**Spring and Spring Boot**

**Frequently Asked Questions**

1. What is Spring?

Spring framework can be used in normal java applications to achieve loose coupling between different components by implementing dependency injection and we can perform cross cutting tasks such as logging and authentication using spring support for aspect oriented programming.

2. What are the advantages of spring framework?

Spring Framework is built on top of two design concepts – Dependency Injection and Aspect Oriented Programming.

Some of the advantages of using Spring Framework are:

* Reducing direct dependencies between different components of the application, usually Spring IoC container is responsible for initializing resources or beans and inject them as dependencies.
* Writing unit test cases are easy in Spring framework because our business logic doesn’t have direct dependencies with actual resource implementation classes. We can easily write a test configuration and inject our mock beans for testing purposes.
* Reduces the amount of boiler-plate code, such as initializing objects, open/close resources. I like JdbcTemplate class a lot because it helps us in removing all the boiler-plate code that comes with JDBC programming.
* Spring framework is divided into several modules, it helps us in keeping our application lightweight. For example, if we don’t need Spring transaction management features, we don’t need to add that dependency in our project.
* Spring framework support most of the Java EE features and even much more. It’s always on top of the new technologies, for example there is a Spring project for Android to help us write better code for native android applications. This makes spring framework a complete package and we don’t need to look after different framework for different requirements.

3. What is the role of IOC container in spring?

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets information from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

4. What is autowiring in spring? What are the autowiring modes?

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

Autowiring can't be used to inject primitive and string values. It works with reference only.

|  |  |  |
| --- | --- | --- |
| **Sl** | **Mode** | **Description** |
| 1) | No | It is the default autowiring mode. It means no autowiring by default. |
| 2) | byName | The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method. |
| 3) | byType | The byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method. |
| 4) | Constructor | The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters. |
| 5) | Autodetect | It is deprecated since Spring 3. |

5. What are the advantages of JdbcTemplate in spring?

Spring JdbcTemplate eliminates all the below mentioned problems of JDBC API. It provides you methods to write the queries directly, so it saves a lot of work and time.

The problems of JDBC API are as follows:

* We need to write a lot of code before and after executing the query, such as creating connection, statement, closing resultset, connection etc.
* We need to perform exception handling code on the database logic.
* We need to handle transaction.
* Repetition of all these codes from one to another database logic is a time consuming task.

6. What are the AOP implementation?

AOP implementations are provided by:

* AspectJ
* Spring AOP
* JBoss AOP

7. What is the front controller class of Spring MVC?

Front Controller is a initial level of contract point for handling a request. The front controller provides a centralized entry point for that controls and manages web request handling by centralizing decision point and [controls. In](http://controls.in/) Spring MVC **org.springframework.web.servlet.DispatcherServlet** is a front controller who handles all the user request and process the request as per there mapping.

8. What does the ViewResolver class?

public interface **ViewResolver**

Interface to be implemented by objects that can resolve views by name.

View state doesn't change during the running of the application, so implementations are free to cache views.

Implementations are encouraged to support internationalization, i.e. localized view resolution.

**Ex:** [**resolveViewName**](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ViewResolver.html#resolveViewName-java.lang.String-java.util.Locale-)(java.lang.String viewName, java.util.Locale locale)

Resolve the given view by name.

### 09. Name some of the important Spring Modules.

Some of the important Spring Framework modules are:

* **Spring Context** – for dependency injection.
* **Spring AOP** – for aspect oriented programming.
* **Spring DAO** – for database operations using DAO pattern
* **Spring JDBC** – for JDBC and DataSource support.
* **Spring ORM** – for ORM tools support such as Hibernate
* **Spring Web Module** – for creating web applications.
* **Spring MVC** – Model-View-Controller implementation for creating web applications, web services etc.

### 10. What is a Spring Bean?

Any normal java class that is initialized by Spring IoC container is called Spring Bean. We use Spring ApplicationContext to get the Spring Bean instance.

Spring IoC container manages the life cycle of Spring Bean, bean scopes and injecting, any required dependencies in the bean.

### 11. What are some of the important Spring annotations you have used?

Some of the [Spring annotations](https://www.journaldev.com/16966/spring-annotations) that I have used in my project are:

**@Controller** – for controller classes in Spring MVC project.

**@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through [Spring MVC RequestMapping Annotation Examples](https://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example)

**@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.

**@PathVariable** – for mapping dynamic values from the URI to handler method arguments.

**@Autowired** – for autowiring dependencies in spring beans.

**@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

**@Service** – for service classes.

**@Scope** – for configuring scope of the spring bean.

**@Configuration**, **@ComponentScan** and **@Bean** – for java based configurations.

AspectJ annotations for configuring aspects and advices,

**@Aspect**, **@Before**, **@After**, **@Around**, **@Pointcut** etc.

### 12. What is Spring MVC Interceptor and how to use it?

Spring MVC Interceptors are like Servlet Filters and allow us to intercept client request and process it. We can intercept client request at three places – **preHandle**, **postHandle** and **afterCompletion**.

We can create spring interceptor by implementing HandlerInterceptor interface or by extending abstract class **HandlerInterceptorAdapter**.

We need to configure interceptors in the spring bean configuration file. We can define an interceptor to intercept all the client requests or we can configure it for specific URI mapping too.

### 13. What is Spring Boot Initializr?

Spring Boot Initializr is a Spring Boot tool to bootstrap Spring Boot or Spring Applications very easily.

Spring Boot Initializr comes in the following forms:

* Spring Boot Initializr with Web Interface
* Spring Boot Initializr with IDEs/IDE Plugins
* [Spring Boot Initializr with Spring Boot CLI](https://www.journaldev.com/8609/spring-boot-initilizr-with-spring-boot-cli)
* [Spring Boot Initializr with ThirdParty Tools](https://www.journaldev.com/8650/spring-boot-initilizr-with-thirdparty-tools)

### 14. Why we need Spring Boot Initializr?

Spring Boot Initializr simplifies Spring Applications Development by providing initial project structure and build scripts.

* It reduces Development time
* It increases Productivity

15. What are the advantages of using Spring Boot?

* It is very easy to develop Spring Based applications with Java or Groovy.
* It reduces lots of development time and increases productivity.
* It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
* It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
* It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
* It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
* It provides CLI (Command Line Interface) tool to develop and test Spring Boot (Java or Groovy) Applications from command prompt very easily and quickly.
* It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle
* It provides lots of plugins to work with embedded and in-memory Databases very easily.

16. Why is it “opinionated”?

It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort. Due to opinionated view of spring boot, what is required to get started but also we can get out if not suitable for application.  
• Spring Boot uses sensible defaults, “opinions”, mostly based on the   
 classpath contents.  
• For example   
– Sets up a JPA Entity Manager Factory if a JPA implementation is on the classpath.  
– Creates a default Spring MVC setup, if Spring MVC is on the classpath.  
• Everything can be overridden easily – But most of the time not needed.

17. What is the difference between an embedded container and a WAR?

There is no force to go container less  
– Embedded container is just one feature of Spring Boot  
• Traditional WAR also benefits a lot from Spring Boot  
– Automatic Spring MVC setup, including DispatcherServlet  
– Sensible defaults based on the classpath content  
– Embedded container can be used during development

18. What embedded containers does Spring Boot support?  
 Spring Boot includes support for embedded Tomcat, Jetty, and Undertow servers.  
 By default the embedded server will listen for HTTP requests on port 8080.

19. What is Actuator in Spring Boot?  
 Spring Boot Actuator is a sub-project of Spring Boot. It adds several production grade services to your application with little effort on your part.  
 There are also has many features added to your application out-of-the-box for managing the service in a production (or other) environment.   
 They’re mainly used to expose different types of information about the running application – health, metrics, info, dump, env etc.

20. What is the configuration file name used by Spring Boot?  
 The configuration file used in spring boot projects is application.properties. This file is very important where we would over write all the default configurations. Normally we have to keep this file under the resources folder of the project.