Spring Boot

Rapid application Development

create stand alone spring based application

Spring Boot does not generate code and there is absolutely no requirement for XML configuration.

Embed Tomcat, Jetty or Undertow directly. You don't need to deploy WAR files

It provides opinionated 'starter' POMs to simplify your Maven configuration.

It provides production-ready features such as metrics, health checks and externalized configuration

SpringApplication

It is a class which provides the convenient way to bootstrap a spring application which can be started from main method. You can call start your application just by calling a static run() method.

@SpringBootApplication

**public** **static** **void** main(String[] args){

1. SpringApplication.run(className.**class**, args);
2. }
3. **<parent>**
4. **<groupId>**org.springframework.boot**</groupId>**
5. **<artifactId>**spring-boot-starter-parent**</artifactId>**
6. **<version>**1.4.2.RELEASE**</version>**
7. **</parent>**
8. **<dependencies>**
9. **<dependency>**
10. **<groupId>**org.springframework.boot**</groupId>**
11. **<artifactId>**spring-boot-starter-web**</artifactId>**
12. **</dependency>**
13. **</dependencies>**

## ­--------------------------------------------------------------------------

## Simple Project 1:

## ---------------------------------------------------------------------------

## 

## 

*@RestController*

*@EnableAutoConfiguration*

**public** **class** Example {

*@RequestMapping("/")*

String home() {

**return** "Hello World!";

}

## ---------------------------------------------------------------------

## @SpringBootApplication

It is composed of:

1. @SpringBootConfiguration
2. @EnableAutoConfiguration
3. @ComponentScan(excludeFilters = @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class))

#### @SpringBootConfiguration

|  |
| --- |
| @Configuration  public @interface SpringBootConfiguration  {      //more code  } |

1. This annotation adds @Configuration annotation to class which **mark the class a source of bean definitions for the application context.**

#### @EnableAutoConfiguration

This tells spring boot to auto configure important bean definitions based on added dependencies in pom.xml by start adding beans based on classpath settings, other beans, and various property settings.

#### @ComponentScan

This annotation tells spring boot to scan base package, find other beans/components and configure them as well.

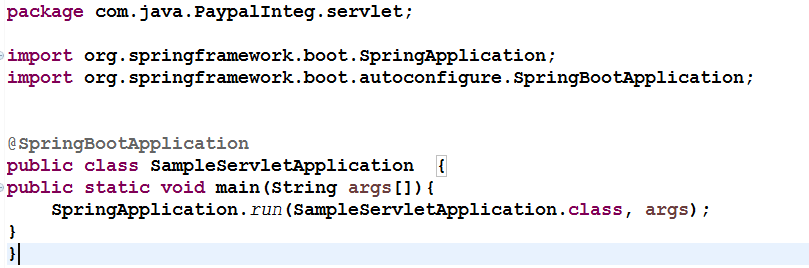
Normally you would add @EnableWebMvc for a Spring MVC app, but Spring Boot adds it automatically when it sees **spring-webmvc** on the classpath. This flags the application as a web application and activates key behaviors such as setting up a DispatcherServlet.

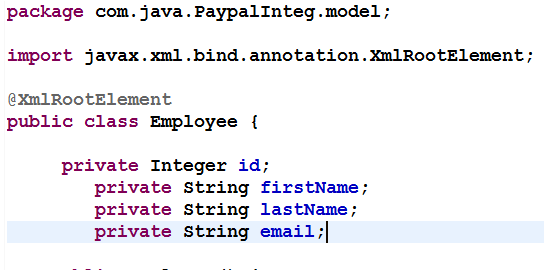
**Place @EnableAutoConfiguration in a root package so that all sub-packages and classes can be searched.**

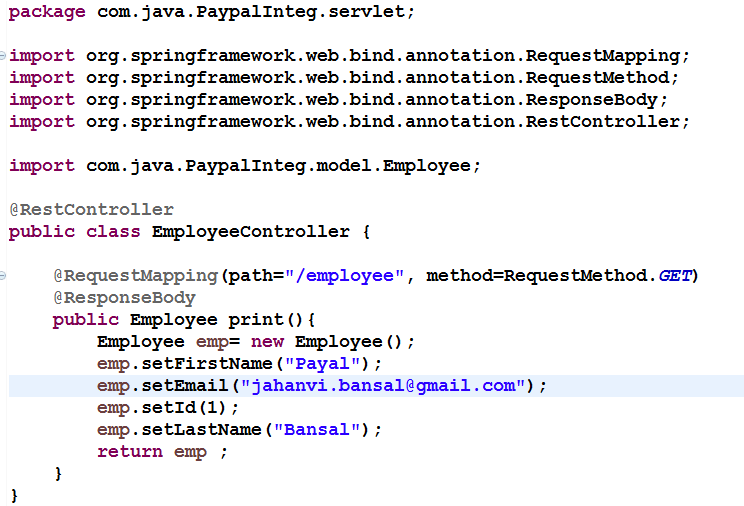
**---------------------------------------------------------------------------------------------------------**

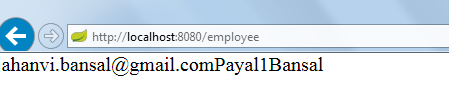
**Simple Project2:**

**---------------------------------------------------------------------------------------------------------**

****

****

****

****

**Note: Main class should be in parent/same package as Controller class**

**--------------------------------------------------------------------------------------------------------**

**Spring Boot +Hiberrnate:**

The data source properties starting with spring.datasource.\* will automatically be read by spring boot JPA. To change the hibernate properties we will use prefix spring.jpa.properties.\* with hibernate property name