Search...

# Spring Interview Questions and Answers

Last Updated : 23 Jul, 2025

Spring is a widely used framework in the Java ecosystem, known for its flexibility, powerful features, and ability to support enterprise-level applications. It's the preferred choice for top companies like **Uber**, **Google**, **Netflix**, and **Amazon** due to its robustness and performance.

In this interview preparation guide, we will provide you with the top Spring Java interview questions and answers for 2024, catering to both freshers and experienced professionals (2, 5, or 10 years). It covers essential concepts and advanced topics, offering valuable insights to help you prepare for your next interview.

**Spring Framework** is an open-source, lightweight, and easy-to-use framework that can be considered a framework of frameworks containing various frameworks, including the topics mentioned below.

> ***Beginners:*** *Spring fundamentals , Spring Dependency Injection, and Spring Bean*
> ***Experienced:*** *AOP in Spring, Spring MVC, Spring JDBC , Spring Hibernate, Spring Web Service, EJB (Enterprise JavaBeans), and JSF (JavaServer Faces).*

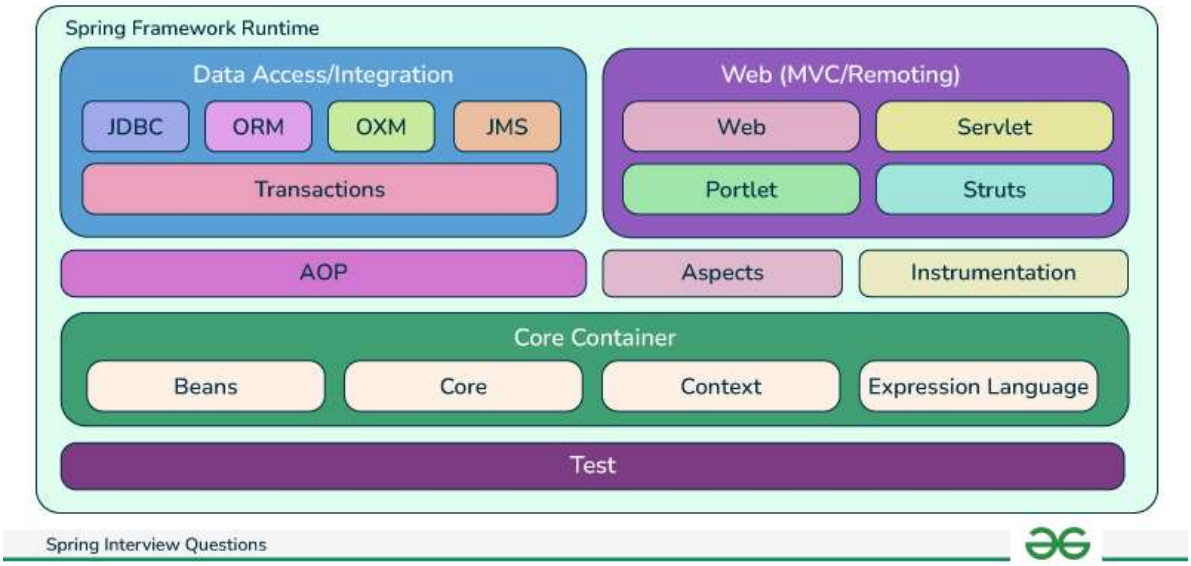
## Table of Content

# Spring, Spring Core, Spring IoC Interview Questions

## 1. What is Spring Framework

Spring framework is an open-source Java framework that supports building robust Java applications. It mainly handles all the infrastructure-related aspects allowing the developer to focus more on application development, making it the world's most popular Java framework.

- Spring is used in every domain, even in big techs like Amazon, Google, etc.
- Features like *IoC* and *DI* provide a set of features and functionality.
- Increased productivity as redundant configuration is not required.
- Enormous community support.




## 2. Overview of versions of Spring Framework

- **Spring 2.5** (2007): support for annotations was introduced, reducing the need for XML configurations.
- **Spring 3.0** (2009): Introduction of Spring Expression language, profile for environment-specific configurations.
- **Spring 4.0** (2013): Added support for Java 8, and introduced Spring Websocket module.

- **Spring 5.0** (2017): Added Spring WebFlux module for reactive programming, Supports Kotlin development.
- **Spring 6.0** (2022): Adopted Java 17, Jakarta EE 9+, and enhanced support for cloud-native and reactive applications.
- **Spring 6.1** (2023): Improved performance, refinements in AOT processing, and optimizations for native images.
- **Spring 6.2** (2024): Further enhancements to cloud integration, modularization improvements, and better support for virtual threads.

### 3. What are the features of Spring Framework?

- Modular Design
- Dependency Injection
- Aspect-oriented programming
- Transaction management
- Data access
- Model-View-Controller(MVC)
- Web development
- Testing
- Spring Cloud

### 4. What are the advantages of using Spring Framework

- **High Productivity:** Reduced boilerplate code(Lombok, etc), faster development(auto-config), and simplified testing(JUnit).
- **Easy to Maintain:** Loose coupling, separation of concerns, and cleaner code structure.
- **Security:** Dedicated security framework, built-in authentication and authorization, and data protection features.
- **Large Community Support:** Large and active community support, documentation support, etc.

### 5. How do we configure our Spring Application?

- **Java Annotations:** Clean and concise, but limited flexibility for large applications.
- **XML Configuration Files:** Centralized lengthy configurations, but wordy and less maintainable compared to annotations.
- **Java Configuration Classes:** Code-based configuration support
- **Property Sources:** Decouples configuration from code, allows dynamic changes but requires additional management.
- **Spring Boot:** Simplifies configuration and application development, but may not be suitable for complex applications.

## 6. Explain Inversion of Control(IoC) and types of IoC containers.

IoC stands for Inversion of Control means transferring the control of managing the dependencies and their injection when required from the application to the container/framework, It increases code scalability, maintainability, and easy testing.

**Types of IoC Containers:**

- **Bean Factory:** Basic container, that provides basic object creation and dependency injection for spring applications.
- **Application Context:** Advanced container, is an implementation of BeanFactory. Can manage object lifecycles, events, and resource access.


## 7. Explain Dependency Injection(DI) and its types?

Dependency Injection is used by the framework to auto-inject the dependencies into the beans when beans are created, hence increasing developers' productivity by reducing boilerplate code.

Types of Dependency Injection-

- **Constructor injection** - injection using constructor
- **Setter injection** - using setter methods
- **Field injection** - directly into the fields

### 8. Types of Metadata in Spring Framework?

- **Annotations:** Provide information about beans and their dependencies.

```
<beans>
<context:annotation-config/>
<!-- bean definitions go here -->
</beans>
```

- **XML Configuration:** Defines bean configurations and dependencies in an XML file.

```
<beans>
    <bean id="beanService" class="com.GeeksforGeeks.beanService">
        <property name="beanService" value="Bean Service"/>
    </bean>
</beans>
```

- **Java Configuration:** Uses Java class to define bean configurations and dependencies. Best alternative for XML-based configurations
  - Mainly *@Configuration* and *@Bean* annotations are used for configuration.
- **Property Sources:** Store configuration settings in external sources like environment variables and property files.

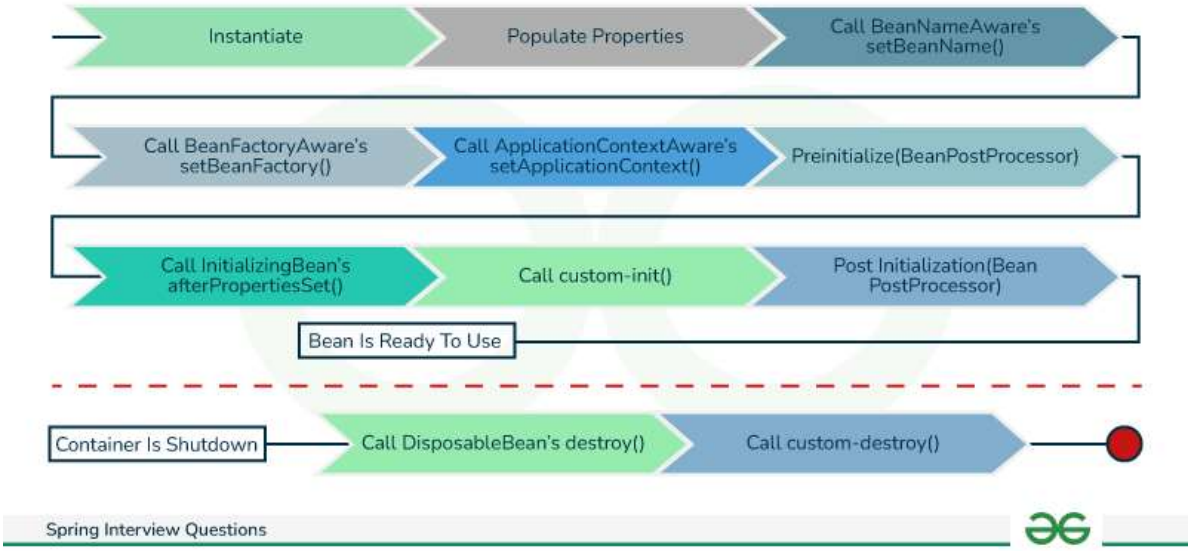
### 9. Explain Spring Beans and their scopes.

Objects managed by the Spring container, defined by configuration and annotated with *@Component*.

- **Scopes:** Define the lifecycle and lifetime of a bean:
  - **Singleton:** Single instance throughout the application.
  - **Prototype:** A new instance is created for each request.
  - **Request:** A new instance is created for each HTTP request.
  - **Session:** A new instance is created for each user session.

### 10. What do you understand by the Bean life cycle in a Spring Bean Factory Container?

- **Bean Instantiation:** Creation of bean class instance.
- **Bean Post-processing:** Use of post-processors for customizing the beans.
- **Bean Initialization:** Use of *@PostConstruct* to set up the beans using methods.
- **Bean Usage:** Injection of beans for application-wide use
- **Bean Destruction:** Destroys the bean through methods annotated with *@PreDestroy*




### 11. Explain Autowiring and its types.

Autowiring reduces the efforts of object instantiation by auto injection of dependencies into beans managed by spring.

**Types of Autowiring:**

- **No auto wiring:** Setter or constructor-based dependency injection.
- **By name:** Matches bean names with property names for injection.
- **By type:** Matches bean types with property types for injection.
- **Constructor:** Injects dependencies through the bean's constructor.

## Spring Boot Interview Questions

### 12. Explain Spring Boot and its advantages

Spring Boot framework is built to simplify the process of development and deployment for Spring applications. It reduces the configuration efforts by auto-configuring beans and contiainers making the development faster and more efficient.

**Advantages of Spring Boot:**

- **Reduced configuration:** Reduces XML configuration files, making development configuration easy.
- **Automatic configuration:** Automatically configures beans and management of dependencies.
- **Embedded server:** *Tomcat* is a built-in server that reduces the effort for separate server configuration.
- **Starter POMs:** Provides pre-configured starter POMs that simplify dependency management.
- **Rapid application development:** Enables rapid application development by simplifying the development process and reducing configuration overhead.
- **Cloud-friendly:** Best choice for cloud deployment with support across various cloud platforms.

### 13. Differentiate between Spring and Spring Boot

| Features | Spring | Spring Boot |
|---|---|---|
| **Focus** | Framework for building Java application | Simplifies Spring application development |
| **Configuration** | Extensive XML configuration | Minimal configuration is required, auto-config based on dependencies |
| **Server** | A separate configuration of the server is required | Embedded server for independent execution |
| **Development focus** | customization | Easy to use and application development is easy |
| **Application type** | suitable for microservice and monolithic | best for microservice architecture supports monolithic as well. |

### 14. Explain some of the most used Spring Boot annotations

- ***@SpringBootApplication:*** Used to denote a class as Spring Boot application, it contains features of three annotations listed.

```
@SpringBootApplication =  +
```

- ***@Configuration***: Used to declare a class as a configuration setting for bean definitions.
- ***@Component***: Used to mark the class as Spring bean and its management is handled by Spring container.
- ***@Autowired:*** It injects the required dependencies into a bean automatically.
- ***@RestController:*** Used to denote a class as a REST controller.

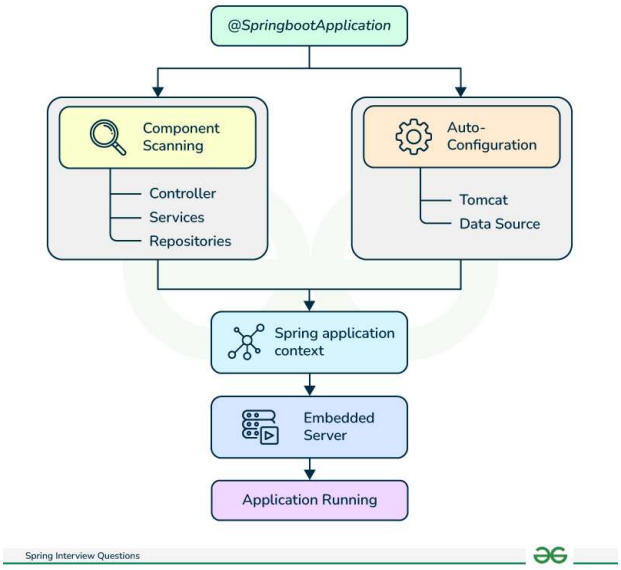
```
@RestController = @ResponseBody + @Controller
```

- ***@Bean:*** Used for the creation of beans and registering them with the Spring container for auto management.

## 15. Explain the internal working of *@SpringBootApplication*

The *@SpringBootApplication* annotation performs several tasks as it contains features of three annotation will are *@Configuration*, *@EnableAutoConfiguration*, and *@ComponentScan.*

- *@SpringBootApplication* enables component scanning for beans and auto-configuration.
- Registers various beans, including Spring Boot auto-configuration beans.
- Instantiates the embedded server
- Runs the application




## 16. Explain types of configuration in Spring Boot

Spring Boot uses a layered approach to configuration:

- **Default configuration:** Defaults configuration with available libraries and dependencies.

- **Custom configuration:** It can override default configuration through
  - properties files
  - environment variables
  - annotations
- **External configuration:** import configuration from external sources like
  - Git repositories
  - cloud platforms

### 17. Explain the role of the Tomcat server in the Spring Boot Application

Spring Boot includes an embedded Tomcat server by default. This server is responsible for:

- Receiving and processing HTTP requests acts as a bridge between the user and the application.
- Managing web resources like HTML files, and JSPs for dynamic content generation.
- Built-in authentication which can be modified by Spring Security.

### 18. What are Profiles in Spring Boot

In Spring Boot, Profiles allow configuration for applications differently in different environments, such as

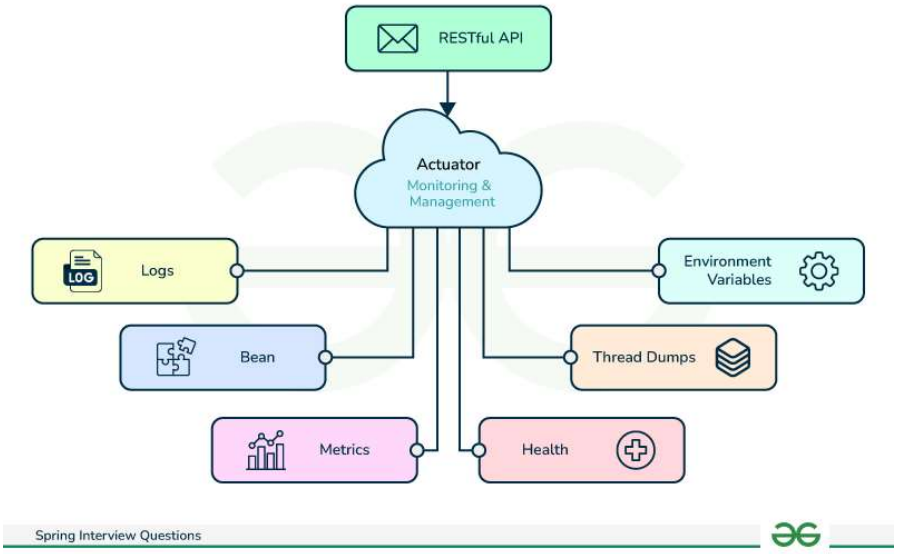
- Development
- Staging
- Production

Separate configuration files are defined for each profile, which can be activated using environment variables or command-line arguments.

### 19. What is an Actuator and its usage in Spring Boot

Spring Boot Actuator provides a RESTful API for monitoring and managing Spring Boot applications. These endpoints provide information about applications that can be used to optimize resources and debug issues, including:

- Environment variables
- Thread dumps
- Health checks
- Metrics
- Beans
- Logs




## AOP, Hibernate, JDBC Interview Question

### 20. What is Spring AOP and proxy pattern?

Aspect-oriented programming (AOP) is a design pattern that helps us manage aspects like logging, security, and transaction management in applications. Spring AOP provides an implementation of AOP using dynamic proxies.

The proxy pattern is a software design pattern that creates a proxy object that intercepts incoming requests and controls access to another object

before reaching the bean. In Spring AOP, dynamic proxies are used to implement aspects.

### 21. Explain key components of AOP.

- **Aspect:** A building block bundles together cross-cutting concerns. It has two main parts i.e. advice and pointcut.
  - **Advice:** The code that is executed before, after, or around a method invocation.
  - **Pointcut:** Condition triggering the tasks(advice).
- **Join point:** A specific point in the program execution where an aspect can be applied. Common join points are
  - method calls
  - field access
  - object creation
- **Weaving:** Spring supports *weaving* at compile, load, and runtime for integrating *aspects* in the application at *join points*.

### 23. Differentiate between Spring AOP and AspectJ AOP?

| Feature | Spring AOP | AspectJ AOP |
|---|---|---|
| **Programming model** | Annotation or XML configuration supported | Dedicated AspectJ compiler |
| **Weaving** | Dynamic proxy weaving at runtime | runtime weaving supported |
| **Supported features** | Aspect composition, pointcuts, advice, etc | control flow join and aspect inheritance |

### 24. What are the advantages of AOP and its implementation?

AOP helps to maintain, modify, and understand code easily,

- **Modularization:** separation of concerns like logging, security, etc from core business logic, to increase maintainability.
- **Reusability:** Bundles concerns in reusable aspects, improving code reusability.
- **Interception:** Allows interception and modification of method calls, enabling features like logging, security, and caching.

### 25. Explain Hibernate ORM and ways to access it in Spring.

- **Hibernate ORM:** Hibernate is an object-relational mapping (ORM) framework, that provides a bridge between Java objects and relational database tables. Overall no need to write SQL queries manually. Hibernate works by
  - Persistence Context
  - Mapping
  - Session factory
  - Session
- **Access in Spring:** Spring provides several ways to integrate with Hibernate:
  - HibernateTemplate (legacy)**:** Less preferred choice, It facilitates simpler data access through methods like get, load, and save.
  - Spring Data JPA**:** Recommended way, It simplifies data access using JPA annotations.
  - Direct JDBC Template**:** preferred choice for advanced scenarios, provides more control over data access.

### 26. Explain Hibernate Validator Framework and HibernateTemplate class?

- **Hibernate Validator Framework:** Provides validation against defined constraints, and prevents invalid data from entering the application. A few examples are listed-

    - @NotNull
    - @Size
    - @Email
- **HibernateTemplate class:** Provides an interface for data access operations like the one below, without writing SQL queries.
    - get
    - load
    - save
    - update
    - delete

### 27. Explain Spring JDBC API and its classes.

- **Spring JDBC:** Spring provides a simple way in the form of a JDBC abstraction layer to establish a bridge between database and application. It reduces boilerplate code and configurations.
- **Key classes:**
    - JdbcTemplate: Provides simple methods for executing SQL statements and working with data exchange for applications.
    - DataSource: Establish the connection(bridge) of data exchange from database.
    - SimpleJdbcCall: method present in Spring JDBC API, used for interacting with database-stored procedures.

### 28. What are the advantages of JdbcTemplate in Spring?

- **Reduces boilerplate code:** no need to write raw JDBC codes, also bundles common operations.
- **Exception handling:** auto handling and conversion of SQLExceptions into Spring's DataAccessException.
- **Prepared statements:** Uses prepared statements to prevent SQL injection attacks.

- **Data binding:** instead of SQL statements it uses prepared statements, which have better-
  - Security - prevents SQL injection attacks
  - Performance - improved query performance

### 29. Fetching records using Spring JdbcTemplate?

Use the query method of JdbcTemplate with the appropriate SQL query and result extractor.

```
List<User> users = jdbcTemplate.query("SELECT * FROM users", new
BeanPropertyRowMapper<>(User.class));
```

This code snippet fetches all users from the user's table and maps them to User objects using the BeanPropertyRowMapper.

## Spring MVC Interview Question


*Spring Interview Questions and Answers*

### 30. What do you understand from Spring MVC and its components?

Spring MVC is a web framework built on top of the core Spring Framework that provides a model-view-controller(MVC) architecture for building web

applications. It simplifies web development by separating business logic from presentation and handling request routing and dispatching.

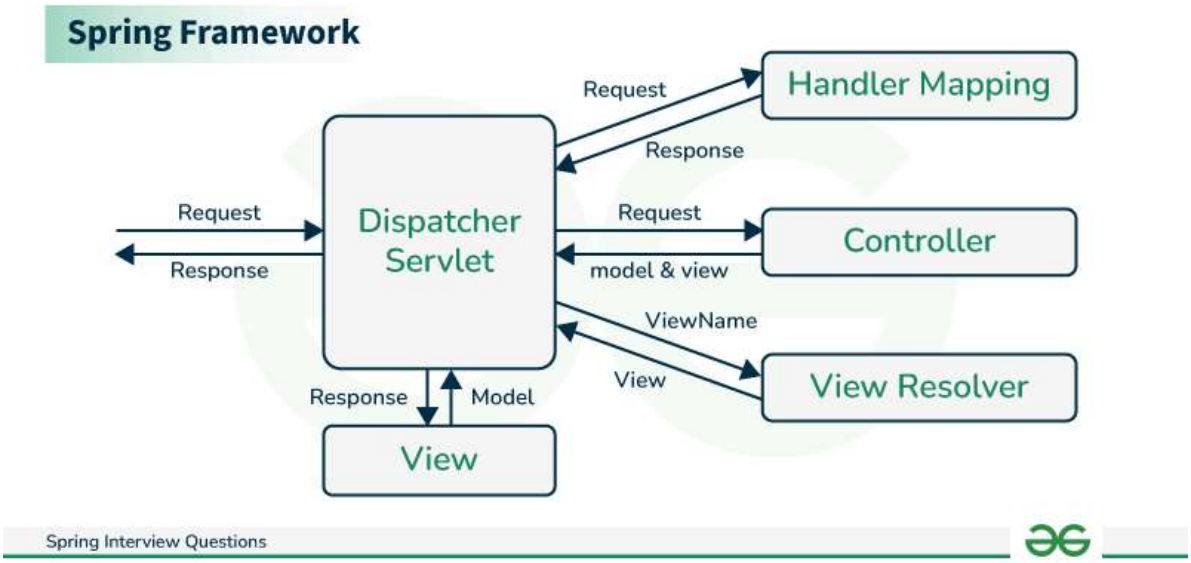
**Components:**

- **DispatcherServlet:** Receives all requests and routes them to the appropriate controller.
- **Model:** Java objects that are passed between controller and view.
- **View:** User interface for displaying the model.
- **Controller:** Central component to handle user requests and responses, from model to view.

### 31. Explain DispatcherServlet and Request Flow in Spring MVC?

It is the central component of the Spring MVC framework and acts as the front controller, receiving all incoming requests and dispatching them to relevant controllers based on the request URL and mapping configuration hence maintaining the overall request-response cycle.

- **Request Flow:**
  1. The client sends a request to the DispatcherServlet.
  2. DispatcherServlet identifies the appropriate controller based on request mapping.
  3. The controller processes the request, interacts with the model, and returns a model object.
  4. DispatcherServlet selects the appropriate view based on the returned view name.
  5. View renders the model data into the final response and sends it back to the client.




## 32. Explain Interceptors in Spring MVC?

Interceptors are reusable components that intercept request processing and response generation phases in the lifecycle of web applications. They can be used for tasks in which a concern has to be applied globally across multiple controllers like logging, authentication, caching, and authorization.

## 33. Design Patterns used in Spring MVC?

Spring MVC is built on the top of two

- **MVC Pattern:** Separates the application into three layers *i.e.* presentation, business logic, and data access layers.
- **Front Controller Pattern:** Single entry point for all incoming requests, DispatcherServlet receives it and re-directs it to appropriate controllers.
- **Template Method Pattern:** View resolvers use templates to render views with consistency in the presentation layer.
- **Strategy Pattern:** Different view resolvers can be used based on the required view technology such as
  - InternalResourceViewResolver
  - ThymeleafViewResolver

## 34. Explain the most important Spring MVC annotations

- **@Controller:** Marks the class as a controller in the Spring MVC framework, It handles and processes all the incoming requests and returns appropriate view or response.

```
@Controller
public class GeeksController {

}
```

- **@RequestMapping:** Used to map a controller method to a specific URL pattern, it can handle various HTTP methods like
  - GET
  - POST
  - PUT
  - DELETE

```
@Controller
@RequestMapping("/geeks")
public class GeeksController{

}
```

- **@ModelAttribute:** It is used to add an attribute to the model for the view.

```
@Controller
@RequestMapping("/geeks")
public class GeeksController{

        @ModelAttribute("geek")
        public Geek getGeek(){
                return service.getGeek();
        }
```

```
}
```

- **@RequestParam:** Extracts data from the request parameters into method arguments, allowing to access values present in the request URL.

```
@Controller
@RequestMapping("/geeks")
public class GeeksController{

        @RequestMapping("/get")
        public String getGeek(@RequestParm("Geek") Geek geek){
                return "geekDetails";
        }

}
```

- @PathVariable: Extracts data from the URL path into method arguments.

```
@Controller
@RequestMapping("/geeks")
public class GeeksController{

        @RequestMapping("/get/{id}")
        public String getGeek(@PathVariable("GeekId") Long id){
                return "geekDetails";
        }

}
```

- @SessionAttribute: Used in cases when model attributes are supposed to be stored across multiple requests.

## 35. Importance of session scope

Session scope plays an important role in maintaining beans for a specific duration which stores crucial information like login credentials, etc.

A few important are listed-

- Avoiding Data Repetition
- Application Security
- Reduced Database Access

### 36. How to get ServletConfig and ServletContext objects in Spring Bean?

Use @Autowired annotation to inject them into the bean.

- Simply declare the field and annotate it with *@Autowired*

```
@Autowired
private ServletConfig servletConfig;

@Autowired
private ServletContext servletContext;
```

### 37. Explain data validation in Spring Web MVC Framework

Spring provides various ways to validate data:

- **Bean Validation API:** Annotations like *@NotNull* and *@Size* can be used to validate bean properties.



validation based on annotations.
- **Validator interface:** Custom validation logic can be implemented using the Validator interface.

### 38. Differentiate between a Bean Factory and an Application Context.

- **Bean Factory:** Creates and manages beans.

- **Application Context:** Provides additional features like event handling, internationalization, and resource management beyond basic bean management.

### 39. What is i18n and localization in Spring MVC

Spring MVC supports *i18n* and localization, allowing you to develop applications that can be adapted to different languages and cultural contexts.

### 40. Exception Handling in Spring MVC

Spring MVC provides various mechanisms for handling exceptions:

- **@ExceptionHandler annotation:** Defines methods to handle specific exceptions.
- **Global exception handler:** Handles all uncaught exceptions.
- **Error pages:** Customized error pages can be displayed for different HTTP error codes.

### 41. What is ViewResolver class

ViewResolver is responsible for resolving the view name returned by the controller to the actual view implementation.

### 42. What do you understand by MultipartResolver?

MultipartResolver handles file uploads in Spring MVC applications. It parses multipart requests and extracts uploaded files

## Spring 5 Interview Questions (Reactive Programming)

### 43. What Is Spring WebFlux and its types?

Spring Webflux is used to develop applications with faster response time and improved scalability. It uses the principles of reactive programming

and non-blocking APIs to handle asynchronous requests of data streams.

**Types of Spring WebFlux:**

- **Functional:** Developers use lambdas and streams to create reactive applications.
- **Annotation-based:** follows Spring MVC style for configuring controllers, handlers, and filters.

## 44. What is Spring Reactive Web?

Spring Reactive Web is a sub-framework within Spring WebFlux that provides functionalities for building reactive web applications. It includes components like:

Reactive programming is used for developing high scalability and responsive web applications for handling asynchronous and non-blocking operations efficiently, to provide these features Spring WebFlux provides a sub-framework Spring Reactive Web.

It has a few components-

- **WebClient:** It is used for making Non-blocking HTTP requests
- **Server-Sent Events (SSE):** These provide the feature of real-time communication between server and client.
- **WebSocket:** Used for applications with interactivity such as Chat applications, etc.

## 45. What are Reactive Streams API?

Reactive Streams API provides a foundational building block for asynchronous data processing in Reactive. It defines a set of interfaces and methods for publishers, subscribers, and subscriptions, enabling interoperability between different reactive libraries.

Also, It has features like-

- publisher-subscriber model

- backpressure handling
- resilient systems in the modern software landscape

### 46. Different types of resources or media types supported by Spring WebFlux

Spring WebFlux supports various media types for request and response data, including:

- JSON
- XML
- Plain text
- HTML
- Multipart/form-data
- Custom media types

### 47. Exception handling in Spring Webflux?

Spring WebFlux provides various ways to handle exceptions:

- **GlobalExceptionHandler:** Handles all uncaught exceptions in the application.
- **WebExceptionHandler:** Handles exceptions specific to web requests.
- **ReactiveExceptionHandler:** Handles exceptions specific to reactive streams.

## Conclusion

In conclusion, You are preparing for a Spring interview requires a solid understanding of core Spring concepts such as inversion of control, dependency injection, and Spring MVC framework and It's essential to be able to articulate your knowledge effectively and demonstrate practical experience through projects or hands-on practice.

Additionally, staying updated with the latest features and advancements in the Spring ecosystem can give you a competitive edge. By mastering these fundamentals and staying current with industry trends, you'll be

well-equipped to ace your Spring interview and excel in your career as a developer.

Comment | More info | Advertise with us