Spring Framework Questions

Q1. What Is Spring Framework?

Spring is the most broadly used framework for the development of Java Enterprise Edition applications. Further, the core features of Spring can be used in developing any Java application.

We use its extensions for building various web applications on top of the Jakarta EE platform. We can also just use its dependency injection provisions in simple standalone applications.

Q2. What Are the Benefits of Using Spring?

Spring targets to make Jakarta EE development easier, so let's look at the advantages:

- **Lightweight** – There is a slight overhead of using the framework in development.
- **Inversion of Control (IoC)** – Spring container takes care of wiring dependencies of various objects instead of creating or looking for dependent objects.
- **Aspect-Oriented Programming (AOP)** – Spring supports AOP to separate business logic from system services.
- **IoC container** – manages Spring Bean life cycle and project-specific configurations
- **MVC framework** – used to create web applications or RESTful web services, capable of returning XML/JSON responses
- **Transaction management** – reduces the amount of boilerplate code in JDBC operations, file uploading, etc., either by using Java annotations or by Spring Bean XML configuration file
- **Exception Handling** – Spring provides a convenient API for translating technology-specific exceptions into unchecked exceptions.

Q3. What Spring Sub-Projects Do You Know? Describe Them Briefly.

- **Core –** a key module that provides fundamental parts of the framework, such as IoC or DI
- **JDBC –** enables a JDBC-abstraction layer that removes the need to do JDBC coding for specific vendor databases
- **ORM integration –** provides integration layers for popular object-relational mapping APIs, such as JPA, JDO and Hibernate
- **Web –** a web-oriented integration module that provides multipart file upload, Servlet listeners and web-oriented application context functionalities
- **MVC framework –** a web module implementing the Model View Controller design pattern
- **AOP module –** aspect-oriented programming implementation allowing the definition of clean method-interceptors and pointcuts

Q4. What Is Dependency Injection?

- Dependency injection, an aspect of Inversion of Control (IoC), is a general concept stating that we do not create our objects manually but instead describe how they should be created. Then an IoC container will instantiate required classes if needed.

Q5. How Can We Inject Beans in Spring?

A few different options exist in order to inject Spring beans:

- Setter injection
- Constructor injection
- Field injection

The configuration can be done using XML files or annotations.

Q6. Which Is the Best Way of Injecting Beans and Why?

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. This is because constructor injection allows injecting values to immutable fields and makes testing easier.

Q7. What Is the Difference Between BeanFactory and ApplicationContext?

BeanFactory is an interface representing a container that provides and manages bean instances. The default implementation instantiates beans lazily when *getBean()* is called.

In contrast, *ApplicationContext* is an interface representing a container holding all information, metadata and beans in the application. It also extends the *BeanFactory* interface, but the default implementation instantiates beans eagerly when the application starts. However, this behavior can be overridden for individual beans.

Q8. What Is a Spring Bean?

The Spring Beans are Java Objects that are initialized by the Spring IoC container.

Q9. What Is the Default Bean Scope in Spring Framework?

By default, a Spring Bean is initialized as a *singleton*.

Q10. How to Define the Scope of a Bean?

In order to set Spring Bean's scope, we can use *@Scope* annotation or "scope" attribute in XML configuration files. Note that there are five supported scopes:

- **Singleton**
- **Prototype**
- **Request**

- **Session**

Q11. Name Some of the Design Patterns Used in the Spring Framework?

- **Singleton Pattern** – singleton-scoped beans
- **Factory Pattern** – Bean Factory classes
- **Prototype Pattern** – prototype-scoped beans
- **Adapter Pattern** – Spring Web and Spring MVC
- **Proxy Pattern** – Spring Aspect-Oriented Programming support
- **Template Method Pattern** – *JdbcTemplate*, *HibernateTemplate*, etc.
- **Front Controller** – Spring MVC *DispatcherServlet*
- **Data Access Object** – Spring DAO support
- **Model View Controller** – Spring MVC

Q12. How Does the Scope Prototype Work?

Scope *prototype* means that every time we call for an instance of the Bean, Spring will create a new instance and return it. This differs from the default *singleton* scope, where a single object instance is instantiated once per Spring IoC container.

Q13. What Is a Controller in Spring MVC?

Simply put, all the requests processed by the *DispatcherServlet* are directed to classes annotated with *@Controller*. Each controller class maps one or more requests to methods that process and execute the requests with provided inputs.

Q14. How Does the *@RequestMapping* Annotation Work?

The *@RequestMapping* annotation is used to map web requests to Spring Controller methods. In addition to simple use cases, we can use it for mapping of HTTP headers, binding parts of the URI with *@PathVariable,* and working with URI parameters and the *@RequestParam* annotation.

Q15. How Does the *@PathVariable* Annotation Work?

The Spring framework provides two main annotations for this purpose: @PathVariable and @RequestParam. The @PathVariable annotation is used to retrieve data from the URL path. By defining placeholders in the request mapping URL, you can bind those placeholders to method parameters annotated with @PathVariable.

Q16. How Does the @RequestParam Annotation Work?

@RequestParam annotation tells a method parameter to bind to a web request parameter. It can extract query parameters from a request URL, or form data, and bind them to a controller's method argument. Query parameters are typically added to the URL after a question mark (?), and separated by ampersands (&). For example, in the URL http://example.com/api/products?id=123&name=Laptop, id and name are query parameters