Spring MVC Most Asked Interview Questions

Q1. What is Spring MVC?

Spring MVC is a module in the Spring Framework that implements the Model-View-Controller design pattern for building web applications. It simplifies the development process by providing a flexible and configurable way to handle HTTP requests and responses.

Q2. Explain the flow of Spring MVC.

In Spring MVC, a request is first received by the DispatcherServlet, which acts as a front controller. It then delegates the request to a handler based on mappings. The handler processes the request, populating the model if necessary, and returns a view name. The DispatcherServlet then renders the view using the model data.

Q3. What is Dispatcher Servlet.

The DispatcherServlet is a central servlet in Spring MVC that dispatches requests to various handlers, coordinates the workflow by working with different components, and handles web application configuration elements like view resolution, locale, theme resolution, and more.

Q4. What is the use of @Controller annotation.

The @Controller annotation in Spring MVC is used to mark a class as a Spring Web controller. This annotation allows the Spring container to automatically detect the annotated class and register it as a controller in the application context, enabling it to handle requests.

Q5. What is the use of InternalResourceViewResolver?

InternalResourceViewResolver in Spring MVC helps in resolving views by prefixing and suffixing the view name with a specified string. It primarily resolves views to JSPs or other resource types within the application.

Q6. Difference between @RequestParam and @PathVariable.

@RequestParam is used to extract query parameters from the request URL, while @PathVariable is used to extract values from the URI path within a URL. For example, @RequestParam reads "id" from `/user?id=123`, while @PathVariable reads "id" from `/user/123`.

Q7. Explain @Service and @Repository annotations.

@Service and @Repository are stereotypes in Spring, marking classes at the service layer and repository (or DAO) layer, respectively. @Service is used for service classes, and @Repository is used for database access classes, enabling Spring's data handling support and exception translation.

Q8. What is the purpose of @ModelAttribute?

The @ModelAttribute annotation is used to bind method parameters or method return values to a named model attribute, exposed to a web view. It can also be used to automatically populate model attributes with data from HTTP requests.

Q9. Explain @RequestBody annotation.

The @RequestBody annotation is used to bind HTTP request bodies with a method parameter in controller functions. It allows you to consume data in various formats like JSON or XML and automatically deserialize it into a Java object.

Q10. What is the use of BindingResult.

BindingResult is used to handle and retrieve validation errors after Spring has bound web request parameters to an object. It is passed as a parameter immediately after the model attribute in controller methods and can check for errors in the object fields.

Q11. How does Spring MVC support for validation.

Spring MVC supports validation through the JSR-303/JSR-349 Bean Validation APIs. By annotating fields of a model class with standard annotations like @NotNull, @Size, etc., and using a Validator implementation, Spring can automatically ensure that model attributes adhere to defined rules before processing them.

Q12. Explain @GetMapping and @PostMapping.

@GetMapping and @PostMapping are specialized versions of the @RequestMapping annotation that specifically handle HTTP GET and POST requests, respectively. They reduce the configuration overhead by providing a shortcut to define handler methods for these HTTP methods.

Q13. How to send the data from Controller to UI.

Data can be sent from a Controller to the UI by adding attributes to the Model object or using a ModelAndView object. These attributes are then accessed in the view layer, typically through JSPs or other view templates, to render the response.

Q14. What is the purpose of query parameter?

Query parameters in URLs are used to pass small amounts of data from client to server. They are part of the URL and help in filtering resources, managing sessions, tracking user activities, and more, enhancing the interactivity of HTTP requests.

Q15. Describe the annotations to validate the form data.

To validate form data, annotations like @NotNull, @Min, @Max, @Size, and @Pattern can be used. These annotations are part of the Java Bean Validation API, which Spring integrates to automatically apply validation rules before processing the form submission.

Q16. What do you know about Thymeleaf?

Thymeleaf is a modern server-side Java template engine for web and standalone

environments. It is often used with Spring MVC as it provides a natural templating capability that can process HTML, XML, JavaScript, CSS, and text.

Q17. Explain the use of @ResponseBody annotation.

The @ResponseBody annotation tells a controller that the return type should be written directly to the HTTP response body. This is useful when you want to send a response in a raw or JSON format directly from a controller method.

Q18. What is the role of Handler Mapper in Spring MVC.

HandlerMapper is responsible for mapping HTTP requests to handler objects (controllers). It interprets the URL or headers of a request and directs it to the appropriate controller based on mappings defined in the application.

Q19. How will you map the incoming request to a Controller method.

Incoming requests are mapped to controller methods by using annotations such as @RequestMapping, @GetMapping, and @PostMapping, which specify the URL patterns and HTTP methods that the methods are responsible for handling.

Q20. How to bind the form data to Model Object in Spring MVC.

Form data is bound to model objects in Spring MVC using @ModelAttribute annotation. This automatically populates a model object with request parameters matching the object's field names.

Q21. What is Spring MVC Tag library.

The Spring MVC Tag Library is a collection of custom JSP tags that integrate with the Spring MVC framework, providing functionalities like data binding, themes, and form handling within JSP files, simplifying the development of view components.

Q22. Difference between @Controller and @RestController annotations.

@Controller is typically used for view-oriented applications where the controller returns a view. @RestController combines @Controller and @ResponseBody, simplifying the creation of RESTful web services by handling responses without relying on view resolvers.

Q23. Explain few tags in Spring MVC tag library.

In the Spring MVC tag library, tags like `<form:form>`, `<form:input>`, `<form:errors>` are commonly used. `<form:form>` binds a form to a model attribute, `<form:input>` binds input fields to model attributes, and `<form:errors>` displays validation errors.

Q24. Explain the use of @ResponseEntity annotation.

@ResponseEntity is used to create a full HTTP response, including status code, headers, and body. It's useful in RESTful controllers where you need to provide detailed responses directly from your methods.

Q25. How to handle exceptions in Spring MVC.

Exceptions in Spring MVC can be handled using the @ExceptionHandler annotation within a controller, or more globally with @ControllerAdvice. These allow for centralized exception handling, enabling a cleaner way to deal with errors and custom responses.