### What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

What are the types of IOC container in spring?

There are two types of IOC containers in spring framework.

1. BeanFactory
2. ApplicationContext

### What is the difference between BeanFactory and ApplicationContext?

BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n etc.

### What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

|  |  |  |
| --- | --- | --- |
| **No.** | **Scope** | **Description** |
| 1) | singleton | The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope. |
| 2) | prototype | The bean instance will be created each time when requested. |
| 3) | request | The bean instance will be created per HTTP request. |
| 4) | session | The bean instance will be created per HTTP session. |
| 5) | globalsession | The bean instance will be created per HTTP global session. It can be used in portlet context only. |

What is an InternalResourceViewResolver in Spring MVC?

The **InternalResourceViewResolver** is a class which is used to resolve internal view in Spring MVC. Here, you can define the properties like prefix and suffix where prefix contains the location of view page and suffix contains the extension of view page. For example:-

1. **<bean** id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
2. **<property** name="prefix" value="/WEB-INF/jsp/"**></property>**
3. **<property** name="suffix" value=".jsp"**></property>**
4. **</bean>**

How to map controller class and its methods with URL?

The **@RequestMapping** annotation is used to map the controller class and its methods. You can specify this annotation on the class name as well as method name with a particular URL that represents the path of the requested page. For example:-

1. @Controller
2. @RequestMapping("/ form")
3. class Demo
4. {
5. @RequestMapping("/show")
6. public String display()
7. {
8. }
10. }

Name the annotations used to handle different types of incoming HTTP request methods?

The following annotations are used to handle different types of incoming HTTP request methods: -

* @GetMapping
* @PostMapping
* @PutMapping
* @PatchMapping
* @DeleteMapping

What is the purpose of @PathVariable annotation in Spring MVC?

The @PathVariable annotation is used to extract the value of the URI template. It is passed within the parameters of the handler method.

For example: -

1. @RequestMapping("/show/{id}")
2. public String handler(@PathVariable("id") String s, Model map)
3. {
4. }

## ResponseEntity

ResponseEntity **represents the whole HTTP response: status code, headers, and body**. As a result, we can use it to fully configure the HTTP response.

If we want to use it, we have to return it from the endpoint; Spring takes care of the rest.

ResponseEntity is a generic type. Consequently, we can use any type as the response body:

## **Introduction**

While working on the REST API, we may need to bind the HTTP request and response body with the domain object. **Spring @RequestBody and @ResponseBody annotations** used to bind these HTTP request/response. Let’s have a closer look at these two Spring annotations.

## **1. Spring RequestBody Annotation**

To put it in layman terms, the @RequestBody annotation binds the HTTPRequest body to the domain object. Spring framework automatically de-serializes incoming HTTPRequest to the Java object using [Http Message Converters](https://www.javadevjournal.com/spring/spring-http-message-converter/).We pass the body of the request through a HttpMessageConverter to resolve the method argument depending on the content type of the request.If above definitions seems complicated, ***the @RequestBody annotation allows us to retrieve the request’s body and automatically*** convert it to Java Object

## **2. Spring ResponseBody Annotation**

To put this in simple words, @ResponseBody tell Spring framework to serialize a return object into JSON or XML and send this information back as part of the HTTPResponse.

With Spring 4.x, If we are using working on the ***REST API***, we should not use @ResponseBody on method level, but <@RestController on a class level.@RestController is a [composed annotation](https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#beans-meta-annotations) that is itself a meta-annotated with @Controller and @ResponseBody.Let’s look at the previous example, our register method is returning Customer object and we like this to return as  JSON in the HTTPResponse.Here is our Customer model.

### What is the role of the Model interface in Spring MVC?

The **Model** interface works as a container that contains the data of the application. Here, data can be in any form such as objects, strings, information from the database, etc.

### 14) What do you mean by ModelAndView in Spring MVC?

The **ModelAndView** is a class that holds both Model and View where the model represents the data, and view represents the representation of that data. This class returns the model and view in the single return value.

### 15) What is ModelMap in Spring MVC?

The **ModelMap** is a class that provides the implementation of Map. It extends the LinkedHashMap class. It facilitates to pass a collection of values as if they were within a Map.

16) What are the ways of reading data from the form in Spring MVC?

The following ways to read the data from the form are: -

* **HttpServletRequest interface** - The **HttpServletRequest** is a java interface present in javax.servlet.http package. Like Servlets, you can use HttpServletRequest in Spring to read the HTML form data provided by the user.
* **@RequestParam annotation** - The **@RequestParam** annotation reads the form data and binds it automatically to the parameter present in the provided method.
* **@ModelAttribute annotation** - The **@ModelAttribute** annotation binds a method parameter or its return value to a named model attribute.

### 19) What is the difference between @ModelAttribute and @RequestBody?

### 19) What is Bean Validation API?

The **Bean Validation API** is a Java specification which is used to apply constraints on object model via annotations. Here, we can validate a length, number, regular expression, etc. Apart from that, we can also provide custom validations.

As Bean Validation API is just a specification, it requires an implementation. So, for that, it uses Hibernate Validator. The Hibernate Validator is a fully compliant JSR-303/309 implementation that allows to express and validate application constraints.

### 20) What is the use of @Valid annotation in Spring MVC?

The **@Valid** annotation is used to apply validation rules on the provided object.

21) What is the purpose of BindingResult in Spring MVC validations?

The **BindingResult** is an interface that contains the information of validations. For example :-

1. @RequestMapping("/helloagain")
2. public String submitForm( @Valid @ModelAttribute("emp") Employee e, BindingResult br)
3. {
4. if(br.hasErrors())
5. {
6. return "viewpage";
7. }
8. else
9. {
10. return "final";
11. }
12. }

22) How to validate user's input within a number range in Spring MVC?

In Spring MVC Validation, we can validate the user's input within a number range by using the following annotations: -

* **@Min annotation** - It is required to pass an integer value with @Min annotation. The user input must be equal to or greater than this value.
* **@Max annotation** - It is required to pass an integer value with @Max annotation. The user input must be equal to or smaller than this value.

### 23) How to validate the user input in a particular sequence in Spring MVC?

The Spring MVC Validation allows us to validate the user input in a particular sequence by using @Pattern annotation. Here, we can provide the required regular expression to **regexp** attribute and pass it with the annotation.

25) What do you understand by Spring MVC Tiles?

The Spring provides integration support with apache tiles framework. So we can manage the layout of the Spring MVC application with the help of spring tiles support. The following are the advantages of Tiles support in Spring MVC: -

* **Reusability:** We can reuse a single component in multiple pages like header and footer components.
* **Centralized control:** We can control the layout of the page by a single template page only.
* **Easy to change the layout:** By the help of a single template page, we can change the layout of the page anytime. So your website can easily adopt new technologies such as bootstrap and jQuery.

**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 thedependencies 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>

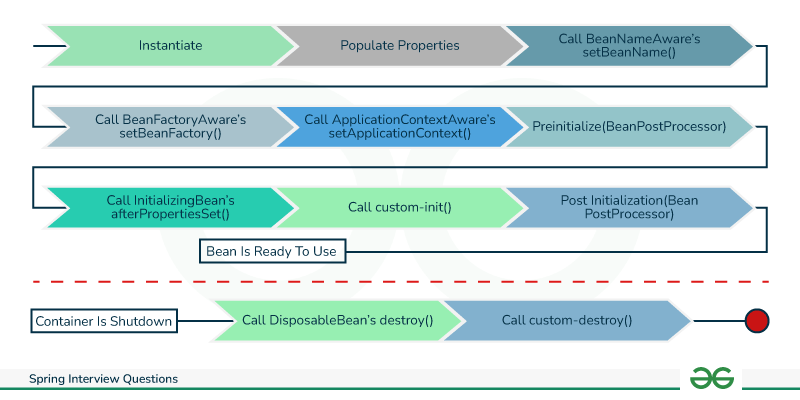
* **Java Configuration:** Uses Java class to define bean configurations and dependencies. Best alternative for XML-based configurations
  + Mainly *@Configuration*and *@Bean* annotations are used for configuration.
* **Property Sources:** Store configuration settings in external sources like environment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



**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.

Q: how @autowired internally works?

Q: difference between applicationcontext and abstractapplicationcontext

Q: have you used @values in java.

Q: what is accelator and spring boot.