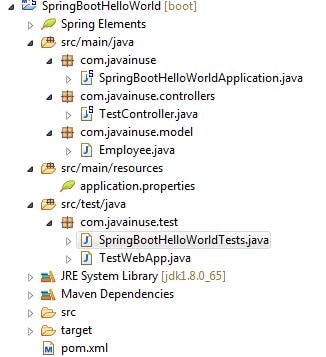
# Spring Boot Unit Test Simple Example

In this post we will write test cases for a Spring Boot Application. Spring Boot provides a number of utilities and annotations to help test a Spring Boot Application.  
Spring Boot Test is provided by two modules -

* spring-boot-test contains core items
* spring-boot-test-autoconfigure supports auto-configuration for tests

spring-boot-starter-test dependency imports both the above Spring Boot test modules as well has JUnit, AssertJ, Hamcrest and a number of other useful libraries.

Maven Project will be as follows-  
  


In the Maven we need the spring boot test dependency.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>1.4.1.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>

**<dependency>**

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

**<artifactId>spring-boot-starter-test</artifactId>**

**<scope>test</scope>**

**</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;

}

}

@RequestMapping maps /employee request to return an employee object.

package com.javainuse.controllers;

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;

@RestController

public class TestController {

@RequestMapping(value = "/employee", method = RequestMethod.GET)

public Employee firstPage() {

Employee emp = new Employee();

emp.setName("emp1");

emp.setDesignation("manager");

emp.setEmpId("1");

emp.setSalary(3000);

return emp;

}

}

Compile and the run the SpringBootHelloWorldApplication.java as a Java application.  
Go to **localhost:8080/employee**  
  
Next we write JUnit test cases for the TestController.  
Define the SpringBootHelloWorldTests.java as follows. Spring Boot provides a @SpringBootTest annotation which can be used as an alternative to the standard spring-test @ContextConfiguration annotation when you need Spring Boot features. The annotation works by creating the ApplicationContext used in your tests via SpringApplication.

package com.javainuse.test;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootHelloWorldTests {

@Test

public void contextLoads() {

}

}

Next extend the SpringBootHelloWorldTests.java and write the test case for the TestController.java

package com.javainuse.test;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import org.junit.Before;

import org.junit.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

import org.springframework.web.context.WebApplicationContext;

public class TestWebApp extends SpringBootHelloWorldTests {

@Autowired

private WebApplicationContext webApplicationContext;

private MockMvc mockMvc;

@Before

public void setup() {

mockMvc = MockMvcBuilders.webAppContextSetup(webApplicationContext).build();

}

@Test

public void testEmployee() throws Exception {

mockMvc.perform(get("/employee")).andExpect(status().isOk())

.andExpect(content().contentType("application/json;charset=UTF-8"))

.andExpect(jsonPath("$.name").value("emp1")).andExpect(jsonPath("$.designation").value("manager"))

.andExpect(jsonPath("$.empId").value("1")).andExpect(jsonPath("$.salary").value(3000));

}

}

Run this TestWebApp class as a JUnit test case.  
