**Spring MVC**

**1. What is MVC?**

**MVC** refers to **Model, View,** and **Controller**. It is an architectural design pattern, which governs the application's whole architecture. It is a kind of design pattern used for solving larger architectural problems.

MVC divides a software application into three parts that are:

* **Model**
* **View**
* **Controller**

**2. What is Spring MVC?**

[**Spring MVC**](https://www.geeksforgeeks.org/spring-mvc-framework/) is a sub-framework of Spring framework which is used to build dynamic web applications and to perform Rapid Application Development (RAD).

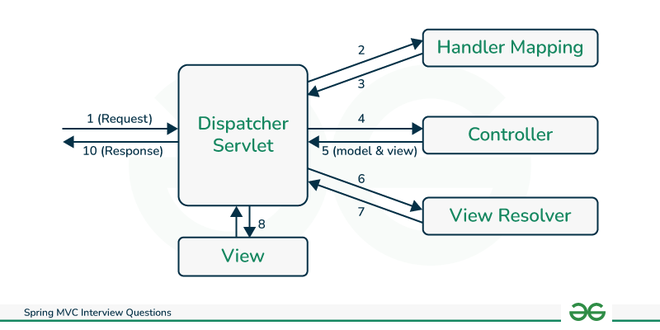
* It is built on the top of the [Java Servlet API](https://www.geeksforgeeks.org/servlet-api).
* It follows the **Model-View-Controller** Architectural design pattern.
* It implements all the basic features of the applicationscore Spring framework like **IOC (Inversion of Control)** and **Dependency Injection (DI)** etc.

**3. Difference between Spring Boot and Spring MVC**

| **Features** | **Spring Boot** | **Spring MVC** |
| --- | --- | --- |
| Build | It is a framework, that helps developers get started with Spring framework with minimal configuration. | It is a web framework built on the top of Java Servlet API. |
| Working | Using Spring Boot, it is easy to create stand-alone dynamic web applications and rapid application development. | It is a part of core Spring framework, which supports Spring's basic features and is used for building web applications using MVC architecture. |
| Productivity | Developers use Spring Boot to save time and increase productivity in developing stand-alone applications and Spring-based projects. | Developers use Spring MVC to create web applications running on a servlet container such as Tomcat. |

**4. Explain Spring MVC Architecture.**

Spring MVC Architectural Flow Diagram:



* First, the request will come in through the browser and it will be received by Dispatcher Servlet, which will act as Front Controller.
* Dispatcher Servlet will take the help of handler mapping and get to know the controller class name associated with the request.
* After this, it will transfer the request to the controller, and then the controller will process the request by executing appropriate methods based on used GET or POST method.
* And it will return the **ModelAndView** object back to the dispatcher servlet.
* Now, the dispatcher servlet sends the model object to the view resolver in xml file to get the view page.
* And finally, the dispatcher servlet will pass the model object to the view page to display the result.

**5. What are the Key Components of Spring MVC Architecture?**

Below are the Key Components of Spring MVC Architecture:

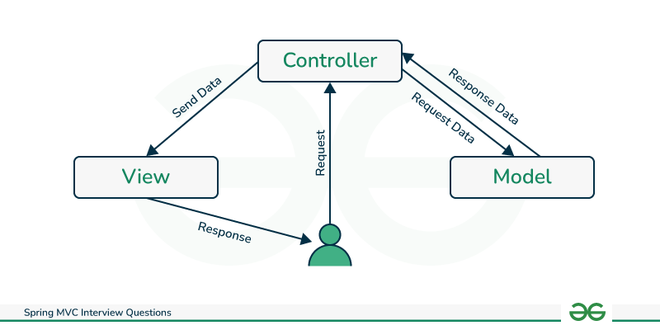
* Dispatcher Servlet
* Handler Mapping
* Controller
* Model
* View
* ViewResolver
* HandlerInterceptor
* LocaleResolver
* MultipartResolver
* WebDataBinder
* ModelAndView
* HandlerExceptionResolver

**6. Explain the Model-View-Controller (MVC) Design Pattern.**

**MVC design pattern** is a way to organize the code in our application. **MVC** refers to **Model, View, and Controller.**

* **Model** - It represents data, which is coming in our website URL as a query parameter.
* **View** - It represents the model data in a structured format (view page) which the end-users are going to see.
* **Controller** - It represents the business logic of an application, resides inside the controller and to mark a class as a controller class we use **@Controller** annotation.

Below is the Model-View-Controller Flow diagram:



**7. What is Dispatcher Servlet in Spring MVC?**

**Dispatcher Servlet** is the **Front Controller** in the Spring MVC framework. This is a class that receives the incoming HTTP requests and maps these requests to the appropriate resources such as **model**, **controller**, and **view**. Also, it sends the appropriate responses to the requests. Dispatcher Servlet manages the entire flow of an application.

**8. Explain the five most used annotations in Spring MVC Project.**

The most used five annotations in the Spring MVC project are:

* **@Controller**: This annotation is used to create classes as controller classes and parallelly it handles the HTTP requests as well.

@Controller  
 public class GfgController {  
 // write code here  
 }

* **@RequestMapping**: To map the incoming HTTP requests with the handler methods inside the controller class, we use @RequestMapping annotation.

@RestController  
 public class GfgController {  
 @RequestMapping(value = "", method = RequestMapping.GET)  
 //write code here  
 }

* **@RequestParam**: To obtain a parameter from URI (Uniform Resource Identifier), we use @RequestParam annotation.

@GetMapping("/clients)  
 public String getClients(@RequestParam(name = "clientname") String name) {  
 //write code here  
 }

* **@PathVariable**: To extract the data from the URI path, we use @PathVariable annotation.

@GetMapping("/client/{clientName}")  
 public String getClientName(@PathVariable(name = "clientName") String name) {  
 //write code here  
 }

* **@ModelAttribute**: This annotation binds method parameter and refers to the model object.

@ModelAttribute("client")  
 public Client client() {  
 //write code here  
 }

**9. What is ViewResolver in Spring MVC?**

In Spring MVC, **ViewResolver** is used to determine how a logical view name is received from a controller and maps that to an actual file. There are different types of ViewResolver classes. Some of them are defined below:

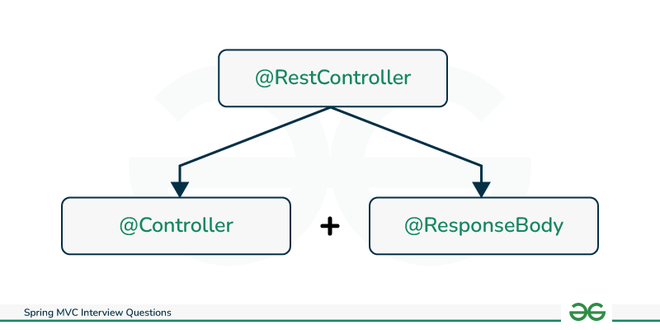
* **InternalResourceViewResolver**: It uses a prefix and suffix to convert a logical view name.
* **ResourceBundleViewResolver**: It uses view beans inside property files to resolve view names.
* **XMLViewResolver**: It also resolves view names in XML files to beans defined in the configuration file.

***Know more about***[*ViewResolver in Spring MVC*](https://www.geeksforgeeks.org/viewresolver-in-spring-mvc)

**10. Difference between *@Controller* and *@RestController***

| **Features** | **@Controller** | **@RestController** |
| --- | --- | --- |
| Usage | It marks a class as a controller class. | It combines two annotations i.e. @Controller and @ResponseBody. |
| Application | Used for Web applications. | Used for RESTful APIs. |
| Request handling and Mapping | Used with @RequestMapping annotation to map HTTP requests with methods. | Used to handle requests like GET, PUT, POST, and DELETE. |

* **@RestController** annotation encapsulates **@Controller** and **@ResponseBody** annotation.

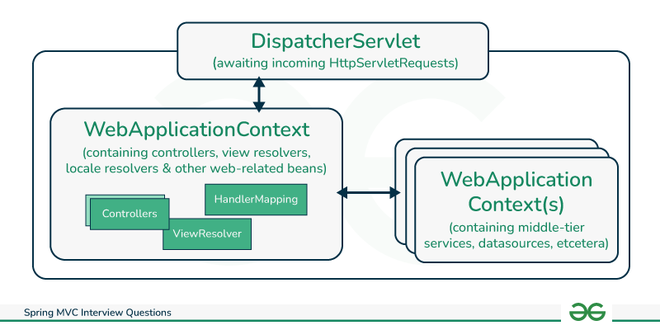


*@RestController = @Controller + @ResponseBody*

**11. What is WebApplicationContext in Spring MVC?**

**WebApplicationContext** is an extension of **ApplicationContext.** It has servlet context information. We can use multiple WebApplicationContext in a single web application which means every dispatcher servlet is associated with a single WebApplicationContext.

A web application can have more than one dispatcher servlet to handle HTTP requests and every front controller has a separate WebApplicationContext configuration file. It is configured using **\*-servlet.xml** file.



**12. What is DTO and Repository Interface in Spring MVC?**

**DTO**: DTO stands for [**Data Transfer Object**](https://www.geeksforgeeks.org/data-transfer-object-dto-in-spring-mvc-with-example). It is a simple model class that encapsulates other different objects into one. Sending the data between client and server requires a model class. When the client requests data from the server, instead of sending multiple responses it will send only one.

*Note: DTO should not contain any additional logic, except the logic for encapsulation.*

data class GfgAuthor (  
 val name: String,  
 val age: Int  
 )

**Repository Interface**: To establish database connectivity and data access, we define logic and methods inside the repository class, and we define this interface by putting **@Repository** annotation.

*Note: Repository implements any one of the pre-defined repositories like CRUD repository or JPA repository.*

**13. How to handle different types of incoming HTTP request methods in Spring MVC?**

To handle different types of HTTP request methods we use **@RequestMapping** annotation in Spring MVC. For mapping incoming HTTP requests with the handler method at the method level or class level, @RequestMapping annotation is being used.

@RequestMapping(value = "", method=RequestMapping.GET)

There are different types of methods for HTTP requests.

* **GET**
* **POST**
* **PUT**
* **DELETE**

For each request,run we can use separate annotations like **@GetMapping, @PostMapping, @PutMapping**, and **@DeleteMapping** instead of passing the method inside the @RequestMapping annotation.

Ex- @GetMapping("/hello")

**14. Difference between *ApplicationContext* and *WebApplicationContext* in Spring MVC**

| **Features** | **ApplicationContext** | **WebApplicationContext** |
| --- | --- | --- |
| **Working** | It is designed for stand-alone applications. | It is designed for web applications that run within a container (web container) like Tomcat or  Jetty. |
| **Configuration** | It is configured using applicationContext.xml or @Configuration and @Bean annotation. | It configured using XML file \*-servlet.xml |
| **Example** | Desktop Applications | RESTful APIs |

**15. How to perform Validation in Spring MVC?**

There are so many different ways to perform validation in Spring MVC.

* **First Approach**: **Annotation based Validation** - Using **@Valid, @NotNull, and @Email** annotations which are based on **JSR-303 Bean Validation** to define validation rules on a model attribute.

public class GfgAuthor {  
 @NotNull  
 private String name;  
 @Email  
 private String emailId;  
 }

* **Second Approach**: By implementing **org.springframework.validation.Validator** interface or we can say this custom validator interface.
* **Third Approach**: Validate manually is the common approach to perform specific validation.

**16. How to perform Exception Handling in Spring MVC?**

Exception means an unexpected error that occurs during application execution. To identify or handle that error, called Exception Handling.

In Spring MVC, we can handle exceptions using many mechanisms, three of which are described below:

* **@ExceptionHandler annotation**: It allows us to define different methods to handle different exceptions in the application.

@ExceptionHandler(ResourceNotFound.class)

* **HandlerExceptionResolver Interface**: To handle exceptions, this interface allows to implement custom logics and allow to create own exception resolver that can handle different types of exceptions thrown by the application.

public class GfgExceptionHandler implements HandlerExceptionResolver

* **Log Exception**: This exception-handling mechanism is used for debugging and analysis of an application.

**17. Difference between *@RequestParam* and *@PathVariable* annotations in Spring MVC**

| **Features** | **@RequestParam** | **@PathVariable** |
| --- | --- | --- |
| **Binding Value** | From the URL query parameter, it binds the value. | From dynamic segments in the URL path, it binds the values. |
| **Requirement** | This annotation is always Optional by default. | This annotation is always Required by default. |
| **Syntactical Example** | @RequestParam(name="author") String author | @PathVariable("author") String author |

**18. Explain Query String and Query Parameter in Spring MVC.**

In Spring MVC, Query String and Query Parameter are used to pass the data to a web application through URL.

* **Query String**: In a URL, the query string comes after "?". It contains key-value pairs that are separated by "&".

https://gfg.org?path?key=value&key1=value1

* **Query Parameter**: In a query string, the key-value pair is called the query parameter.
  + **Key**: name of data
  + **Value**: actual data

To access query parameters **@RequestParam** and **@PathVariable** annotations are used in a Spring MVC application.

**19. Define the purpose of the *@ModelAttribute* annotation.**

**@ModelAttribute** in Spring MVC has many purposes but the two main purposes are defined below:

* This annotation enables binding the method parameter value and returns the value of a method to a named attribute.
* This annotation allows us to populate the model object data and that data can be accessed by viewing pages like **Thymeleaf, Freemarker,** etc. and then we can also view the data.

**20. Difference between *@RequestBody* and *@ResponseBody* Annotation in Spring MVC**

| **Features** | **@RequestBody** | **@ResponseBody** |
| --- | --- | --- |
| **Usage** | This annotation is used to convert incoming HTTP requests from JSON format to domain objects. | This annotation is used to convert domain objects to JSON format. |
| **Binding** | Here method parameter binds the request body. | The Return type of method binds with the response body. |
| **Data Transfer** | Receives data from the end user. | Sends data to end user. |

**21. Explain the Multi Action Controller in Spring MVC.**

[**Multi Action Controller**](https://www.geeksforgeeks.org/spring-multi-action-controller-with-example) in Spring MVC is a unique controller class (MultiActionController) that is used to handle multiple HTTP request types like **GET, PUT, POST** ETC. There are many advantages of Multi Action Controller.

* It reduces code duplication, simplifies maintenance, and increases flexibility.
* It manages and implements CRUD operations.

*Note: Multi Action Controller is not a best option for complex logics.*

**Spring MVC Interview Questions For Experienced(5+years)**

**22. Explain Spring MVC Interceptor.**

Spring MVC Interceptor acts as a middleware in MVC applications. While executing the requests, we can process the requests also after, before or during the execution of the requests. We can make certain changes and can process the requests.

* During Requests - It implements error handling. And it Modifies the request context.

*Note: Servlet filters and AOP (Aspect Oriented Programming) are alternatives to Spring MVC Interceptors.*

**23. Explain the role/purpose of ContextLoaderListener in Spring MVC.**

**ContextLoaderListener** is an important module of Spring MVC. It initializes the Spring ApplicationContext. Some functionalities of ContextLoaderListener is:

* Create ApplicationContext and load the necessary beans.
* Bootstraps the Application.
* Initializes the application even before any web requests are processed.

**24. How to enable CSRF protection in a Spring MVC Application?**

[CSRF](https://www.geeksforgeeks.org/what-is-cross-site-request-forgery-csrf) stands for **Cross Site Request Forgery**. It is a security vulnerability. In this, a hacker can hack end user's browser and send requests to web applications which may cause sensitive data leaks and any unauthorized actions. CSRF protection is enabled by Spring Security for the web applications that are starting from the 4.0 version.

**Steps to follow for enabling CSRF in Web Applications:**

* **Step 1**: The application should use proper HTTP verbs.
* **Step 2**: Verify CSRF protection is enabled in configuration (Enabled after Spring Security 4.0 version)
* **Step 3**: Include the CSRF tokens (Automatically generated by Spring Security).

*Note: To disable CSRF for any specific URL, we can use @CSRFIgnore annotation.*

**25. How to use JSTL with Spring MVC?**

JSTL stands for **JavaServer Pages Standard Tag Library**. It provides tags for working with web pages and its data. We can use JSTL with Spring MVC to simplify the development process.

**Steps to Implementation:**

* **Step 1**: JSTL Library dependencies need to be added.
* **Step 2**: Configure JSTL in Spring MVC by adding **JstlViewResolver** to the configuration file.
* **Step 3**: Use JSP (Java Server Pages) tags.
* **Step 4**: Spring MVC data access in JSTL.

JSTL tags can be combined with Spring Security tags to enhance the development process.

**26. How to integrate the Database with the Spring MVC Project?**

Database Integration is a very vital process in every project. To integrate a database with Spring MVC, follow the below steps:

* **Step 1**: Select the database and load driver.
* **Step 2**: Configure JDBC database connectivity/Configure Spring Data JPA
* **Step 3**: Create Beans (entity object)
* **Step 4**: DataSource Configuration
* **Step 5**: DAO Layer Implementation
* **Step 6**: Controller and Services of Spring MVC need to be used.

**27. How to use *SessionAttributes* in Spring MVC?**

**SessionAttributes** in Spring MVC is used to store model attributes in HTTP sessions and can retrieve them. For this, we use **@SessionAttribute** annotation. It avoids re-creating objects in every request. It can share data between multiple requests.

**Steps to use SessionAttributes in Spring MVC:**

* **Step 1**: Use @SessionAttribute to Controller class or method.
* **Step 2**: Add model attribute to session.
* **Step 3**: In other controller, access the session attributes.

**1. What is Additional configuration file in Spring MVC?**

In Spring MVC, the additional configuration file contains custom configuration properties.

**2. Can we declare a class as a Controller? If yes, then explain how.**

Yes, we can declare a class as a Controller. To make a class as a controller we need to mark the class with @Controller annotation.

**3. State the annotations that are used to handle different HTTP requests.**

To handle different HTTP requests, annotations that are used:

* GET - @GetMapping
* POST - @PostMapping
* PUT - @PutMapping
* DELETE - @DeleteMapping
* PATCH - @PatchMapping

**4. What is ModelInterface?**

In Spring MVC, ModelInterface holds the data, and it transfers the data between View and Controller.

**5. What is ModelMap?**

In Spring MVC, ModelMap is the implementation of Model. It is used to transfer data to Views.

**6. What is ModelAndView?**

Spring MVC, ModelAndView concatenate the Model (data) and the View Name in one object form.

**7. Explain different ways to read data from the FORM in Spring MVC.**

There are different ways to read data from the form in Spring MVC. Two of them are:

* **@RequestParam**: It binds individual form directly to method argument.
* **@ModelAttribute**: It binds the entire form to a POJO (Plain Old Java Object) class.

**8. What is Form tag library in short?**

In Spring MVC, the form tag library is used to build forms and also it integrates data binding in several ways.

**9. What do you mean by Bean Validation in Spring MVC?**

Bean Validation in Spring MVC performs automatic validation in Spring applications, and we can define constraints on model objects.

**10. State the two annotations that are used to validate the user's input within a number range in MVC.**

The two annotations that are used to validate the user's input within a number range in Spring MVC are:

* **@Min:** With this annotation the Integer value is required to pass, and it specifies the minimum value allowed.
* **@Max**: With this annotation the Integer value is required to pass, and it specifies the maximum value allowed.

**Advantages of Spring MVC**

Spring MVC is beneficial in many aspects, it provides value to developers as well as applications. Some advantages of Spring MVC are:

1. It helps keep codes clean and organized and simplifies development & maintenance.
2. It supports most modern APIs.
3. It has a large community that can provide an immense amount of knowledge and help with learning and solving problems.
4. It utilizes loose coupling and lightweight servlets to maximize resource efficiency.
5. It has smooth testing and debugging due to its layered architecture and testable components.
6. It can adapt to different needs and uses light servlets for faster performance.

**Future Trends and Updates in Spring MVC**

Spring MVC is considered the best framework in the JAVA ecosystem because it follows future requirements and slowly adds new features to the framework. Some future updates anticipated in Spring MVC are:

1. **Enhanced Reactive Programming**: Spring MVC may boost its support for asynchronous and non-blocking operations, making web applications faster and more efficient.
2. **Microservices Integration**: There could be easier ways to combine Spring MVC with Spring Boot and Spring Cloud, simplifying the creation of microservices.
3. **Better Cloud Integration**: Expect more features for smoothly connecting Spring MVC applications with various cloud services.
4. **Advanced API Features**: Look for new tools in Spring MVC for easier building, documenting, and managing REST APIs.
5. **AI and ML Integration**: Spring MVC might introduce straightforward methods to incorporate artificial intelligence and machine learning, adding intelligence to web applications.
6. **Performance Improvements**: Continuous efforts are likely to make Spring MVC faster, more memory-efficient, and quicker to start up.