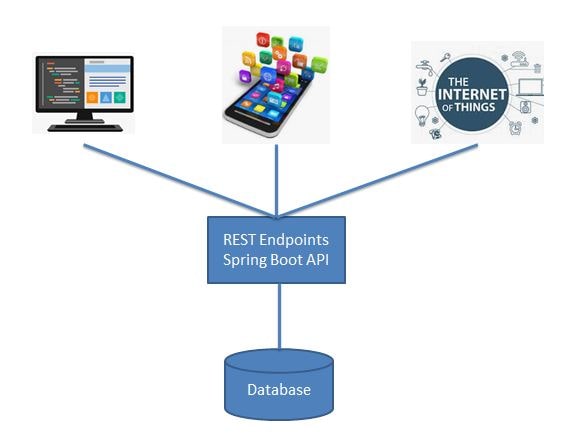
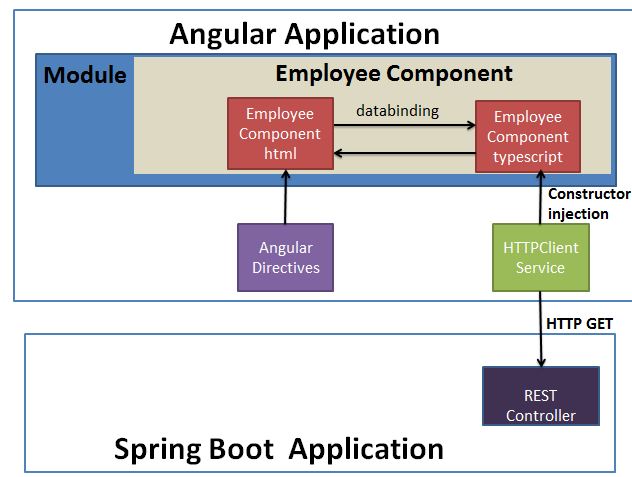
Angular 7 + Spring Boot Application Hello World Example

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.](https://www.javainuse.com/spring/ang7-crud)  
pPreviously we have seen [what is PCF and how to deploy application to PCF.](https://www.javainuse.com/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

[**Angular 7 + Spring Boot Application Hello World Example**](https://www.javainuse.com/spring/ang7-hello)

[Angular 7 + Spring Boot Application CRUD Example](https://www.javainuse.com/spring/ang7-crud)

[Angular 7 + Spring Boot Application Login Example](https://www.javainuse.com/spring/ang7-login)

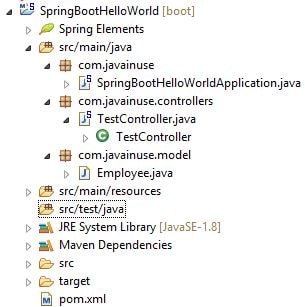
[Angular 7 + Spring Boot Application Basic Authentication Example](https://www.javainuse.com/spring/ang7-basic)

[Angular 7 + Spring Boot JWT Authentication Hello World Example](https://www.javainuse.com/spring/ang7-jwt)

Video

This tutorial is explained in the below Youtube Video.

* 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.](https://www.javainuse.com/spring/bootjdbc)  
However for this tutorial we will be mocking the list of employees to be returned. Maven Project will be as follows-  
  


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="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<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</artifactId>

<version>2.1.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</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(SpringBootHelloWorldApplication.class, args);

}

}

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 : designation.hashCode());

result = prime \* result + ((empId == null) ? 0 : empId.hashCode());

result = prime \* result + ((name == null) ? 0 : name.hashCode());

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.doubleToLongBits(other.salary))

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.GET, produces = "application/json")**

**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);

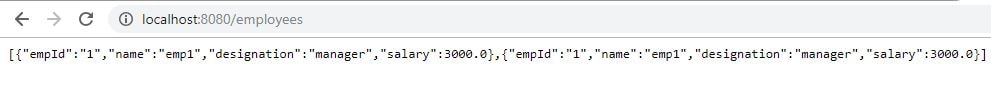
tempEmployees.add(emp1);

tempEmployees.add(emp2);

return tempEmployees;

}

}

Compile and the run the SpringBootHelloWorldApplication.java as a Java application.  
Go to **localhost:8080/employees**   


**** Angular 7 development

Installing Angular CLI

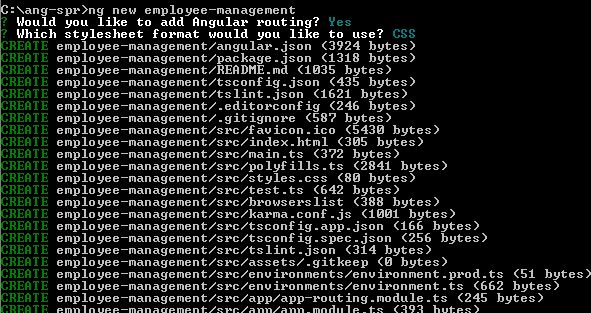
* Install node js by downloading the installable from [Install NodeJS](https://nodejs.org/en/)
* install angular cli using the following command.It wll get us the latest version of angular cli.
* npm install -g @angular/cli

angular cli 7 install

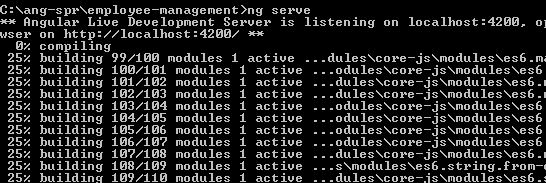
* We can check the angular cli version -
* ng version

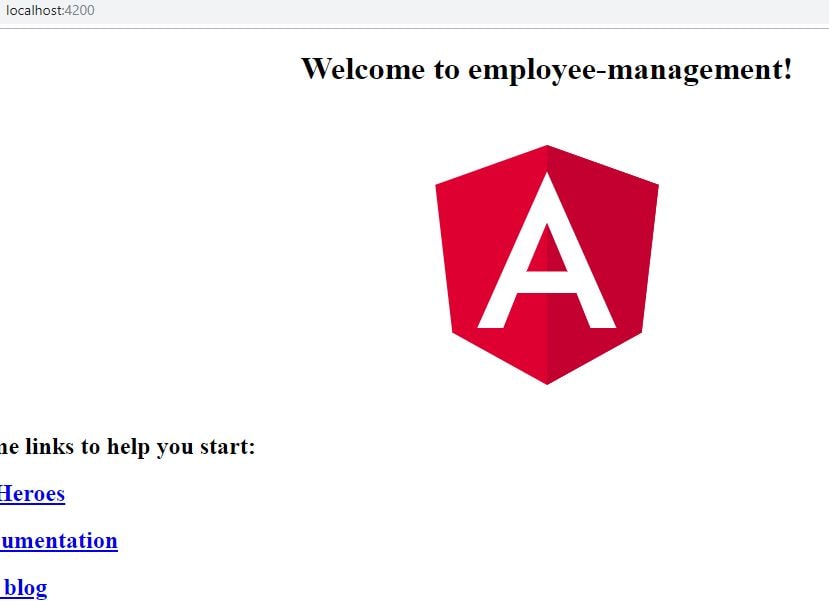


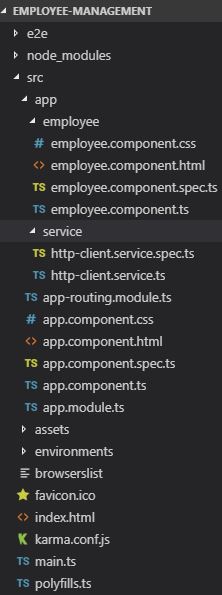
* Next we will create a new angular project using the angular cli as follows-
* ng new employee-management



* 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



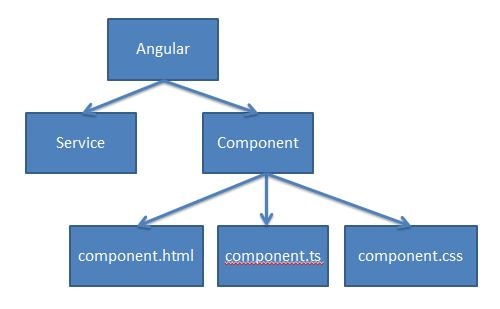
Go to **localhost:4200**   


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.  
Our final angular project will be as follows-   


* 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.

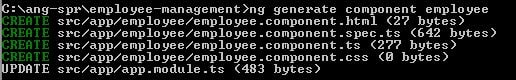
* Service

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

  
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},**

];

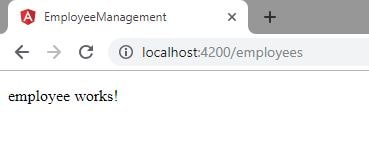
@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

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


* 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

angular generate service  
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.

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:8080/employees');**

**}**

}

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.service';

@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   


Examples:

SpringBoot project with Model,Controller and SpringBootApplication in Eclipse with Maven.

Employee.java

**package** com.cts;

**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 : designation.hashCode());

result = prime \* result + ((empId == **null**) ? 0 : empId.hashCode());

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**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.*doubleToLongBits*(other.salary))

**return** **false**;

**return** **true**;

}

}

============================================================

package com.cts;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

}

**package** com.cts;

**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;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

**public** **class** TestController {

**private** List<Employee> employees = *createList*();

@RequestMapping(value = "/employees", method = RequestMethod.***GET***, produces = "application/json")

**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);

Employee emp3 = **new** Employee();

emp3.setName("emp3");

emp3.setDesignation("Tshaped developer");

emp3.setEmpId("3");

emp3.setSalary(3000);

tempEmployees.add(emp1);

tempEmployees.add(emp2);

tempEmployees.add(emp3);

**return** tempEmployees;

}

}

Angular7application:

npm install –g @angular/cli

ng new angular7app1

cd angular7app1

* + create new employee component
  + ng g c comployee

modify the files

employee.component.ts

import { Component, OnInit } from '@angular/core';

import { HttpClientService } from '../service/http-client.service';

@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;

}

}

Employee.component.html

<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>

Create a service

ng generate service service/httpClient

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:8080/employees');**

**}**

}

In 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 { }

Add CHORS plugin in chrome browser.

Then start server

ng serve

<http://localhost:4200/>

<http://localhost:8086/employees>