

Place the following HTML code in HTML panel. Code is explained with inline comments.



```
<!-- Defines Weather App -->
<div ng-app="WeatherApp">
<!-- Binds WeatherController -->
<div ng-controller="WeatherController">
    <div class="container-fluid">
        <form class="form-inline" role="form">
          <div class="form-group">
            <label for="address">Zipcode</label>
            <!-- Binds address to weather.  -->
            <input type="text" class="form-control" id="address"
placeholder="Zipcode" ng-model="address" size="5">
          </div>
          <!-- ng-click defines event handler. Calls Controller
getWeather(address) -->
            <button class="btn btn-primary" ng-
click="getWeather(address)">Get Weather</button>
        </form>
        <!-- ng-hide hides the block when isForecastUnAvailable is true -->
        <div ng-hide="isForecastUnAvailable">
            <div class="row">
                <!-- {{}} binds data returned from REST call. -->

<h1>{{weather.current_observation.display_location.full}}</h1>
            </div>
            <div class="row">
                <div class="col-md-1">
                  <div>
                    <img alt="Partly Cloudy"
src="{{weather.current_observation.icon_url}}">
                  </div>
                  <div>
                    <span>{{weather.current_observation.weather}}</span>
```



```
                  </div>
                </div>
                <div class="col-md-1">
                  <div style="color: #fe9a3b;">
                    <h1>{{weather.current_observation.temp_f}}°F</h1>
                  </div>
                  <div>
                      <span>Feels Like</span>
                      <span style="color: #fe9a3b;">

<span>{{weather.current_observation.feelslike_f}}°F</span>
                      </span>
                  </div>
                </div>
             </div>
             <div class="row">
             <h3>Forecast:</h3>
             <div class="col-md-2" ng-repeat="forecastday in
weather.forecast.simpleforecast.forecastday">
                  <div>
                    <h4>{{forecastday.date.weekday}}</h1>
                  </div>
                  <div>
                    <img alt="{{forecastday.conditions}}"
src="{{forecastday.icon_url}}">
                  </div>
                  <div style="color: #fe9a3b;">
                      <span style="color:
#fe9a3b;">{{forecastday.high.fahrenheit}}°F/{{forecastday.low.fahrenheit}}°F<
/span>
                  </div>
             </div>

             </div>
        </div>
    </div>
</div>
</div>
```

Ajax Proxy

We cannot directly make an Ajax call to another domain as browsers do not allow cross site scripting for security reasons. Portal has a feature called “Ajax proxy” that allows this to happen. It just needs some configuration.

Copy “proxy-config.xml” from your Portals “IBM\WebSphere\wp_profile\config\cells\\applications\AJAX Proxy Configuration.ear\deployments\AJAX Proxy Configuration\wp.proxy.config.war\WEB-INF” directory and add the following policy.

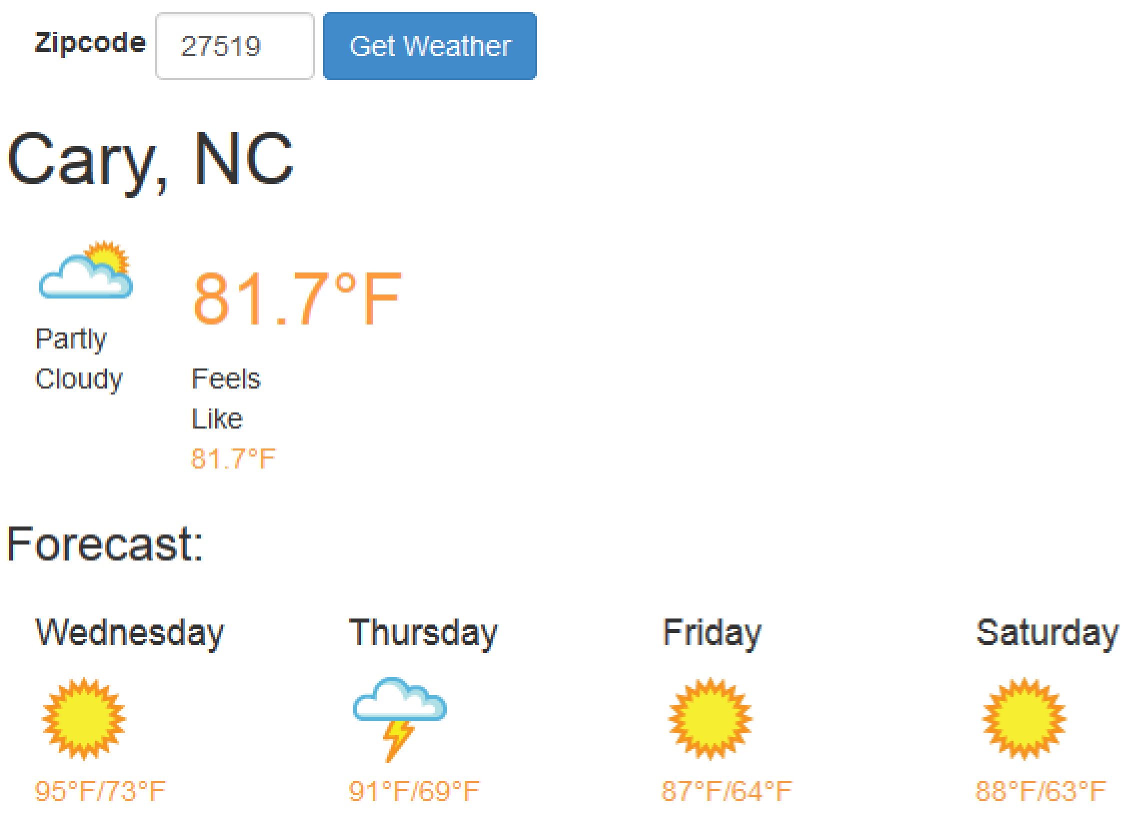
```
    <policy url="http://api.wunderground.com/*" acf="none">
        <actions>
            <method>GET</method>
        </actions>
    </policy>
```

Run the following command:

```
IBM\WebSphere\wp_profile\ConfigEngine>ConfigEngine.bat checkin-wp-proxy-
config -DProxyConfigFileName=C:\IBM\proxy-config.xml
```

C:\IBM\proxy-config.xml is your updated file.

Final output:



Cary, NC

Partly Cloudy

81.7°F

Feels Like 81.7°F

Forecast:

Wednesday 95°F/73°F

Thursday 91°F/69°F

Friday 87°F/64°F

Saturday 88°F/63°F

Features Coming Soon

There are some NEW features that IBM plans to release soon:
1. Upload zip files: In the next version you should be able to upload a zip file with multiple JS, CSS and HTML files.
2. Copy existing scripts: You can copy existing scripts from a different project and

update them as required.
3. WebDav entry point: You should be able to update your files connecting to a WebDav entry point. This should be handy if you use any tool like Brackets to edit your code or store the files in source repository.

References

Greenhouse Script Portlet URL:
https://greenhouse.lotus.com/plugins/plugincatalog.nsf/assetDetails.xsp?action=editDocument&documentId=DDB5C467D991413285257C67002476E0

Script Portlet Sample:
http://openntf.org/main.nsf/project.xsp?r=project/Script%20Portlet%20Samples%20for%20IBM%20WebSphere%20Portal/summary

Script Portlet Webinar:
https://developer.ibm.com/social/2014/05/05/webinar-replay-developing-portlets-using-html-css-javascript/

Script Portlet Presentation:
http://www.slideshare.net/IBMSBT/script-portlet-v30#

Sample Weather Service:
http://www.wunderground.com/weather/api/d/docs?d=data/index

AngularJS:
https://angularjs.org/

AngularJS Video Tutorial:
https://www.udemy.com/angularjs-jumpstart/?couponCode=AngularJSJumpStart40

Fuente
http://www.prolifics.com/in/blog/introduction-script-portlet-using-angularjs

ArquitecturaIBM Consulting