Exercise (Instructions): Setting up a Server using json-server

Exercise Resources

db.json

Objectives and Outcomes

The Node module, *json-server*, provides a very simple way to set up a web server that supports a full-fledged REST API server. We will talk about REST API in the next lesson. It can also serve up static web content from a folder. This lesson will leverage these two features to provide the back-end for your Angular application. In this exercise, you will configure and start a server using *json-server* to enable serving your application data to your Angular application. At the end of this exercise, you will be able to:

- Configure and start a simple server using the json-server module
- Configure your server to serve up static web content stored in a folder named *public*.

Installing json-server

- json-server is a node module, and hence can be installed globally by typing the following at the command prompt:

```
npm install json-server -g
```

If you are using OSX or Linux, use **sudo** at the front of the command. This will install json-server that can be started from the command line from any folder on your computer.

Configuring Server Folder

- At any convenient location on your computer, create a new folder named **json-server**, and move to this folder.
- Download the db.json file provided above to this folder.
- Move to this folder in your terminal window, and type the following at the command prompt to start the server:

```
json-server --watch db.json
```

- This should start up a server at port number 3000 on your machine. The data from this server can be accessed by typing the following addresses into your **browser address bar**:

```
http://localhost:3000/dishes
http://localhost:3000/promotions
```

```
http://localhost:3000/leadership
http://localhost:3000/feedback
```

- Type these addresses into the browser address and see the JSON data being served up by the server. This data is obtained from the db.json file
- The json-server also provides a static web server. Any resources that you put in a folder named **public** in the**json-server** folder above, will be served by the server at the following address:

```
 http://localhost:3000/
```

- Shut down the server by typing **ctrl-C** in the terminal window.

Configuring gulpfile.js to Generate Dist Folder

- In the previous exercises, you configured the **gulpfile.js** to generate the dist folder from the configuration provided in the **menu.html** file. You will now update the **gulpfile.js** to use the **index.html** file for configuration. Update the usemin task in your **gulpfile.js** as follows:

```
gulp.task('usemin',['jshint'], function () {
  return gulp.src('./app/**/*.html')
      .pipe(usemin({
        css:[minifycss(),rev()],
        js: [ngannotate(),uglify(),rev()]
      }))
      .pipe(gulp.dest('dist/'));
});
```

- Also, for the browser-sync task, update the configuration as follows:

```
  browserSync.init(files, {
     server: {
        baseDir: "dist",
        index: "index.html"
     }
  });
```

- Now if you run gulp at the command prompt, it creates the dist folder containing the distribution files.
- Copy the entire contents of the **dist** folder to the **public** folder that you created above.

- Restart the server. Now you can access the website being served up by your server by typinghttp://localhost:3000/ in the browser address bar.

Conclusions

In this exercise, you learnt how to configure and start a simple server using the **json-server** node module. You also learnt how the server can serve up static web content.