SpringBoot Interview Q&A

1) **What is Springboot and features of spring boot?**
   - Auto configuration
   - Starter in the build tools
   - Sprint boot CLI
   - Spring initializer
   - Actuators
2) **What is the difference between Spring vs Springboot?**

Spring	SpringBoot
• Lot of boiler plate code need to implement(configure) to start the Spring Project using JDBC, JPA, Hibernate, MVC etc. • XML Configurations Required • Additional configuration is required to integrate AOP, Batch, Security	• It minimizes writing multiple boilerplate codes, XML configuration and annotation, ultimately enhancing productivity while reducing lots of development time. • Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can run. • It is opinionated and follows the convention over configuration software design paradigm • It makes it easier to integrate the Spring Boot Application with the Spring Ecosystem that majorly includes Spring ORM, Spring JDBC, Spring Security, Spring Data and many other things.
Developer manually define dependencies for Spring project in pom.xml which is tedious job when Spring project upgrade to new version of the Spring Framework	• Spring Boot enables you to rapidly build and create Spring applications utilizing the Spring framework. • SpringBoot comes with concept of starter in pom.xml file which internally take care of downloading the dependencies jars based on your SpringBoot requirement.
To test the Spring project, Server environment setup needs to be done explicitly	• Spring Boot offers embedded http servers such as Jetty, Tomcat, etc. where developer deploy the code and test the functionality which improves the productivity of the developer • It offers Command Line Interface (CLI) tool for develop coding in Groovy to eliminate the boiler platform code.
Spring does not provide support for in memory databases	• It offers several plugins for working with embedded and in-memory databases easily (E.g. H2)

	• SpringBoot provides Actuator to monitor and manage our application • Springboot framework used to develop microservices.

3) What are the frequent Annotations used in SprintBoot?

Annotation	Usage
@SpringBootApplication	apps to use auto-configuration, component scan and be able to define extra configuration on their "application class"
@EnableAutoConfiguration	enable SpringBoot auto-configuration mechanism
@ComponentScan	enable @Component scan on the package where the application is located
@Configuration	allow to register extra beans in the context or import additional configuration classes
@RestController	REST API implementation
@RequestMapping	To map the path to the REST API
@EnableAutoConfiguration	Automatically load configuration based on classpath jars
@Configuration --> @ConditionalOnClass @ConditionalOnMissingClass	These annotations let @Configuration classes be included based on the presence or absence of specific classes.
@ConditionalOnProperty --> havingValue and matchIfMissing	Let configuration be included based on a Spring Environment property
@ConditionalOnResource	Let configuration be included only when a specific resource is present.
@ConditionalOnWebApplication--->@ConditionalOnNotWebApplication	let configuration be included depending on whether the application is a “web application”
@EnableJms	JMS Application.
@EnableCaching	Caching abstraction.
@Test	JUnit.
@EnableRabbit	RabbitMQ.
@EnableBatchProcessing	Spring Batch.
@MessageEndpoint @EnableIntegration	Spring Integration.
@Controller @RestController @EnableWebMvc	Spring MVC + Embedded Tomcat.

@EnableWebSecurity	Spring Security.
@EnableTransactionManagement	Spring Transaction Management.

4) **How do you change tomcat default port number in Springboot?**
   In the application.properties file :
   server.port = <port number>

5) **Where do you configure application related properties?**
   We will define all the configuration related parameters for the spring boot application in application.properties file.

SpringBoot support .properties, yaml files

```
spring.main.banner-mode=off

server.port=8443
server.ssl.enabled=true
server.ssl.key-store=classpath:store.jks
server.ssl.key-password=secret
management.server.port=8080
management.server.ssl.enabled=false

# create and drop tables and sequences,
spring.jpa.hibernate.ddl-auto=create-drop

# Oracle settings
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=password
spring.datasource.driver-class-oracle.jdbc.driver.OracleDriver

# HikariCP settings
# spring.datasource.hikari.*

spring.datasource.hikari.connection-timeout=60000
spring.datasource.hikari.maximum-pool-size=5

# logging
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n
logging.level.org.hibernate.SQL=debug
#logging.level.org.hibernate.type.descriptor.sql=trace
logging.level.=error
```

6) **How do you provide context-path for your application?**
   You can define the context path for your application using `server.servlet.context-path` in application.properties file
   `server.servlet.context-path = /my-custom-path`
7) **How do you define profiles in springboot?**

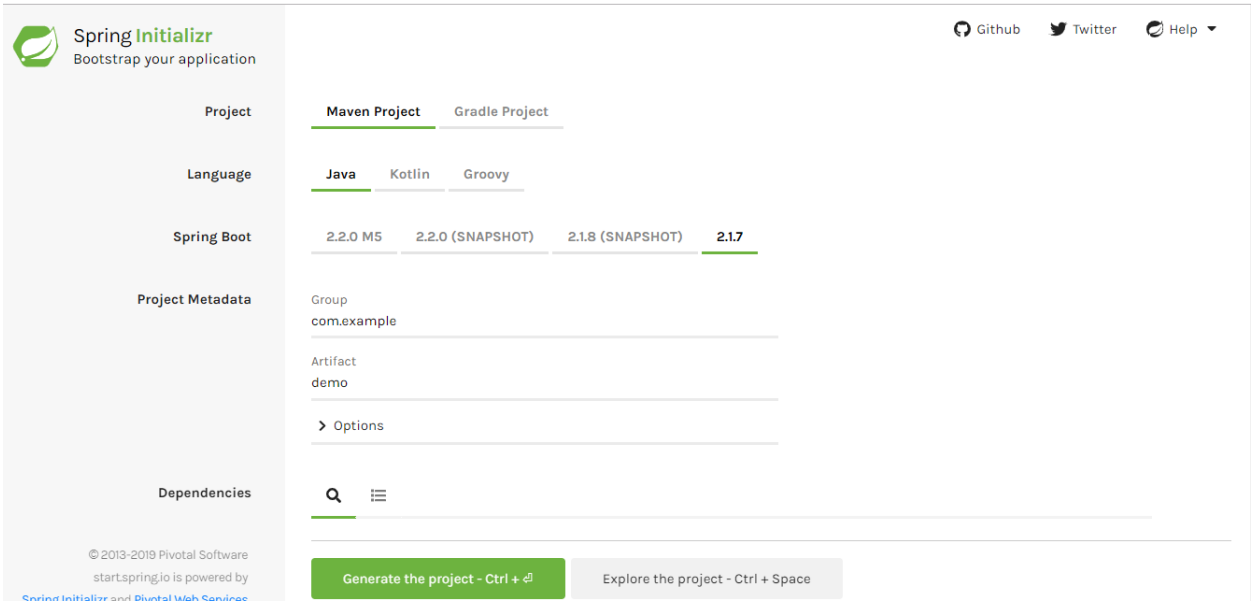
Spring Profiles provide a way to segregate parts of your application configuration and make it be available only in certain environments.
Any `@Component` or `@Configuration` can be marked with `@Profile` to limit when it is loaded

```
@Configuration
@Profile("production")
publicclass ProductionConfiguration {

        // ...

}
```

```
spring.profiles.active=prod
```

If you are running through command line using maven/gradle/java then mustpass command line property as-**Dspring.profiles.active=<profile_name(dev/stage/prod)>**

8) **Spring Initializer (**https://start.spring.io/)?



9) **What are Springboot starters and why we need it?**
starters contain a lot of the dependencies that need to get a project up and running quickly and with a consistent, supported set of managed transitive dependencies.

Name	Description
spring-boot-starter	Core starter, including auto-configuration support, logging and YAML

spring-boot-starter-activemq	Starter for JMS messaging using Apache ActiveMQ
spring-boot-starter-amqp	Starter for using Spring AMQP and Rabbit MQ
spring-boot-starter-aop	Starter for aspect-oriented programming with Spring AOP and AspectJ
spring-boot-starter-artemis	Starter for JMS messaging using Apache Artemis
spring-boot-starter-batch	Starter for using Spring Batch
spring-boot-starter-cache	Starter for using Spring Framework's caching support
spring-boot-starter-cloud-connectors	Starter for using Spring Cloud Connectors which simplifies connecting to services in cloud platforms like Cloud Foundry and Heroku
spring-boot-starter-data-cassandra	Starter for using Cassandra distributed database and Spring Data Cassandra
spring-boot-starter-data-cassandra-reactive	Starter for using Cassandra distributed database and Spring Data Cassandra Reactive
spring-boot-starter-data-couchbase	Starter for using Couchbase document-oriented database and Spring Data Couchbase
spring-boot-starter-data-couchbase-reactive	Starter for using Couchbase document-oriented database and Spring Data Couchbase Reactive
spring-boot-starter-data-elasticsearch	Starter for using Elasticsearch search and analytics engine and Spring Data Elasticsearch
spring-boot-starter-data-jdbc	Starter for using Spring Data JDBC
spring-boot-starter-data-jpa	Starter for using Spring Data JPA with Hibernate
spring-boot-starter-data-ldap	Starter for using Spring Data LDAP
spring-boot-starter-data-mongodb	Starter for using MongoDB document-oriented database and Spring Data MongoDB
spring-boot-starter-data-mongodb-reactive	Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive
spring-boot-starter-data-neo4j	Starter for using Neo4j graph database and Spring Data Neo4j
spring-boot-starter-data-redis	Starter for using Redis key-value data store with Spring Data Redis and the Lettuce client
spring-boot-starter-data-redis-reactive	Starter for using Redis key-value data store with Spring Data Redis reactive and the Lettuce client
spring-boot-starter-data-rest	Starter for exposing Spring Data repositories over REST using Spring Data REST
spring-boot-starter-data-solr	Starter for using the Apache Solr search platform with Spring Data Solr
spring-boot-starter-freemarker	Starter for building MVC web applications using FreeMarker views
spring-boot-starter-groovy-templates	Starter for building MVC web applications using Groovy Templates views
spring-boot-starter-hateoas	Starter for building hypermedia-based RESTful web application with Spring MVC and Spring HATEOAS

spring-boot-starter-integration	Starter for using Spring Integration
spring-boot-starter-jdbc	Starter for using JDBC with the HikariCP connection pool
spring-boot-starter-jersey	Starter for building RESTful web applications using JAX-RS and Jersey. An alternative to spring-boot-starter-web
spring-boot-starter-jooq	Starter for using jOOQ to access SQL databases. An alternative to spring-boot-starter-data-jpa or spring-boot-starter-jdbc
spring-boot-starter-json	Starter for reading and writing json
spring-boot-starter-jta-atomikos	Starter for JTA transactions using Atomikos
spring-boot-starter-jta-bitronix	Starter for JTA transactions using Bitronix
spring-boot-starter-mail	Starter for using Java Mail and Spring Framework's email sending support
spring-boot-starter-mustache	Starter for building web applications using Mustache views
spring-boot-starter-oauth2-client	Starter for using Spring Security's OAuth2/OpenID Connect client features
spring-boot-starter-oauth2-resource-server	Starter for using Spring Security's OAuth2 resource server features
spring-boot-starter-quartz	Starter for using the Quartz scheduler
spring-boot-starter-security	Starter for using Spring Security
spring-boot-starter-test	Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito
spring-boot-starter-thymeleaf	Starter for building MVC web applications using Thymeleaf views
spring-boot-starter-validation	Starter for using Java Bean Validation with Hibernate Validator
spring-boot-starter-web	Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container
spring-boot-starter-web-services	Starter for using Spring Web Services
spring-boot-starter-webflux	Starter for building WebFlux applications using Spring Framework's Reactive Web support
spring-boot-starter-websocket	Starter for building WebSocket applications using Spring Framework's WebSocket support