Search Tutorials

Custom Search

Close X

This site uses coc

es. Learn more

×

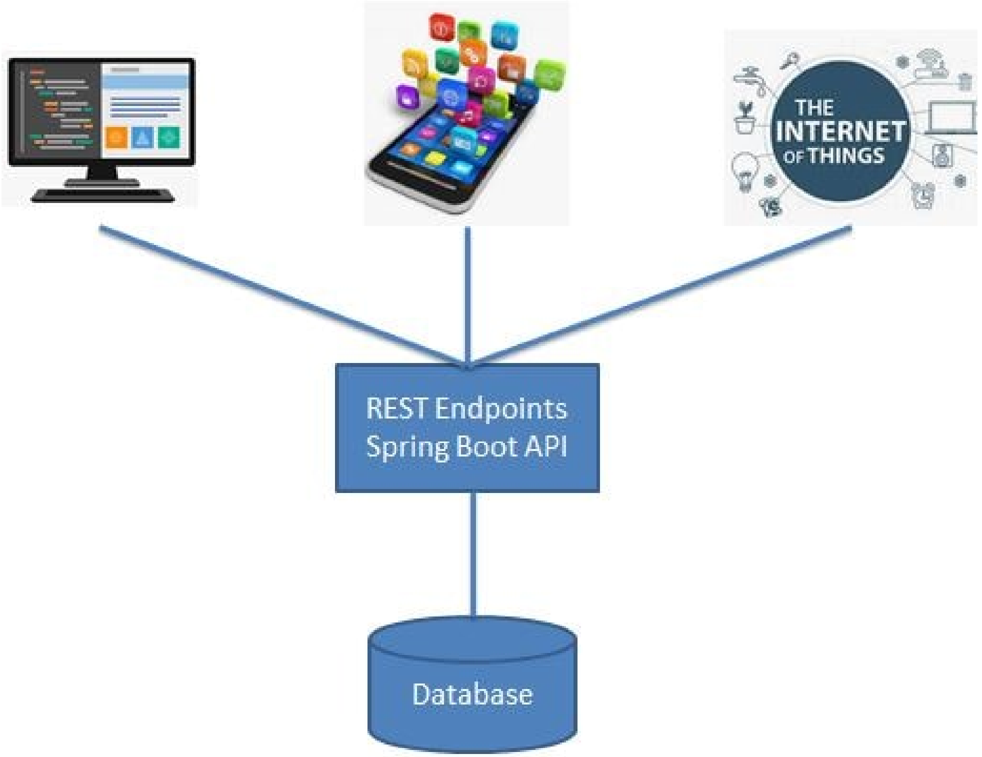
Angular 7 + Spring Boot Application Hello World Example

VOS PRODUITS COQIVOIRE LIVRÉS CHEZ VOUS
COMMANDEZ AU 77 24 90 95 OU DANS LES EFC
*Disponibilité de la ligne

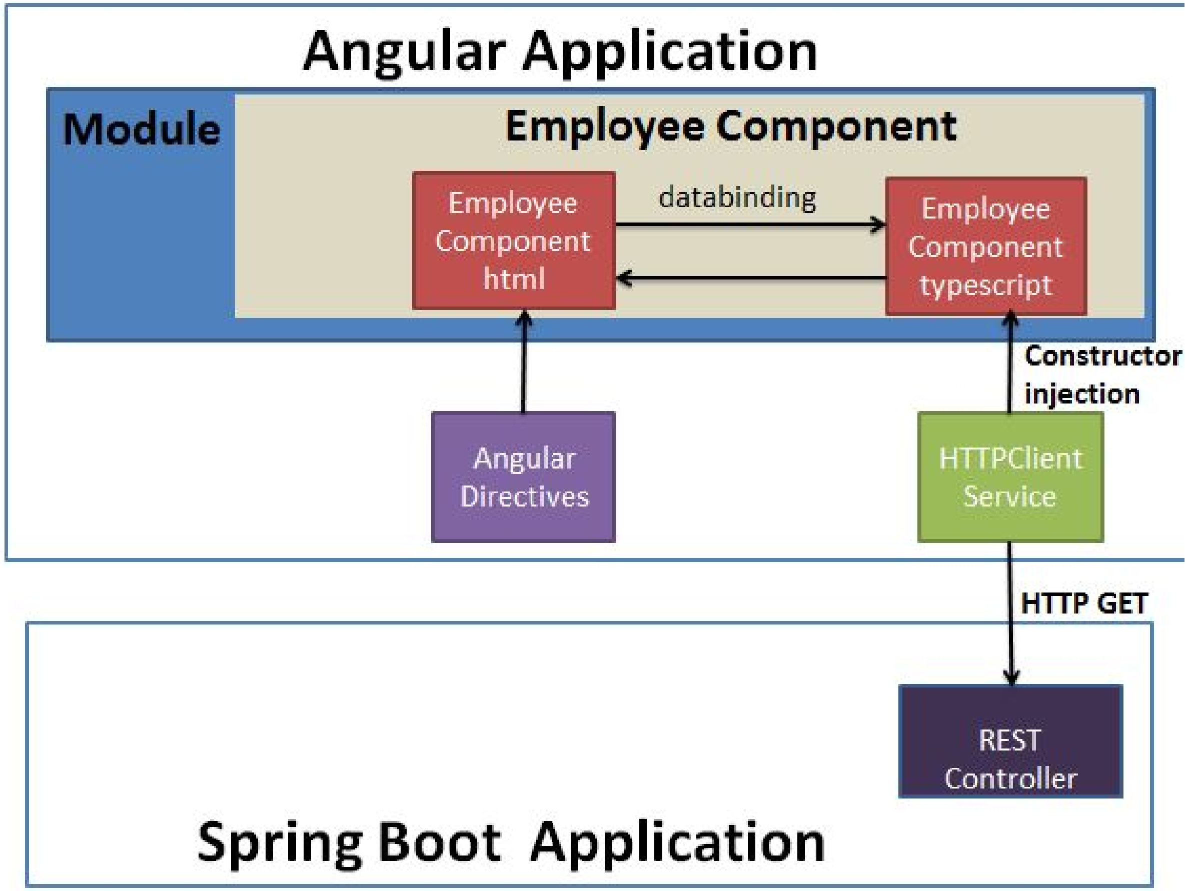
In this tutorial we be creating a full stack application where we expose endpoint using Spring Boot and consume this endpoint using Angular 7 application and display the data. In the next tutorial we will be further enhancing this application and performing CRUD operations. (/spring/ang7-crud)

Previously we have seen what is PCF and how to deploy application to PCF. (/pcf). I have deployed this application we are developing to PCF. **What is a full stack application?**

In full stack application we expose the back end point to get the data. This data can then be used by any application or device as per the need. In future even if another front end device is to be used, there will not be much change and the new device will need to consume these end points.



The project architecture we will be developing is as follows-



Angular 7+ Spring Boot - Table of Contents

Video

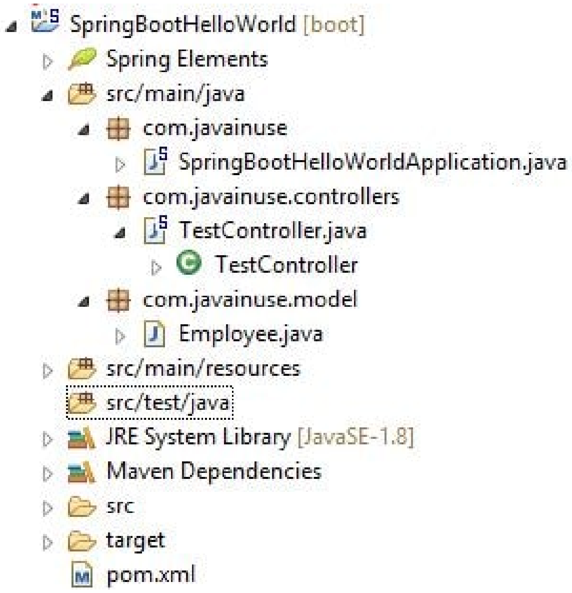
This tutorial is explained in the below Youtube Video.

Angular 7 + Spring Boot Hello World Application Example

Close X

- # Spring Boot Application

We will be creating a simple Spring Boot Application to expose a REST end point to return a list of employees. In a previous tutorial we had seen how to fetch the data using Spring Boot JDBC. (/spring/bootjdbc)
However for this tutorial we will be mocking the list of employees to be returned.
Maven Project will be as follows-



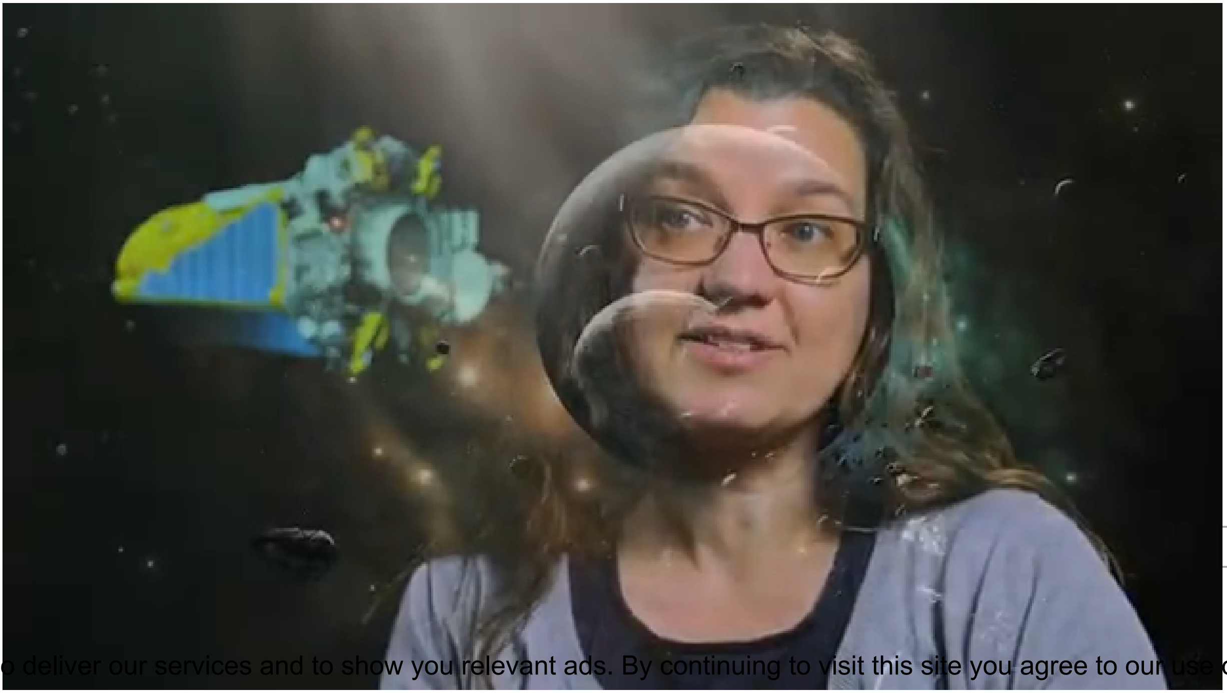
```
SpringBootHelloWorld [boot]
  Spring Elements
  src/main/java
    com.javainuse
      SpringBootHelloWorldApplication.java
    com.javainuse.controllers
      TestController.java
        TestController
    com.javainuse.model
      Employee.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
```

The maven will be as follows -

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 ht
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.javainuse</groupId>
        <artifactId>SpringBootHelloWorld</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>SpringBootHelloWorld</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifact
                <version>2.1.1.RELEASE</version>
                <relativePath /> <!-- lookup parent from reposit
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.bui
                <project.reporting.outputEncoding>UTF-8</project
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</group
                        <artifactId>spring-boot-starter-web</art
                </dependency>
```

Close X

```
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boc
                                <artifactId>spring-boot-maven-pl
                        </plugin>
                </plugins>
        </build>


</project>
```



Create the SpringBootHelloWorldApplication.java as below-

```
package com.javainuse;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBootHelloWorldApplication {

        public static void main(String[] args) {
                SpringApplication.run(SpringBootHelloWorldApplicatio
        }
}
```

Create the Employee model class as follows-

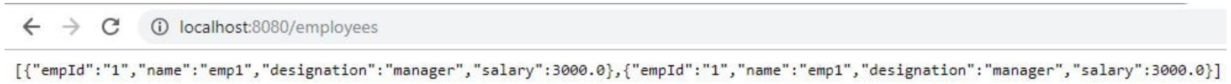
```
package com.javainuse.model;

public class Employee {
        private String empId;
        private String name;
        private String designation;
        private double salary;

        public Employee() {
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getDesignation() {
                return designation;
        }

        public void setDesignation(String designation) {
                this.designation = designation;
        }

        public double getSalary() {
                return salary;
        }
        public void setSalary(double salary) {
                this.salary = salary;
        }

        public String getEmpId() {
                return empId;
        }

        public void setEmpId(String empId) {
                this.empId = empId;
        }

        @Override
        public int hashCode() {
                final int prime = 31;
                int result = 1;
                result = prime * result + ((designation == null) ? 0
                result = prime * result + ((empId == null) ? 0 : emp
                result = prime * result + ((name == null) ? 0 : name
                long temp;
                temp = Double.doubleToLongBits(salary);
                result = prime * result + (int) (temp ^ (temp >>> 32
                return result;
        }

        @Override
        public boolean equals(Object obj) {
                if (this == obj)
                        return true;
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                Employee other = (Employee) obj;
                if (designation == null) {
                        if (other.designation != null)
                                return false;
                } else if (!designation.equals(other.designation))
                        return false;
                if (empId == null) {
                        if (other.empId != null)
                                return false;
                } else if (!empId.equals(other.empId))
                        return false;
                if (name == null) {
                        if (other.name != null)
                                return false;
                } else if (!name.equals(other.name))
                        return false;
                if (Double.doubleToLongBits(salary) != Double.double
                        return false;
                return true;
        }

}
```

@RequestMapping maps /employee request to return a list of employees. Also here we are using **CrossOrigin** annotation to specify that calls will be made to this controller from different domains. In our case we have specified that a call can be made form localhost:4200.

```
package com.javainuse.controllers;

import java.util.ArrayList;
import java.util.List;

import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.javainuse.model.Employee;

@CrossOrigin(origins = "http://localhost:4200")
@RestController
public class TestController {

        private List<Employee> employees = createList();

        @RequestMapping(value = "/employees", method = RequestMethod
        public List<Employee> firstPage() {
                return employees;
        }

        private static List<Employee> createList() {
                List<Employee> tempEmployees = new ArrayList<>();
                Employee emp1 = new Employee();
                emp1.setName("emp1");
                emp1.setDesignation("manager");
                emp1.setEmpId("1");
                emp1.setSalary(3000);
                Employee emp2 = new Employee();
                emp2.setName("emp2");
                emp2.setDesignation("developer");
                emp2.setEmpId("2");
                emp2.setSalary(3000);
                tempEmployees.add(emp1);
                tempEmployees.add(emp2);
                return tempEmployees;
        }

}
```

Compile and the run the SpringBootHelloWorldApplication.java as a Java application.

Go to **localhost:8080/employees**



```
[{"empId":"1","name":"emp1","designation":"manager","salary":3000.0},{"empId":"1","name":"emp1","designation":"manager","salary":3000.0}]
```

-

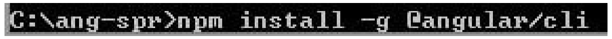
Angular 7 development

Installing Angular CLI

- Install node js by downloading the installable from Install NodeJS (https://nodejs.org/en/)

- install angular cli using the following command.It wll get us the latest version of angular cli.

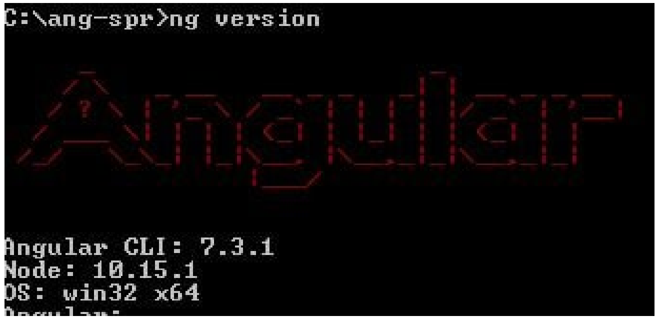
```
        npm install -g @angular/cli
```



```
C:\ang-spr>npm install -g @angular/cli
```

- We can check the angular cli version -

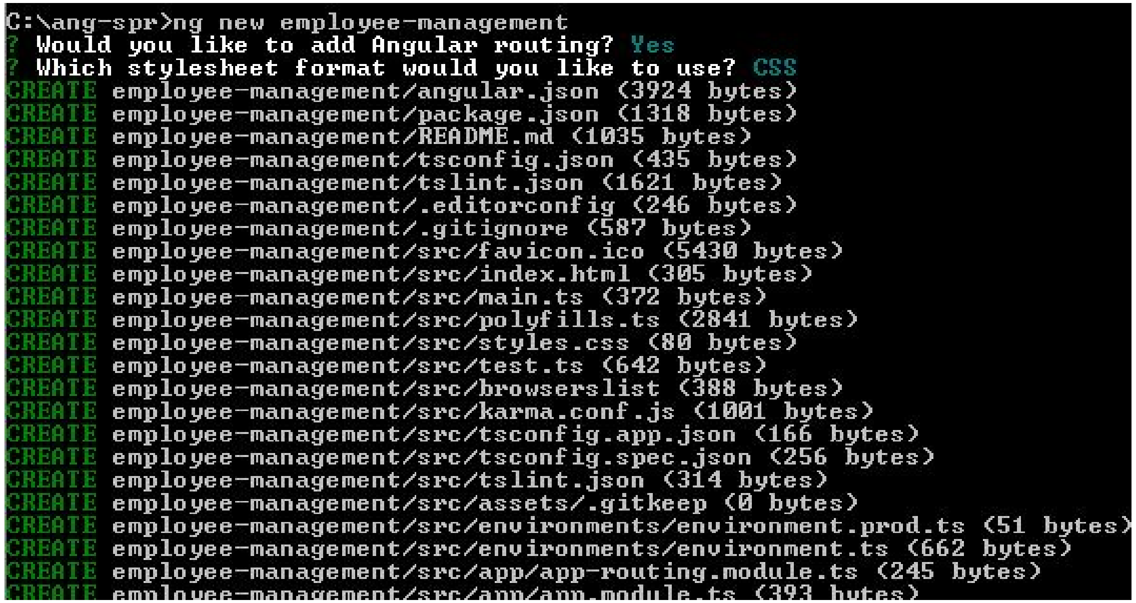
```
        ng version
```



```
C:\ang-spr>ng version

Angular CLI: 7.3.1
Node: 10.15.1
OS: win32 x64
Angular:
```

- Next we will create a new angular project using the angular cli as follows-

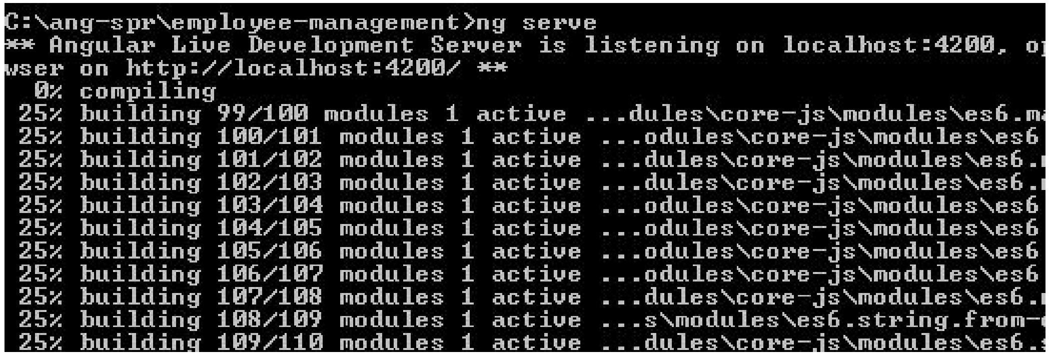
```
        ng new employee-management
```



```
C:\ang-spr>ng new employee-management
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE employee-management/angular.json (3924 bytes)
CREATE employee-management/package.json (1318 bytes)
CREATE employee-management/README.md (1035 bytes)
CREATE employee-management/tsconfig.json (435 bytes)
CREATE employee-management/tslint.json (1621 bytes)
CREATE employee-management/.editorconfig (246 bytes)
CREATE employee-management/.gitignore (587 bytes)
CREATE employee-management/src/favicon.ico (5430 bytes)
CREATE employee-management/src/index.html (305 bytes)
CREATE employee-management/src/main.ts (372 bytes)
CREATE employee-management/src/polyfills.ts (2841 bytes)
CREATE employee-management/src/styles.css (80 bytes)
CREATE employee-management/src/test.ts (642 bytes)
CREATE employee-management/src/browserslist (388 bytes)
CREATE employee-management/src/karma.conf.js (1001 bytes)
CREATE employee-management/src/tsconfig.app.json (166 bytes)
CREATE employee-management/src/tsconfig.spec.json (256 bytes)
CREATE employee-management/src/tslint.json (314 bytes)
CREATE employee-management/src/assets/.gitkeep (0 bytes)
CREATE employee-management/src/environments/environment.prod.ts (51 bytes)
CREATE employee-management/src/environments/environment.ts (662 bytes)
CREATE employee-management/src/app/app-routing.module.ts (245 bytes)
CREATE employee-management/src/app/app.module.ts (393 bytes)
```

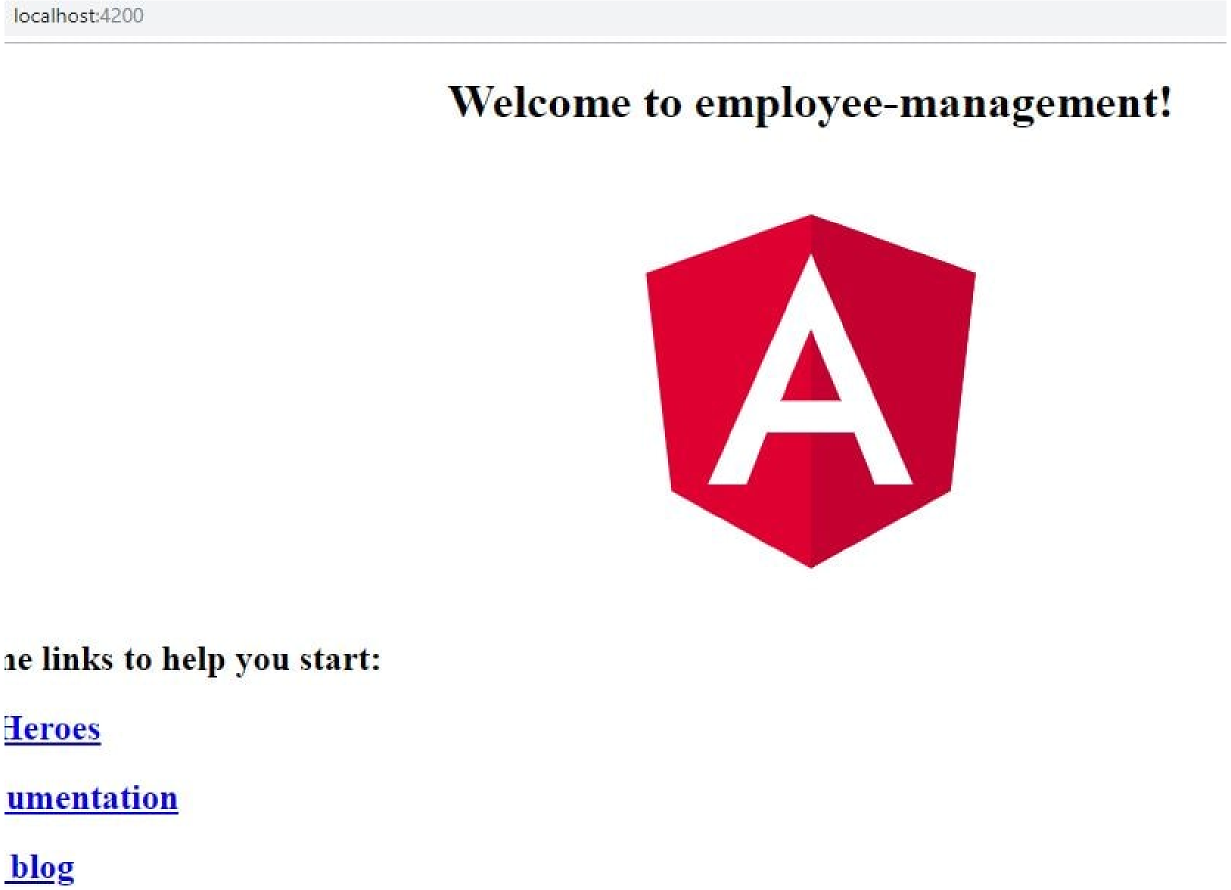
- To get the angular cli project started use the following command. We must go inside the employee-management folder and then use it.

```
    ng serve
```



```
C:\ang-spr\employee-management>ng serve
** Angular Live Development Server is listening on localhost:4200, o
wser on http://localhost:4200/ **
  0% compiling
 25% building 99/100 modules 1 active ...dules\core-js\modules\es6.m
 25% building 100/101 modules 1 active ...odules\core-js\modules\es6
 25% building 101/102 modules 1 active ...dules\core-js\modules\es6.
 25% building 102/103 modules 1 active ...dules\core-js\modules\es6.
 25% building 103/104 modules 1 active ...odules\core-js\modules\es6
 25% building 104/105 modules 1 active ...odules\core-js\modules\es6
 25% building 105/106 modules 1 active ...odules\core-js\modules\es6
 25% building 106/107 modules 1 active ...odules\core-js\modules\es6
 25% building 107/108 modules 1 active ...dules\core-js\modules\es6.
 25% building 108/109 modules 1 active ...s\modules\es6.string.from-
 25% building 109/110 modules 1 active ...dules\core-js\modules\es6.
```

Go to **localhost:4200**



localhost:4200

Welcome to employee-management!

ıe links to help you start:

Ieroes

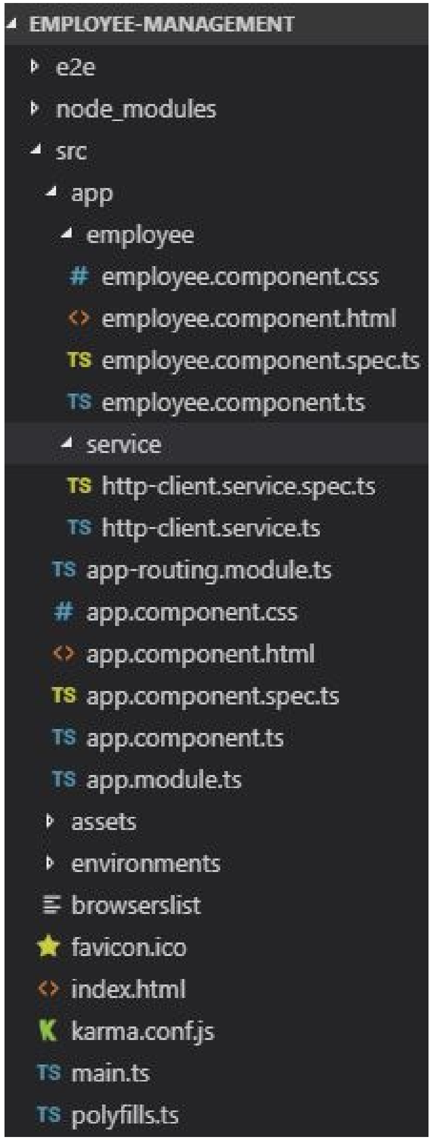
umentation

blog

I will be using the Miscrosoft Visual Studio Code IDE (https://code.visualstudio.com/download) for angular. So import the project we developed earlier in Miscrosoft Visual Studio Code IDE.

Close X

Our final angular project will be as follows-

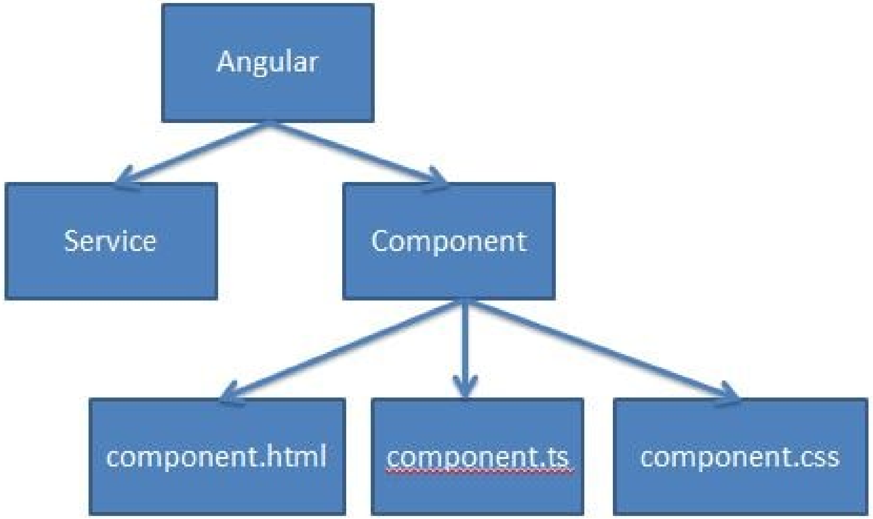


```
EMPLOYEE-MANAGEMENT
  e2e
  node_modules
  src
    app
      employee
        employee.component.css
        employee.component.html
        employee.component.spec.ts
        employee.component.ts
      service
        http-client.service.spec.ts
        http-client.service.ts
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
    environments
    browserslist
    favicon.ico
    index.html
    karma.conf.js
    main.ts
    polyfills.ts
```

- TypeScript

TypeScript is a superset of JavaScript. It is a strongly typed language. So unlike JavaScript we know if some syntax is wrong while typing itself and not at runtime. In Angular it is compiled to JavaScript while rendering application in browser.

- ## Component



In angular we break complex code into reusable parts called components. Major part of the development with Angular 7 is done in the components. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser.

Close X

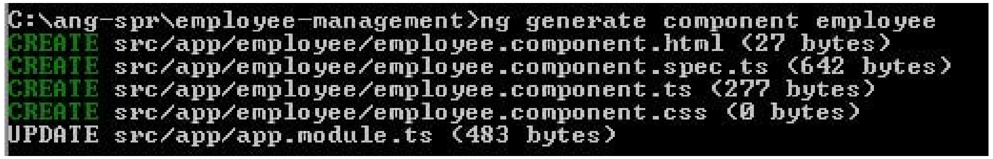
- ## Service

In angular we might have scenarios where some code needs to be reused in multiple components. For example a data connection that fetches data from database might be needed in multiple components. This is achieved using services.

Create employee component

We will be creating Employee Component which will fetch data from spring boot and display it. Lets begin with the employee component Open a command prompt and use the following command-

```
        ng generate component employee
```



```
C:\ang-spr\employee-management>ng generate component employee
CREATE src/app/employee/employee.component.html (27 bytes)
CREATE src/app/employee/employee.component.spec.ts (642 bytes)
CREATE src/app/employee/employee.component.ts (277 bytes)
CREATE src/app/employee/employee.component.css (0 bytes)
UPDATE src/app/app.module.ts (483 bytes)
```

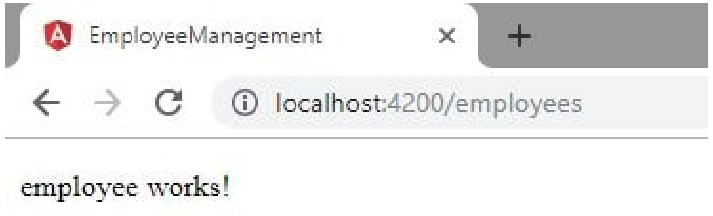
For this angular will have created the following 4 files-

- employee.component.ts
- employee.component.spec.ts
- employee.component.html
- employee.component.css

Next in the app-routing.module.ts we will be defining the url for accessing this component-

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { EmployeeComponent } from './employee/employee.component

const routes: Routes = [
  { path:'', component: EmployeeComponent},
];

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

If we goto localhost:4200 and we can see the following output



employee works!

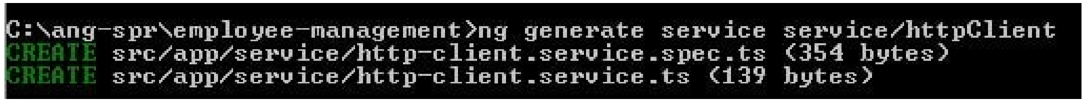
Close X

- ## Create HttpClient Service

Next we will be creating a HTTPClient Service. This service will be having the httpClient and will be responsible for calling http GET request to the backend spring boot application.

In Angular a service is written for any cross cutting concerns and may be used by more than one components

```
    ng generate service service/httpClient
```



```
C:\ang-spr\employee-management>ng generate service service/httpClient
CREATE src/app/service/http-client.service.spec.ts (354 bytes)
CREATE src/app/service/http-client.service.ts (139 bytes)
```

The following service files are created-

- http-client.service.ts
- http-client.service.spec.ts

We will be modifying the http-client.service.ts file. In the constructor define the HTTPClient instance we will be using to make a call to the Spring Boot application. Here we will be using the Angular HTTPClient for calling the Spring Boot API to fetch the employee data. Also we will creating a method which makes call to the spring boot application using the defined httpClient.

Close X

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

export class Employee{
  constructor(
    public empId:string,
    public name:string,
    public designation:string,
    public salary:string,
  ) {}
}

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

  constructor(
    private httpClient:HttpClient
  ) {
     }

     getEmployees()
  {
    console.log("test call");
    return this.httpClient.get<Employee[]>('http://localhost:808
  }
}
```

Also we need to add the HTTPClientModule to the app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { EmployeeComponent } from './employee/employee.component
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    EmployeeComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- ## Insert HttpClient Service in Employee Component

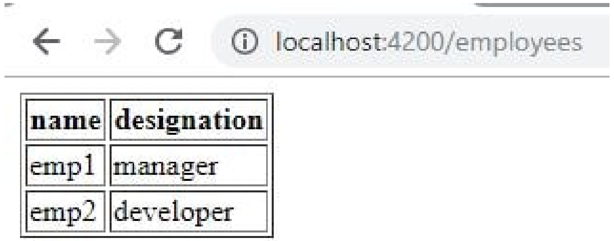
Next using constructor dependency injection we will be providing the EmployeeComponent an instance of HttpClientService. Using this service we make a call to spring boot application to get a list of employees.

```
import { Component, OnInit } from '@angular/core';
import { HttpClientService } from '../service/http-client.servic

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

  employees:string[];

  constructor(
    private httpClientService:HttpClientService
  ) { }

  ngOnInit() {
    this.httpClientService.getEmployees().subscribe(
     response =>this.handleSuccessfulResponse(response),
    );
  }

handleSuccessfulResponse(response)
{
    this.employees=response;
}

}
```

In the employee.component.html we iterate over the list of employees we got in the employee.component.ts file.

```
<table border="1">
  <thead></thead>
  <tr>
    <th>name</th>
    <th>designation</th>
    </tr>

  <tbody>
      <tr *ngFor="let employee of employees">
          <td>{{employee.name}}</td>
          <td>{{employee.designation}}</td>
          </tr>
  </tbody>
    </table>
```

Go to localhost:4200



localhost:4200/employees

name	designation
emp1	manager
emp2	developer

Close X

Download Source Code

Download it -
GITHUB- Angular 7 Hello World example code (https://github.com/JavaInUse/angular7-spring-boot)
Spring Boot Hello World example code (/zip/spring/boot/SpringBootHelloWorld-ang.rar)

Popular Posts

- E-commerce Website - Online Book Store using Angular 8 + Spring Boot (/fullstack/ecommerce)

- Spring Boot +JSON Web Token(JWT) Hello World Example (/spring/boot-jwt)
- Angular 7 + Spring Boot Application Hello World Example (/spring/ang7-hello)
- Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ (/spring/boot-websocket-chat)
- Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example (/pcf/pcf-hello)
- Deploying Spring Based WAR Application to Docker (/devOps/docker/docker-war)
- EIP patterns using Apache Camel (/camel/camel_EIP)
- Spring Cloud- Netflix Eureka + Ribbon Simple Example (/spring/spring_ribbon)
- Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example (/spring/spring_hystrix_circuitbreaker)
- Spring Boot + Swagger Example Hello World Example (/spring/boot_swagger)
- Spring Boot Batch Simple example (/spring/bootbatch)
-

- Spring Boot + Apache Kafka Example (/spring/spring-boot-apache-kafka-hello-world)
- Spring Boot Admin Simple Example (/spring/boot-admin)
- Spring Boot Security - Introduction to OAuth (/spring/spring-boot-oauth-introduction)
- Spring Boot OAuth2 Part 1 - Getting The Authorization Code (/spring/spring-boot-oauth-authorization-code)
- Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data. (/spring/spring-boot-oauth-access-token)
- JBoss Drools Hello World-Stateful Knowledge Session using KieSession (/drools_hello_kie)
- Understand Drools Stateful vs Stateless Knowledge Session (/drools_states)
- JBoss Drools- Understanding Drools Decision Table using Simple Example (/drools/drools_decision)

Close X

See Also

- Spring Boot Interview Questions (/spring/SpringBootInterviewQuestions)
- Spring Batch Interview Questions (/spring/sprbatch_interview)
- Spring AOP Interview Questions (/spring/spring-AOP-interview-quesions)
- Angular 2 Interview Questions (/angular/ang2_intvw)
- Apache Camel Interview Questions (/camel/Apache_Camel_Questions)
- JBoss Fuse Interview Questions (/camel/JBoss_Fuse_Questions)
- Drools Interview Questions (/drools/drools_intvw)
- Java 8 Interview Questions (/java/java8_intvw)
- Spring Cloud Interview Questions (/spring/spring-cloud-interview-questions)

- Microservices Interview Questions (/spring/microservices-interview-quesions)
- Java HashMap and ConcurrentHashMap Interview Questions (/java/java_map_intvw)
- Mule ESB frequently asked interview questions (/misc/muleintvw)
- Apache Kafka Interview Questions (/misc/apache-kafka-interview-questions)
- Tosca Testing Tool Interview Questions (/misc/tosca-testing-tool-interview-questions)
- Top Maven Build Tool Interview Questions (/misc/maven-interview-questions)
- Top Gradle Build Tool Interview Questions (/misc/gradle-interview-questions)
- Miscellaneous Topics (/misc)

Close X

Close X

✕