What is Spring Boot?

Over the years spring has become more and more complex as new functionalities have been added. Just visit the page-<https://spring.io/projects> and we will see all the spring projects we can use in our application for different functionalities. If one has to start a new spring project we have to add build path or add maven dependencies, configure application server, add spring configuration . **So a lot of effort is required to start a new spring project as we have to currently do everything from scratch. Spring Boot is the solution to this problem**. Spring boot has been built on top of existing spring framework. Using spring boot we avoid all the boilerplate code and configurations that we had to do previously. Spring boot thus helps us use the existing Spring functionalities more robustly and with minimum efforts.  
[More details and miscellaneous examples](https://www.javainuse.com/spring/sprboot)

What are advantages of Spring Boot?

The advantages of Spring Boot are

* Reduce Developement, Testing time and efforts.
* Use of JavaConfig helps avoid usage of XML.
* Avoid lots of maven imports and the various version conflicts.
* Provide Opinionated Development approach.
* Quick start to development by providing defaults.
* No Separate Web Server Needed.Which means that you no longer have to boot up Tomcat, Glassfish, or anything else.
* Requires less configuration-Since there is no web.xml file. Simply add classes annotated with@Configuration and then you can add methods annotated with@Bean, and Spring will automagically load up the object and manage it like it always has. You can even add @Autowired to the bean method to have Spring autowire in dependencies needed for the bean.
* Environment Based Configuration-Using these properties, you can pass into the application which environment you are using with:-Dspring.profiles.active={enviornment}. Spring will then load up the subsequent application properties file at (application-{environment}.properties) after loading up the main application properties file.

Which build tool have you used to develop Spring Boot Application?

Spring Boot application can be developed using Maven as well as Gradle.  
[Spring Boot application using Maven](https://www.javainuse.com/spring/SpringBoot_HelloWorld)  
[Spring Boot application using Gradle](https://www.javainuse.com/spring/SpringBoot_HelloWorld_gradle)

**<dependencies>**

**<dependency>**

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

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

**</dependency>**

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

What is JavaConfig?

Spring JavaConfig is a product of the Spring community that provides a pure-Java approach to configuring the Spring IoC Container. It thus helps avoid using XML configurations. The advantages of using JavaConfig are  
The advantages of JavaConfig are

* Object-oriented configuration. Because configurations are defined as classes in JavaConfig, users can take full advantage of object-oriented features in Java. One configuration class may subclass another, overriding its @Bean methods, etc.
* Reduced or eliminated XML configuration. The benefits of externalized configuration based on the principles of dependency injection have been proven. However, many developers would prefer not to switch back and forth between XML and Java. JavaConfig provides developers with a pure-Java approach to configuring the Spring container that is conceptually similar to XML configuration. It is technically possible to configure the container using only JavaConfig configuration classes, however in practice many have found it ideal to mix-and-match JavaConfig with XML.
* Type-safe and refactoring-friendly. JavaConfig provides a type-safe approach to configuring the Spring container. Thanks to Java 5.0's support for generics, it is now possible to retrieve beans by type rather than by name, free of any casting or string-based lookups.

### How to reload my changes on Spring Boot without having to restart server?

This can be achieved using DEV Tools. With this dependency any changes you save, the embedded tomcat will restart. Spring Boot has a Developer tools (DevTools) module which helps to improve the productivity of developers. One of the key challenge for the Java developers is to auto deploy the file changes to server and auto restart the server. Developers can reload changes on Spring Boot without having to restart my server. This will eliminates the need for manually deploying the changes every time. Spring Boot doesnt have this feature when it has released its first version. This was a most requested features for the developers. The module DevTools does exactly what is needed for the developers. This module will be disabled in the production environment. It also provides H2-database console for better testing the application. The following dependency is used

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

### What is Actuator in Spring Boot?

Spring boot actuator is one of the important feature in spring boot framework. Spring boot actuator helps you to access the current state of the running application in production environment. There are several metrics that has to be checked and monitored in the production environment. Even some external applications may be using those services to trigger the alert message to concerned person. Actuator module exposes set of REST endpoints that can be directly accessed as a HTTP URL to check the status.

### How to run Spring boot application to custom port?

In order to run a spring boot application on a custom port you can specify the port in application.properties.  
**server.port=8090**

What is ELK stack?How to use it with Spring Boot?

The ELK Stack consists of three open-source products - Elasticsearch, Logstash, and Kibana from Elastic.

* Elasticsearch is a NoSQL database that is based on the Lucene search engine.

 Logstash is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to various targets. It is a dynamic data collection pipeline with an extensible plugin ecosystem and strong Elasticsearch synergy

 Kibana is a visualization UI layer that works on top of Elasticsearch.These three projects are used together for log analysis in various environments. So Logstash collects and parses logs, Elastic search indexes and store this information while Kibana provides a UI layer that provide actionable insights.

### What is Swagger? Have you implemented it using Spring Boot?

Swagger is widely used for visualizing APIs, and with Swagger UI it provides online sandbox for frontend developers. For the tutorial, we will use the Springfox implementation of the Swagger 2 specification. Swagger is a tool, a specification and a complete framework implementation for producing the visual representation of RESTful Web Services. It enables documentation to be updated at the same pace as the server. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Thus Swagger removes the guesswork in calling the service.

### What is Spring Profiles? How do you implement it using Spring Boot?

Spring Profiles allows users to register beans depending on the profile(dev, test, prod etc). So when the application is running in DEVELOPMENT only certain beans can be loaded and when in PRODUCTION certain other beans can be loaded. Suppose our requirement is that the Swagger documentation be enabled only for the QA environment and disabled for all others. This can be done using Profiles. Spring Boot makes using Profiles very easy.

We will define the Spring Profile for the Swagger implementation class SwaggerConfig such that it will get loaded only when the deployment is for QA else it will be disabled. For this in the application.properties file define the active profiles as follows-

spring.profiles.active=swagger-disabled-for-qa

The SwaggerConfig class annoatate with the Profile tag as follows-

package com.javainuse.swaggertest;

import static com.google.common.base.Predicates.or;

import static springfox.documentation.builders.PathSelectors.regex;

**@Profile("swagger-enabled-for-qa")**

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket postsApi() {

return new Docket(DocumentationType.SWAGGER\_2).groupName("public-api")

.apiInfo(apiInfo()).select().paths(postPaths()).build();

}

private Predicate postPaths() {

return or(regex("/api/posts.\*"), regex("/api/javainuse.\*"));

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder().title("JavaInUse API")

.description("JavaInUse API reference for developers")

.termsOfServiceUrl("http://javainuse.com")

.contact("javainuse@gmail.com").license("JavaInUse License")

.licenseUrl("javainuse@gmail.com").version("1.0").build();

}

}

Currently if the application is deployed then the swagger will be disabled. Go to [**http://localhost:8080/swagger-ui.html**](http://localhost:8080/swagger-ui.html).