**ILP-003 – Tips on Implementation details**

**Always refer to Microsoft Teams for the latest version of the document.**

**Description of the ILP-003 project**

The client side application written in Angular will interface to the server using HTTP protocol to obtain data from the Express server or save the data on to the Express server. The data required for Issues should be managed in a JSON memory object (OR) in a JSON file on Express server.

The implementation will be similar to 16-httpservice and empinfo-app demo code in Angular. (Refer 16-httpservice demo and empinfo-app demo in Angular Webex session).

**Default Port 3000 for Express Server**

In Express Server, 3000 is the default port that is used, though you can configure to whatever port you prefer by setting in the environment variable.

In 16-httpservice demo, the URL is set to <http://localhost:4200/employees> and hence the Express Server had listened at port 4200.

Since the template code generated by “express generator” assumes a default port of 3000, make sure in the client side Angular application also, you use the port 3000.

While implementing ILP-003, you are free to use the default port 3000 or whichever port you prefer, but make sure it is consistent on both Angular and Express code.

**Implementation of Service in Angular**

In Employee Information application demo, the service uses localStorage to store and retrieve data. In ILP-003, you need to interface with Express Server to store and retrieve data. Hence, IssueService need to inject Http service and make Http calls using the Http module of Angular.

Implementing IssueTracker application in Angular involves CRUD operations which can be implemented as per the below guidelines.

**Read Operation**

For R (i.e. Read), you can refer to 16-httpservice demo of Angular, which uses **http.get**.

**Create Operation**

For C (i.e. Create), you need to use **“http.post”**. Refer <https://angular.io/guide/http> to read about how to use **“http.post”**. You can also refer the API documentation - <https://angular.io/api/common/http/HttpClient#post>. You need to provide **“URL”** and **“body”** parameters. **“options”** parameter can be ignored in the current case.

**Note:** In Employee Information demo, while adding a new employee, ID generation is done in the Add Employee Form component. But in ILP-003, ID generation will happen on the server side in the Express server and not in the Angular code.

**Update Operation**

For U (i.e. Update), you need to use **“http.put”** as per REST convention. The usage is exactly similar to **“http.post”** which you can refer to the link - <https://angular.io/guide/http>. You can also refer to the API documentation - <https://angular.io/api/common/http/HttpClient#put>. You need to provide **“URL”** and **“body”** parameters. **“options”** parameter can be ignored in the current case.

**Delete Operation**

For D (i.e. Delete), you need to use **“http.delete”** as per REST convention. The usage is similar to **“http.get”.** You can also refer to the API documentation - <https://angular.io/api/common/http/HttpClient#delete>.

**Invoking subscribe for the CRUD operations and handling successful execution - IMPORTANT**

If you refer 16-httpservice demo of Angular, the Angular service code of “getEmployees” will return an Observable. This will be immediately returned but the actual employees data may take longer time to arrive as these are asynchronous network requests.

For Add, Update and Delete operations also, it has to be implemented in the same way as done in 16-httpservice demo of Angular for getEmployees (i.e. Read - http.get). This means the service methods will return the Observable and **“subscribe”** need to be called in the Angular UI component code.

In case of Add and Update operations, in the Angular UI Component code, **“subscribe”** need to be invoked and on success **(i.e. the first function parameter of subscribe)**, **“router.navigate”** should be called. The second function parameter of subscribe is to handle errors.

In case of Delete operation, in the Angular UI Component code, **“subscribe”** need to be invoked and on success **(i.e. the first function parameter of subscribe)**, there are **2 ways** of handling depending upon how Express code of **“router.delete”** is implemented.

1. If the Express code of “delete” deletes the item and **returns null**, in the Angular side, the service method to get all items (e.g. getAllEmployees) should be called and set the employees property in the same way as done in 16-httpservice demo of Angular. The second function parameter of subscribe is to handle errors. This method involves 2 network calls.
2. If the Express code of “delete” deletes the item and **returns the remaining items**, in the Angular side, the remaining items received can be read and set and since a change has occurred, due to data binding, the item deleted will be removed from the UI and remaining items will be displayed. This method involves 1 network call.

**Building a production version of Angular application**

To build a production version of Angular application, you need to issue the command **“ng build –prod”** as already explained in “ILP-003\_OutputAndCodeSubmissiondetails\_v0.1docx”.

In Employee Information Demo, while doing “ng build –prod”, you might get compilation errors because some variables like “name”, “location”, “email”, “mobile” was not declared in “addemployee-form.component.ts”. The debug build did not generate any errors but production build is not passing through. Hence, these variables should be declared in “addemployee-form.component.ts” as given below.

name: string;

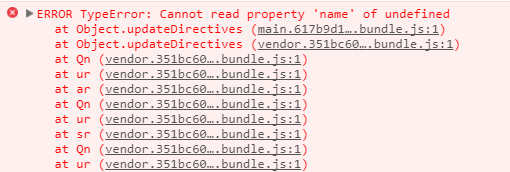
location: string;

email: string;

mobile: string;

You may face a similar problem while doing a production build for ILP-003 IssueTracker Angular application also. **Hence, take care of it if such an error occurs.**

In Employee Information Demo, while doing Edit of an employee in production version, although the application works properly, you **might get** errors in JavaScript console as shown in the screenshot below. The debug build did not show any such errors in JavaScript console while doing an Edit.



To fix the error, the “employee” property should be initialized as given below in “editemployee-form.component.ts”.

**employee: any = {};**

You may face a similar problem while doing a production build for ILP-003 IssueTracker Angular application also. **Hence, take care of it if such an error occurs.**

**Implementation of route handling in Express**

The Comment Application demonstrated in the Express Webex session reads the Comment JSON data and passes the JSON data to Jade template which is converted to HTML by the Jade engine. This is done by invoking **“res.render”** in comment.js.

In ILP-003, the JSON data read should be directly transferred to the client side Angular application by invoking **“res.json”**. Please note that for all cases of route handling (Create, Read, Update and Delete), **“res.json”** only should be used.

**Update and delete route handling**

In Comment Application demo, for update and delete, router.post('/edit/:id’) and router.get('/delete/:id') respectively has been used. But as per REST convention, it is recommended to use ‘PUT’ for update and ‘DELETE’ for delete. Hence, in ILP-003, use **router.put**('/edit/:id’) and **router.delete**('/delete/:id').

In client side Angular application, it is for this reason it is mentioned that “http.put” and “http.delete” should be used for Update and Delete.

**Navigating to the main page after Add, Update and Delete operations**

In Comment Application demo, after adding, updating and deleting a comment, we invoke **res.redirect('/comment').** But in ILP-003, we should **NOT invoke** res.redirect() .

**Reason:** Express Server only provides the required data to the client side Angular application. Client side routing and Navigation will be handled by the client side Angular application.

**Running Angular, Express applications independently in Development Phase**

In Angular Webex session, when demonstrating 16-httpservice demo, Angular production build was generated in “dist” directory. Then Express code and Angular application resided in the same directory and “node server.js” was invoked to start Express server and Angular application was also served.

If you refer **“ILP-003\_OutputAndCodeSubmissiondetails\_vx.y.docx”**, a similar approach has been mentioned (i.e. Angular production build is done and the files generated in “dist” directory is copied to “public” directory of Express and it becomes an integrated Express Angular code. **This is fine to do for final production build.** But when you are in development phase, every time doing a production build of Angular and copying in Express folder is highly unproductive. So, in developmental phase we can do it in the following way.

1. Angular CLI application can be run in the normal way as “ng serve” and access through <http://localhost:4200>. Here 4200 is the default port.
2. Express application can also be run in the normal way as “npm start” and by default it will listen at port 3000.
3. Angular CLI application will make “http” calls to Express when it requires data. **But there is a problem here.** Angular application is running in its own developmental server listening at port 4200 and Express is another server listening at port 3000. Hence, the calls from Angular application will not be serviced by Express server because communication is now happening between two different servers and it will not be allowed as per the error shown in the screenshot below.



**How to make it work such that access is allowed**

**(IMPORTANT: Do not include the below code snippets in production build in real projects)**

There are 2 different methods of making it work. I will explain below one of the method which works straightaway in your current code with just 2 lines of addition in Express code.

Add the following 2 lines in the **“app.js”** code of Express.

**var cors = require('cors');** // Include this line among your other declarations.

**app.use(cors());** // Include this line where you find a list of use statements **but it should be put before “app.use” of route handling statements (i.e**. before app.use('/', routes); etc).

**Note:** Install “cors” module by giving **“npm install cors”** in Express project folder.

Once you write the above code in Express, you can independently run the client and server code.

To run Angular application “ng serve”.

To run Express application “npm start”.

In the browser, you will only give <http://localhost:4200>.