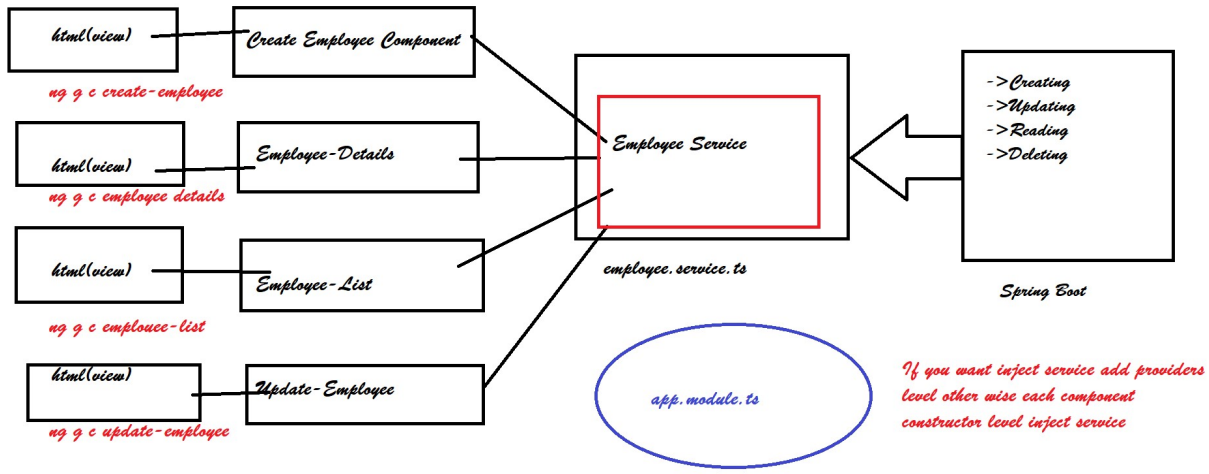
Angular + Spring Boot + MySQL CURD Application



2. Second Part: Angular 9 Client App Development

I assume that you have installed **Node.js**. Now, we need to check the **Node.js** and NPM versions. Open the terminal or Node command line then type these commands.

```
C:\Angular>node -v
v10.15.3

C:\Angular>npm -v
6.9.0
```

Install the latest version of Angular CLI 9

To install or update Angular 9 CLI, type this command in the terminal:

```
npm install -g @angular/cli@9.0.0-rc.7
```

Note that we are installing Angular CLI 9.

Now, let's check the latest version of Angular CLI:

```
C:\Angular\angular9-springboot-crud-tutorial>ng --version
     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|      \____|_____|___|
                |___/


Angular CLI: 9.0.0-rc.7
Node: 10.15.3
OS: win32 x64

Angular:
...
Ivy Workspace:
```

Angular + Spring Boot + MySQL CURD Application

```
Package                      Version
------------------------------------------------------
@angular-devkit/architect    0.900.0-rc.7
@angular-devkit/core         9.0.0-rc.7
@angular-devkit/schematics   9.0.0-rc.7
@schematics/angular          9.0.0-rc.7
@schematics/update           0.900.0-rc.7
rxjs                         6.5.3
```

Create an Angular 9 App using Angular CLI 9

The **Angular CLI** is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications.

If you are new to Angular CLI then check out official documentation at **https://cli.angular.io**.

Let's use the below command to generate an Angular 9 Client application. We name this project as "angular9-springboot-client".

```
ng new angular9-springboot-client
```

Identify Components, Services, and Modules

Let's list out what are **components**, **services**, and **modules** we are going to create in this application. We will use Angular CLI to generate components, services because Angular CLI follows best practices and saves much of time.

Components

- create-employee
- employee-list
- employee-details

Services

- employee.service.ts - Service for Http Client methods

Modules

- FormsModule
- HttpClientModule
- AppRoutingModule

Employee Class (Typescript class)

- employee.ts: class Employee (id, firstName, lastName, emailId)

In this next step, we will generate these components, classes, and services using **Angular CLI.**

Angular + Spring Boot + MySQL CURD Application

Create Service & Components using Angular CLI

Let's auto-generate the service and components using **Angular CLI**. Change your project directory to **angular9-springboot-client\src\app** and run the following commands:

```
- ng g s employee
- ng g c create-employee
- ng g c employee-details
- ng g c employee-list
```

Here is complete command and output for your reference:

```
C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client>cd src/app

C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client\src\app>ng g s employee
CREATE src/app/employee.service.spec.ts (343 bytes)
CREATE src/app/employee.service.ts (137 bytes)

C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client\src\app>ng g c create-employee
CREATE src/app/create-employee/create-employee.component.html (34 bytes)
CREATE src/app/create-employee/create-employee.component.spec.ts (685 bytes)
CREATE src/app/create-employee/create-employee.component.ts (304 bytes)
CREATE src/app/create-employee/create-employee.component.css (0 bytes)
UPDATE src/app/app.module.ts (509 bytes)

C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client\src\app>ng g c employee-list
CREATE src/app/employee-list/employee-list.component.html (32 bytes)
CREATE src/app/employee-list/employee-list.component.spec.ts (671 bytes)
CREATE src/app/employee-list/employee-list.component.ts (296 bytes)
CREATE src/app/employee-list/employee-list.component.css (0 bytes)
UPDATE src/app/app.module.ts (617 bytes)

C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client\src\app>ng g c employee-list
ERROR! src/app/employee-list/employee-list.component.html already exists.
ERROR! src/app/employee-list/employee-list.component.spec.ts already exists.
ERROR! src/app/employee-list/employee-list.component.ts already exists.
ERROR! src/app/employee-list/employee-list.component.css already exists.
The Schematic workflow failed. See above.

C:\Angular\angular9-springboot-crud-tutorial\angular9-springboot-client\src\app>ng g c employee-details
CREATE src/app/employee-details/employee-details.component.html (35 bytes)
CREATE src/app/employee-details/employee-details.component.spec.ts (692 bytes)
CREATE src/app/employee-details/employee-details.component.ts (308 bytes)
CREATE src/app/employee-details/employee-details.component.css (0 bytes)
UPDATE src/app/app.module.ts (737 bytes)
```

Integrate JQuery and Bootstrap with Angular

Use NPM to download **Bootstrap** & **JQuery**. **Bootstrap** and **jQuery** will be installed into the **node_modules** folder.

```
npm install bootstrap jquery --save
```

Configure installed **Bootstrap** & **JQuery** in an `angular.json` file:

```
...

"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "node_modules/jquery/dist/jquery.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
```

```
]

...
```

If bootstrap won't work then try to import bootstrap CSS in style.css like:

```
/* You can add global styles to this file, and also import other style files */

@import '~bootstrap/dist/css/bootstrap.min.css';

.footer {
    position: absolute;
    bottom: 0;
    width:100%;
    height: 70px;
    background-color: blue;
    text-align: center;
    color: white;
}
```

Let's discuss each of the above generate components and service files and we will customize it as per our requirement.

Table of contents

1. Create an Employee class
2. Employee Service
3. Creating Employee List Template and Component
4. Create Add Employee Template and Component
5. Update Employee Template and Component
6. Create View Employee Details Template and Component

1. Create an Employee Model (TypeScript)

Path - src/app/employee.ts

Before defining the **EmployeeListComponent**, let's define an **Employee** class for working with employees. create a new file `employee.ts` inside `src/app` folder and add the following code to it -

```
export class Employee {
    id: number;
    firstName: string;
    lastName: string;
    emailId: string;
    active: boolean;
}
```

2. Creating Employee List Template and Component

Employee List Component

Path - src/app/employee-list/employee-list.component.ts

Let's create **EmployeeListComponent** component which will be used to display a list of employees, create a new employee, and delete an employee.

Angular + Spring Boot + MySQL CURD Application

Update/remove the content of `employee-list.component.ts` inside `src/app` directory and add the following code to it -

```
import { EmployeeDetailsComponent } from '../employee-details/employee-details.component';
import { Observable } from "rxjs";
import { EmployeeService } from "../employee.service";
import { Employee } from "../employee";
import { Component, OnInit } from "@angular/core";
import { Router } from '@angular/router';

@Component({
  selector: "app-employee-list",
  templateUrl: "./employee-list.component.html",
  styleUrls: ["./employee-list.component.css"]
})
export class EmployeeListComponent implements OnInit {
  employees: Observable<Employee[]>;

  constructor(private employeeService: EmployeeService,
    private router: Router) {}

  ngOnInit() {
    this.reloadData();
  }

  reloadData() {
    this.employees = this.employeeService.getEmployeesList();
  }

  deleteEmployee(id: number) {
    this.employeeService.deleteEmployee(id)
      .subscribe(
        data => {
          console.log(data);
          this.reloadData();
        },
        error => console.log(error));
  }

  employeeDetails(id: number){
    this.router.navigate(['details', id]);
  }
}
```

Employee List Template

Path - src/app/employee-list/employee-list.component.html

Add `employee-list.component.html` file with the following code to it -

```
<div class="panel panel-primary">
  <div class="panel-heading">
    <h2>Employee List</h2>
  </div>
  <div class="panel-body">
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Firstname</th>
          <th>Lastname</th>
          <th>Email</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let employee of employees | async">
          <td>{{employee.firstName}}</td>
          <td>{{employee.lastName}}</td>
          <td>{{employee.emailId}}</td>
          <td><button (click)="deleteEmployee(employee.id)" class="btn btn-danger">Delete</button>
```

```
            <button (click)="employeeDetails(employee.id)" class="btn btn-info" style="margin-left:
10px">Details</button>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

3. Create Add Employee Template and Component

Create Employee Component

Path - src/app/create-employee/create-employee.component.ts

CreateEmployeeComponent is used to create and handle a new employee form data. Add the following code to it -

```
import { EmployeeService } from '../employee.service';
import { Employee } from '../employee';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-create-employee',
  templateUrl: './create-employee.component.html',
  styleUrls: ['./create-employee.component.css']
})
export class CreateEmployeeComponent implements OnInit {

  employee: Employee = new Employee();
  submitted = false;

  constructor(private employeeService: EmployeeService,
    private router: Router) { }

  ngOnInit() {
  }

  newEmployee(): void {
    this.submitted = false;
    this.employee = new Employee();
  }

  save() {
    this.employeeService.createEmployee(this.employee)
      .subscribe(data => console.log(data), error => console.log(error));
    this.employee = new Employee();
    this.gotoList();
  }

  onSubmit() {
    this.submitted = true;
    this.save();
  }

  gotoList() {
    this.router.navigate(['/employees']);
  }
}
```

Create Employee Template

Path - src/app/create-employee/create-employee.component.html

Angular + Spring Boot + MySQL CURD Application

The `create-employee.component.html` shows the add employee HTML form. Add the following code to it -

```
<h3>Create Employee</h3>
<div [hidden]="submitted" style="width: 400px;">
  <form (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="name">First Name</label>
      <input type="text" class="form-control" id="firstName" required [(ngModel)]="employee.firstName" name="firstName">
    </div>

    <div class="form-group">
      <label for="name">Last Name</label>
      <input type="text" class="form-control" id="lastName" required [(ngModel)]="employee.lastName" name="lastName">
    </div>

    <div class="form-group">
      <label for="name">First Name</label>
      <input type="text" class="form-control" id="emailId" required [(ngModel)]="employee.emailId" name="emailId">
    </div>

    <button type="submit" class="btn btn-success">Submit</button>
  </form>
</div>

<div [hidden]="!submitted">
  <h4>You submitted successfully!</h4>
  <!-- <button class="btn btn-success" (click)="newEmployee()">Add</button> -->
</div>
```

4. Update Employee Template and Component

Let's create update employee component with following Angular CLI command:

```
> ng g c update-employee
```

Update Employee Component

Path - `src/app/update-employee/update-employee.component.ts`

`UpdateEmployeeComponent` is used to update an existing employee. In this `UpdateEmployeeComponent`, we first get the employee object using REST API and populate in HTML form via data binding. Users can edit the employee form data and submit the form.

Let's add the following code to `UpdateEmployeeComponent` -

```
import { Component, OnInit } from '@angular/core';
import { Employee } from '../employee';
import { ActivatedRoute, Router } from '@angular/router';
import { EmployeeService } from '../employee.service';

@Component({
  selector: 'app-update-employee',
  templateUrl: './update-employee.component.html',
  styleUrls: ['./update-employee.component.css']
})
export class UpdateEmployeeComponent implements OnInit {

  id: number;
  employee: Employee;

  constructor(private route: ActivatedRoute,private router: Router,
    private employeeService: EmployeeService) { }
```

Angular + Spring Boot + MySQL CURD Application

```
  ngOnInit() {
    this.employee = new Employee();

    this.id = this.route.snapshot.params['id'];

    this.employeeService.getEmployee(this.id)
      .subscribe(data => {
        console.log(data)
        this.employee = data;
      }, error => console.log(error));
  }

  updateEmployee() {
    this.employeeService.updateEmployee(this.id, this.employee)
      .subscribe(data => console.log(data), error => console.log(error));
    this.employee = new Employee();
    this.gotoList();
  }

  onSubmit() {
    this.updateEmployee();
  }

  gotoList() {
    this.router.navigate(['/employees']);
  }
}
```

Update Employee Template

Path - src/app/update-employee/update-employee.component.html The update-employee.component.html shows the update employee HTML form. Add the following code to this file -

```
<h3>Update Employee</h3>
<div [hidden]="submitted" style="width: 400px;">
  <form (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="name">First Name</label>
      <input type="text" class="form-control" id="firstName" required [(ngModel)]="employee.firstName" name="firstName">
    </div>

    <div class="form-group">
      <label for="name">Last Name</label>
      <input type="text" class="form-control" id="lastName" required [(ngModel)]="employee.lastName" name="lastName">
    </div>

    <div class="form-group">
      <label for="name">First Name</label>
      <input type="text" class="form-control" id="emailId" required [(ngModel)]="employee.emailId" name="emailId">
    </div>

    <button type="submit" class="btn btn-success">Submit</button>
  </form>
</div>
```

5. Create View Employee Details Template and Component

Here we create view employee details functionality. Let's create an HTML template and component of **Employee** details functionality.

Angular + Spring Boot + MySQL CURD Application

Employee Details Component

Path - src/app/employee-details/employee-details.component.ts

The EmployeeDetailsComponent component used to display a particular employee detail. Add the following code to it -

```
import { Employee } from '../employee';
import { Component, OnInit, Input } from '@angular/core';
import { EmployeeService } from '../employee.service';
import { EmployeeListComponent } from '../employee-list/employee-list.component';
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-employee-details',
  templateUrl: './employee-details.component.html',
  styleUrls: ['./employee-details.component.css']
})
export class EmployeeDetailsComponent implements OnInit {

  id: number;
  employee: Employee;

  constructor(private route: ActivatedRoute,private router: Router,
    private employeeService: EmployeeService) { }

  ngOnInit() {
    this.employee = new Employee();

    this.id = this.route.snapshot.params['id'];

    this.employeeService.getEmployee(this.id)
      .subscribe(data => {
        console.log(data)
        this.employee = data;
      }, error => console.log(error));
  }

  list(){
    this.router.navigate(['employees']);
  }
}
```

Employee Details Component Template

Path - src/app/employee-details/employee-details.component.html

The `employee-details.component.html` displays a particular employee detail. Add the following code to it -

```
<h2>Employee Details</h2>

<hr/>
<div *ngIf="employee">
  <div>
    <label><b>First Name: </b></label> {{employee.firstName}}
  </div>
  <div>
    <label><b>Last Name: </b></label> {{employee.lastName}}
  </div>
  <div>
    <label><b>Email Id: </b></label> {{employee.emailId}}
  </div>
</div>

<br>
<br>
```

Angular + Spring Boot + MySQL CURD Application

```
<button (click)="list()" class="btn btn-primary">Back to Employee List</button><br>
```

6. Employee Service

Path - src/app/employee.service.ts

The **EmployeeService** will be used to get the data from the backend by calling spring boot APIs. Update the `employee.service.ts` file inside `src/app` directory with the following code to it -

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class EmployeeService {

  private baseUrl = 'http://localhost:8080/springboot-crud-rest/api/v1/employees';

  constructor(private http: HttpClient) { }

  getEmployee(id: number): Observable<any> {
    return this.http.get(`${this.baseUrl}/${id}`);
  }

  createEmployee(employee: Object): Observable<Object> {
    return this.http.post(`${this.baseUrl}`, employee);
  }

  updateEmployee(id: number, value: any): Observable<Object> {
    return this.http.put(`${this.baseUrl}/${id}`, value);
  }

  deleteEmployee(id: number): Observable<any> {
    return this.http.delete(`${this.baseUrl}/${id}`, { responseType: 'text' });
  }

  getEmployeesList(): Observable<any> {
    return this.http.get(`${this.baseUrl}`);
  }
}
```

This completed the development of CRUD operations using Angular 9.

Table of contents

- npm package.json - Configure Dependencies
- App Routing Module
- App Component
- App Component Template
- App Module
- Main Index Html File
- Main (Bootstrap) File
- Polyfills
- TypeScript tsconfig.json

npm package.json - Configure Dependencies

Path: /package.json

The `package.json` file contains project configuration information including package dependencies which get installed when you run *`npm install`*.

Note that angular version `~9.0.0-rc.7` in the dependencies section in the below file.

```
{
  "name": "angular9-springboot-client",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~9.0.0-rc.7",
    "@angular/common": "~9.0.0-rc.7",
    "@angular/compiler": "~9.0.0-rc.7",
    "@angular/core": "~9.0.0-rc.7",
    "@angular/forms": "~9.0.0-rc.7",
    "@angular/platform-browser": "~9.0.0-rc.7",
    "@angular/platform-browser-dynamic": "~9.0.0-rc.7",
    "@angular/router": "~9.0.0-rc.7",
    "bootstrap": "^4.4.1",
    "jquery": "^3.4.1",
    "rxjs": "~6.5.3",
    "tslib": "^1.10.0",
    "zone.js": "~0.10.2"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.900.0-rc.7",
    "@angular/cli": "~9.0.0-rc.7",
    "@angular/compiler-cli": "~9.0.0-rc.7",
    "@angular/language-service": "~9.0.0-rc.7",
    "@types/node": "^12.11.1",
    "@types/jasmine": "~3.5.0",
    "@types/jasminewd2": "~2.0.3",
    "codelyzer": "^5.1.2",
    "jasmine-core": "~3.5.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~4.3.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage-istanbul-reporter": "~2.1.0",
    "karma-jasmine": "~2.0.1",
    "karma-jasmine-html-reporter": "^1.4.2",
    "protractor": "~5.4.2",
    "ts-node": "~8.3.0",
    "tslint": "~5.18.0",
    "typescript": "~3.6.4"
  }
}
```

App Routing Module

Path: /src/app/app.routing.module.ts

Angular + Spring Boot + MySQL CURD Application

Routing for the Angular app is configured as an array of **Routes**, each component is mapped to a path so the Angular Router knows which component to display based on the URL in the browser address bar.

```
import { EmployeeDetailsComponent } from './employee-details/employee-details.component';
import { CreateEmployeeComponent } from './create-employee/create-employee.component';
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { UpdateEmployeeComponent } from './update-employee/update-employee.component';

const routes: Routes = [
  { path: '', redirectTo: 'employee', pathMatch: 'full' },
  { path: 'employees', component: EmployeeListComponent },
  { path: 'add', component: CreateEmployeeComponent },
  { path: 'update/:id', component: UpdateEmployeeComponent },
  { path: 'details/:id', component: EmployeeDetailsComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

App Component

Path: /src/app/app.component.ts

The app component is the root component of the application, it defines the root tag of the app as with the selector property of the **@Component** decorator.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Angular 9 + Spring Boot 2 CRUD Tutorial';
}
```

App Component Template

Path: /src/app/app.component.html

Defines the HTML template associated with the root AppComponent.

```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
  <!-- Links -->
  <ul class="navbar-nav">
    <li class="nav-item">
        <a routerLink="employees" class="nav-link" routerLinkActive="active">Employee List</a>
    </li>
    <li class="nav-item">
        <a routerLink="add" class="nav-link" routerLinkActive="active">Add Employee</a>
    </li>
  </ul>
</nav>
<div class="container">
  <br>
  <h2 style="text-align: center;">{{title}}</h2>
  <hr>
  <div class="card">
```

Angular + Spring Boot + MySQL CURD Application

```
    <div class="card-body">
  <router-outlet></router-outlet>
    </div>
  </div>
</div>

<footer class="footer">
  <div class="container">
      <span>All Rights Reserved 2019 @JavaGuides</span>
  </div>
</footer>
```

App Module

Path: /src/app/app.module.ts

Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { CreateEmployeeComponent } from './create-employee/create-employee.component';
import { EmployeeDetailsComponent } from './employee-details/employee-details.component';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { HttpClientModule } from '@angular/common/http';
import { UpdateEmployeeComponent } from './update-employee/update-employee.component';
@NgModule({
  declarations: [
    AppComponent,
    CreateEmployeeComponent,
    EmployeeDetailsComponent,
    EmployeeListComponent,
    UpdateEmployeeComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Main Index Html File

Path: /src/index.html

The main `index.html` file is the initial page loaded by the browser that kicks everything off. Webpack bundles all of the javascript files together and injects them into the body of the index.html page so the scripts get loaded and executed by the browser.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Angular9SpringbootClient</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

```
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Main (Bootstrap) File

Path: /src/main.ts

The main file is the entry point used by angular to launch and bootstrap the application.

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

Polyfills

Path: /src/polyfills.ts

Some features used by Angular 8 are not yet supported natively by all major browsers, polyfills are used to add support for features where necessary so your Angular 8 application works across all major browsers.

```
import 'core-js/features/reflect';
import 'zone.js/dist/zone';
```

TypeScript tsconfig.json

Path: /tsconfig.json

The tsconfig.json file configures how the TypeScript compiler will convert TypeScript into JavaScript that is understood by the browser. More information is available on the TypeScript docs.

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "module": "esnext",
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
```

```
      "es2018",
      "dom"
    ]
  }
}
```

This completes the Angular 9 web application configuration and app-related components.

Running Angular 9 Client Application

Let's run the above developed Angular App with a command:

```
ng serve
```

By default, the Angular app runs on **4200** port but you can change default port with the following command:

```
ng serve --port 4201
```

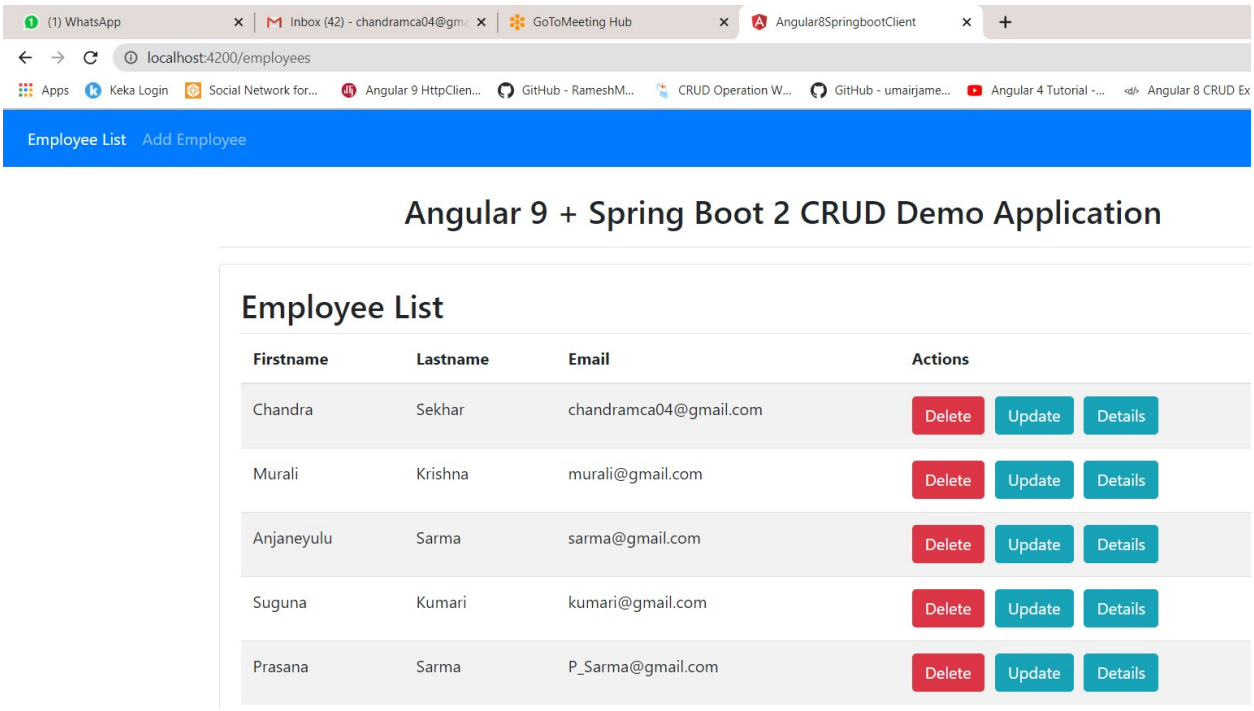
Demo

Hit http://localhost:4200 link in a browser will host this Angular 9 CRUD app.

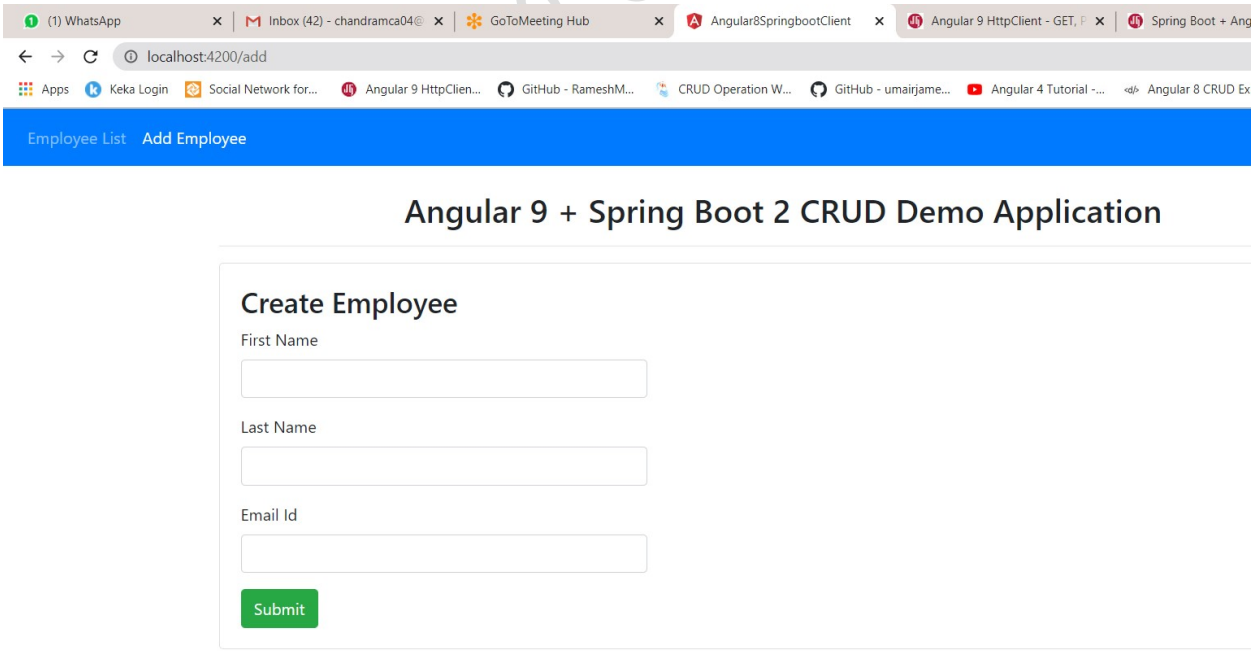
Below are the screenshots shows UI of our **Employee Management System** App:

Angular + Spring Boot + MySQL CURD Application

Employee List Page

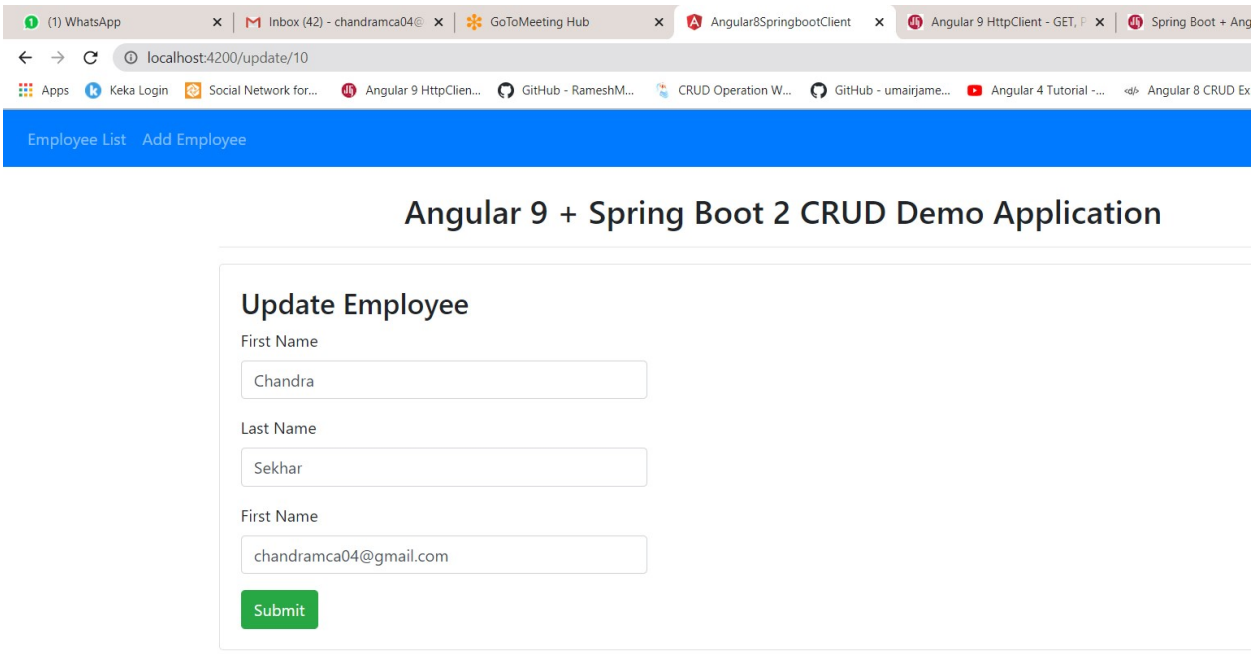


Add Employee Page

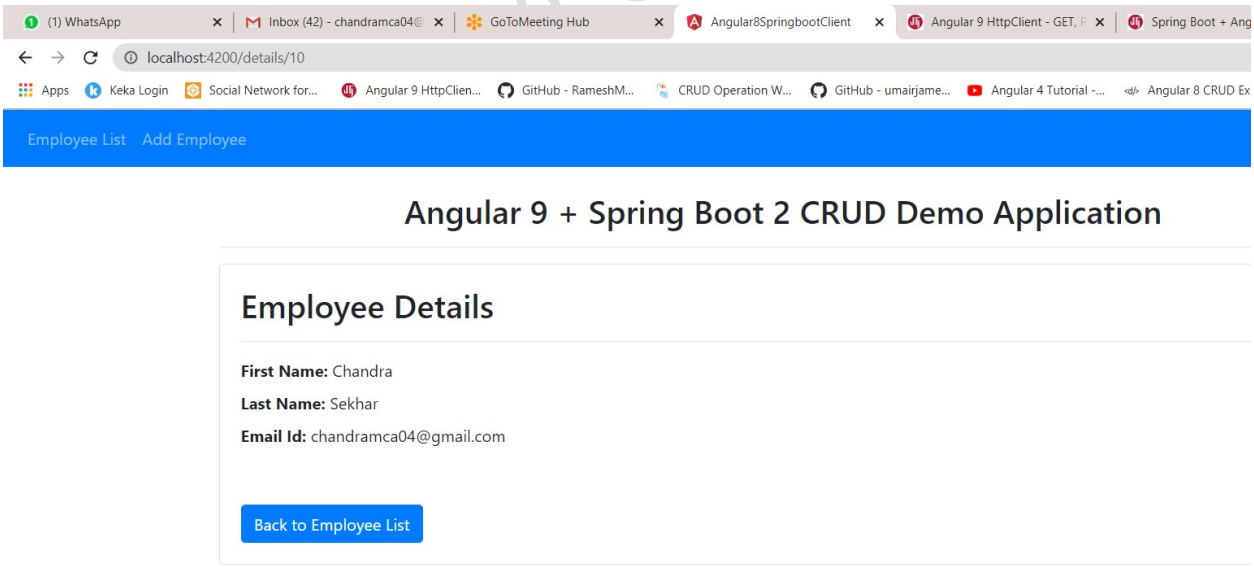


Angular + Spring Boot + MySQL CURD Application

Update Employee Page

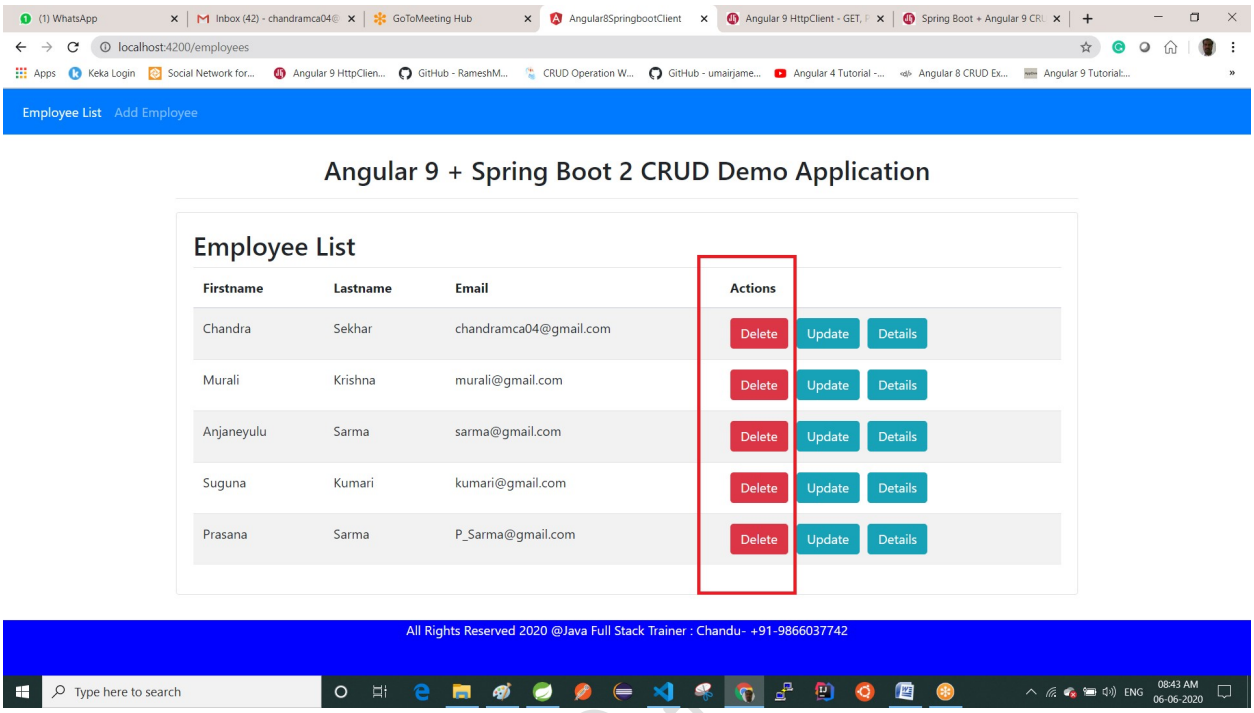


View Employee Details Page

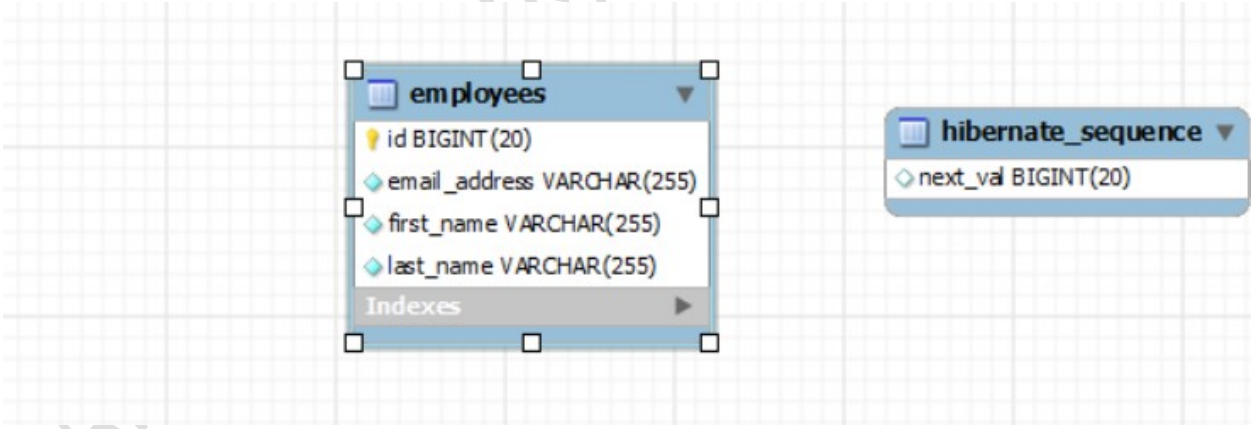


Angular + Spring Boot + MySQL CURD Application

Delete Employee



But not least Database Table also required this actually first you need to create




```
CREATE DATABASE  IF NOT EXISTS `users_database` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `users_database`;

-- MySQL dump 10.13  Distrib 5.1.40, for Win32 (ia32)

--
```

Angular + Spring Boot + MySQL CURD Application

```
-- Host: localhost    Database: users_database

-- ------------------------------------------------------

-- Server version           5.5.25


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--

-- Table structure for table `employees`

--


DROP TABLE IF EXISTS `employees`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `employees` (

  `id` bigint(20) NOT NULL,

  `email_address` varchar(255) NOT NULL,
```

Angular + Spring Boot + MySQL CURD Application

```
 `first_name` varchar(255) NOT NULL,
 `last_name` varchar(255) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `employees`
--

LOCK TABLES `employees` WRITE;
/*!40000 ALTER TABLE `employees` DISABLE KEYS */;
INSERT INTO `employees` VALUES
(10,'chandramca04@gmail.com','Chandra','Sekhar'),(22,'murali@gmail.com','Murali','Krishna'),(26,'sarma@gmail.com','Anjaneyulu','Sarma'),(27,'kumari@gmail.com','Suguna','Kumari'),(28,'P_Sarma@gmail.com','Prasana','Sarma');
/*!40000 ALTER TABLE `employees` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Angular + Spring Boot + MySQL CURD Application

```
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;


CREATE DATABASE  IF NOT EXISTS `users_database` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `users_database`;

-- MySQL dump 10.13  Distrib 5.1.40, for Win32 (ia32)

--

-- Host: localhost    Database: users_database

-- ------------------------------------------------------

-- Server version           5.5.25


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--

-- Table structure for table `hibernate_sequence`

--
```

Angular + Spring Boot + MySQL CURD Application

```
DROP TABLE IF EXISTS `hibernate_sequence`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `hibernate_sequence` (

  `next_val` bigint(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;



--

-- Dumping data for table `hibernate_sequence`

--



LOCK TABLES `hibernate_sequence` WRITE;

/*!40000 ALTER TABLE `hibernate_sequence` DISABLE KEYS */;

INSERT INTO `hibernate_sequence` VALUES (29);

/*!40000 ALTER TABLE `hibernate_sequence` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;



/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Angular + Spring Boot + MySQL CURD Application

```
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```