Starting with Angular application to read XML file and display on HTML using Eclipse

Versions used

java version "11.0.5" 2019-10-15 LTS

Java(TM) SE Runtime Environment 18.9 (build 11.0.5+10-LTS)

Eclipse : Eclipse IDE for Java Developers

Version: 2019-09 R (4.13.0)

Build id: 20190917-1200

Angular CLI version 8.3.25

Node.js : 10.15.0

NPM : 6.4.1

Installing Angular plugin for eclipse -

it will also install other necessary programms such as npm and node if you do not have them already installed on your system.

In Eclipse

Go to Help -> Eclipse marketplace

Search for Angular

click on install for Angular CLI version 8.3.25

select new theme if you wish to

press finish.

restart eclipse.

Now you will get terminal for angular projects called Terminal+.

It's time to create new project.

Go to File -> new-> Angular project

Write name of the project - > AngularInvoice

let angular version, node version and npm version as mentioned, initialize project contents with

click on new Angular application

click next and you will see the commands which will be executed to create your project-

The following commands will be executed to create your project.:

npm install @angular/cli@8.3.25

ng new AngularInvoice --skip-git --interactive=false

npm install --save-dev angular-ide

click on finish

new project will be created and node and npm will get installed for the project.

copy code from

https://www.c-sharpcorner.com/article/reading-xml-file-in-angular-8/

create new xml file called database.xml in assets folder of the project and copy paste below code

1. <?xml version="1.0" encoding="UTF-8"?>
2. <Employee>
3. <emp>
4. <id>1</id>
5. <name>Faisal</name>
6. <gender>Male</gender>
7. <mobile>514545</mobile>
8. </emp>
9. <emp>
10. <id>2</id>
11. <name>Bhavdip</name>
12. <gender>Male</gender>
13. <mobile>5431643</mobile>
14. </emp>
15. <emp>
16. <id>3</id>
17. <name>Irshad</name>
18. <gender>Male</gender>
19. <mobile>43265436</mobile>
20. </emp>
21. <emp>
22. <id>4</id>
23. <name>Keyur</name>
24. <gender>Male</gender>
25. <mobile>5435431</mobile>
26. </emp>
27. <emp>
28. <id>5</id>
29. <name>Tabish</name>
30. <gender>Male</gender>
31. <mobile>432656</mobile>
32. </emp>
33. </Employee>

open terminal+ and install "timers" npm package.

1. npm install timers

This package is necessary for reading an XML file with xml2js package.

Open the index.html present at root folder and add a reference for Bootstrap and jQuery

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>ReadXmlAngular8</title>
6. <**base** href="/">
7. <meta name="viewport" content="width=device-width, initial-scale=1">
8. <link rel="icon" type="image/x-icon" href="favicon.ico">
9. <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
10. <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script> <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script> <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
11. </head>
12. <body>
13. <app-root></app-root>
14. </body>
15. </html>

Open the app.component.ts file and add the following code in it.

1. import { Component } from '@angular/core';
2. import xml2js from 'xml2js';
3. import { HttpClient, HttpHeaders } from '@angular/common/http';
4. @Component({
5. selector: 'app-root',
6. templateUrl: './app.component.html',
7. styleUrls: ['./app.component.css']
8. })
9. export **class** AppComponent {
10. title = 'read-xml-angular8';
11. **public** xmlItems: any;
12. constructor(**private** \_http: HttpClient) { **this**.loadXML(); }
13. loadXML() {
14. **this**.\_http.**get**('/assets/database.xml',
15. {
16. headers: **new** HttpHeaders()
17. .**set**('Content-Type', 'text/xml')
18. .append('Access-Control-Allow-Methods', 'GET')
19. .append('Access-Control-Allow-Origin', '\*')
20. .append('Access-Control-Allow-Headers', "Access-Control-Allow-Headers, Access-Control-Allow-Origin, Access-Control-Request-Method"),
21. responseType: 'text'
22. })
23. .subscribe((data) => {
24. **this**.parseXML(data)
25. .then((data) => {
26. **this**.xmlItems = data;
27. });
28. });
29. }
30. parseXML(data) {
31. **return** **new** Promise(resolve => {
32. var k: **string** | number,
33. arr = [],
34. parser = **new** xml2js.Parser(
35. {
36. trim: **true**,
37. explicitArray: **true**
38. });
39. parser.parseString(data, function (err, result) {
40. var obj = result.Employee;
41. **for** (k **in** obj.emp) {
42. var item = obj.emp[k];
43. arr.push({
44. id: item.id[0],
45. name: item.name[0],
46. gender: item.gender[0],
47. mobile: item.mobile[0]
48. });
49. }
50. resolve(arr);
51. });
52. });
53. }
54. }

if you have given other name to your xml file, change line 14 in above code and replace the name.   **this**.\_http.**get**('/assets/database.xml',

Here is the code for app.component.html file.

1. <div **class**="container">
2. <table **class**="table table-bordered table-hover">
3. <tr>
4. <th>Id</th>
5. <th>Name</th>
6. <th>Gender</th>
7. <th>Mobile</th>
8. </tr>
9. <tr \*ngFor="let item of xmlItems">
10. <td>{{item.id}}</td>
11. <td>{{item.name}}</td>
12. <td>{{item.gender}}</td>
13. <td>{{item.mobile}}</td>
14. </tr>
15. </table>
16. </div>

Finally, add the HttpClientModule reference in the app.module.ts file and that’s it.

1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { AppRoutingModule } from './app-routing.module';
4. import { AppComponent } from './app.component';
5. import { HttpClientModule } from '@angular/common/http';
6. @NgModule({
7. declarations: [
8. AppComponent
9. ],
10. imports: [
11. BrowserModule,
12. AppRoutingModule,
13. HttpClientModule
14. ],
15. providers: [],
16. bootstrap: [AppComponent]
17. })
18. export **class** AppModule { }

Error occured

[Cannot find module './app-routing.module'](https://stackoverflow.com/questions/40042602/angular2-tutorial-tour-of-heroes-cannot-find-module-app-routing-module)

Solution

We got this error because

ng **new**  AngularInvoice --routing

routing option was missing when we first created the project.

https://shermandigital.com/blog/configure-routing-in-an-angular-cli-project/

Configure Routing In An Angular CLI Project

To add routing, follow the steps below after [generating an Angular CLI project](https://github.com/angular/angular-cli#generating-and-serving-an-angular-project-via-a-development-server).

1. Open a terminal in the root directory of the application and create a dashboard component: ng g component dashboard
2. Open index.html and ensure there is a <base href="/"> tag at the top of the head section.
3. In the root directory of the application, run the command: ng g module app-routing
4. Open app-routing.module.ts and paste the following configuration:

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from '../dashboard/dashboard.component'

const routes: Routes = [

{

path: '',

component: DashboardComponent,

},

];

@NgModule({

imports: [

RouterModule.forRoot(routes)

],

exports: [

RouterModule

],

declarations: []

})

export class AppRoutingModule { }

We export the RouterModule to indicate that our module has a dependency on this module.

5.Open app.module.ts and add the AppRoutingModule to the imports configuration section.

...

import { AppRoutingModule } from './app-routing/app-routing.module';

...

@NgModule({

...

imports: [

BrowserModule,

FormsModule,

HttpModule,

AppRoutingModule,

],

...

})

export class AppModule { }

6. In app.component.html, add <router-outlet></router-outlet> in the space where views should appear.

7. Save all files. The dashboard view is now displayed within the app component.

8. Now add other routes in app-routing.module.ts.

...

const routes: Routes = [

{

path: '/my-new-route',

component: MyNewRouteComponent,

},

{

path: '',

component: DashboardComponent,

},

];

...

All this will create new server named AngularInvoice which you can see by clicking on servers tab from left side of the panel.

right click on the name and click on start server. This will execute the command in the project folder on Terminal+

'node\_modules/angular-ide/bin/ng' 'serve' 'AngularInvoice' --port=4200 --live-reload=true

After compiling successfully , you will get the message on Terminal+.

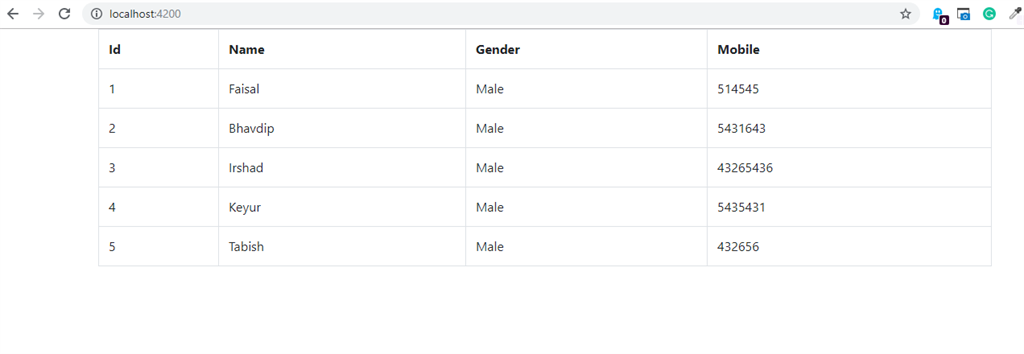
Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200

｢wdm｣: Compiled successfully.

Now all you have to do is open any web browser and go to the link

http://localhost:4200/

And the output will be



adding upload file component

for reference related to uploading multiple files

https://www.ahmedbouchefra.com/angular-tutorial-example-upload-files-with-formdata-httpclient-rxjs-and-material-progressbar/

[Angular Material](https://material.angular.io/) provides Material Design components that allow developers to create professional UIs.

Go to your terminal, and run the following command from the root folder of your project:

ng add @angular/material

You’ll be prompted to choose a theme, let’s go with **Indigo/Pink**.

For the other questions — **Set up HammerJS for gesture recognition?** and **Set up browser animations for Angular Material?** Press **Enter** in your keyboard to choose the default answers.

Next, open the src/styles.css file and add a theme:

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

*.hidden* {

visibility: *hidden*;

width: 1px;

height: 1px;

}

add button to form with onclick function

<button (click)=*"handleClick()"*>Click Me </button>

<a (click)=*"handleClick()"* href=*"javascript:undefined"*>

<i class=*"la la-upload"*></i>

</a>

<input class=*"hidden"* type=*"file"* id=*"upload-file"* name=*"upload-file"* accept=*".xml"* ngf-max-size=*"20MB"* (change)=*"addAttachment($event)"*>

**loadXML**(mesg) {

alert(mesg.name);

**this**.\_http.get('/assets/'+ mesg.name,

add following code to app.component.ts ->AppComponent class

**handleClick**() {

document.getElementById('upload-file').click();

}

**addAttachment**(fileInput: any) {

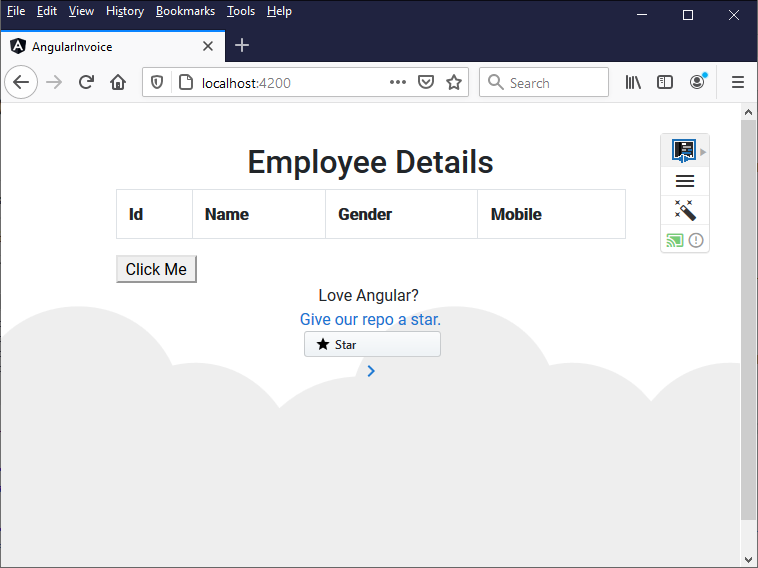
**const** fileReaded = fileInput.target.files[0];

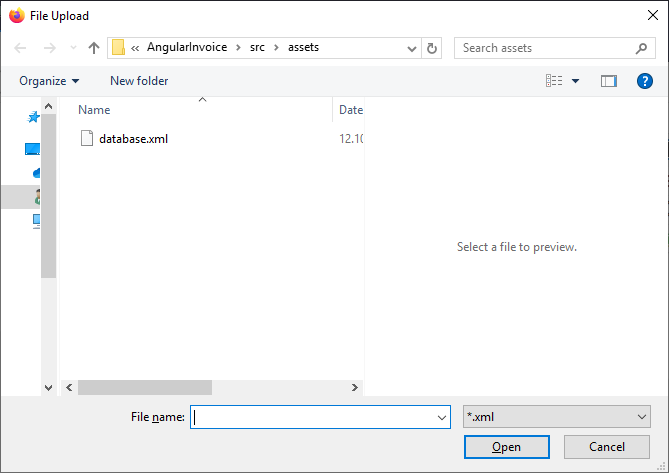
**this**.loadXML(fileReaded);

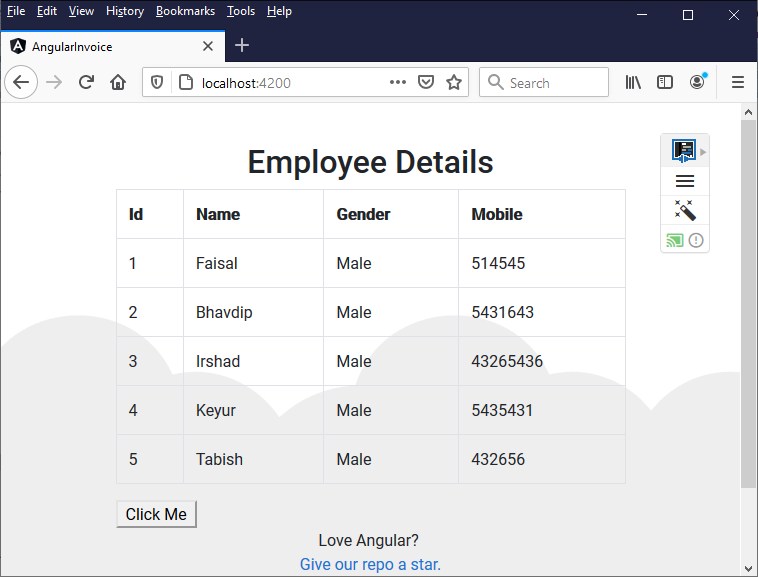
// handle the rest

}

save the changes and server will automatically compile it the browser will be refreshed to reflect changes.







Deploy to GitHub page (reference https://angular.io/guide/deployment)

Another simple way to deploy your Angular app is to use [GitHub Pages](https://help.github.com/articles/what-is-github-pages/).

1. You need to [create a GitHub account](https://github.com/join) if you don't have one, and then [create a repository](https://help.github.com/articles/create-a-repo/) for your project. Make a note of the user name and project name in GitHub.
2. Build your project using Github project name, with the Angular CLI command [ng build](https://angular.io/cli/build) and the options shown here:

$ ng build --prod --output-path docs --base-href /AngularInvoice/

completed with warning

WARNING in ./node\_modules/xml2js/node\_modules/sax/lib/sax.js

Module not found: Error: Can't resolve 'stream' in 'C:\Users\shruti\eclipse-workspace\AngularInvoice\node\_modules\xml2js\node\_modules\sax\lib'

Solution -

For Angular to recognise xml2js, the following steps are required:

1. Add the timers-browserify node module using "

npm install timers-browserify" amd

npm install --save stream timers

build again and there are no warnings anymore.

1. When the build is complete, make a copy of docs/index.html and name it docs/404.html.
2. Commit your changes and push.
3. On the GitHub project page, configure it to [publish from the docs folder](https://help.github.com/articles/configuring-a-publishing-source-for-github-pages/#publishing-your-github-pages-site-from-a-docs-folder-on-your-master-branch).

On GitHub, navigate to your site's repository.

Under your repository name, click **Settings**

Under "GitHub Pages", use the **None** or **Branch** drop-down menu and select a publishing source.

Optionally, use the drop-down menu to select a folder for your publishing source.

Click **Save**.

You can see your deployed page at https://<user\_name>.github.io/<project\_name>/.