

## Spring Interview Questions and Answers

72 Comments

Filed Under: [Interview Questions](#)

[Home](#) » [Interview Questions](#) » Spring Interview Questions and Answers



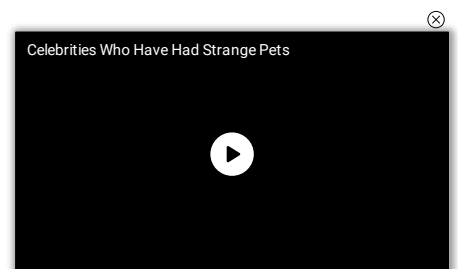
I have posted a lot of [Spring Tutorials](#) recently. This post will help you get through **Spring Interview Questions** explaining the core concepts in detail.

**Spring Framework** is one of the most popular Java EE frameworks for web applications. **Dependency Injection** and **Aspect Oriented Programming** are at the heart of Spring framework. If you are good in Spring Framework, the chances of getting selected get high.

**Pro Tip:** [Core Java](#) is the base of any Java based framework, so if you are preparing for interviews then please go through [Core Java Interview Questions](#) and [Java Interview Questions](#) articles.

### Spring Interview Questions

1. [What is Spring Framework?](#)
2. [What are some of the important features and advantages of Spring Framework?](#)
3. [What do you understand by Dependency Injection?](#)
4. [How do we implement DI in Spring Framework?](#)
5. [What are the new features in Spring 5?](#)
6. [What is Spring WebFlux?](#)
7. [What are the benefits of using Spring Tool Suite?](#)
8. [Name some of the important Spring Modules?](#)
9. [What do you understand by Aspect Oriented Programming?](#)
10. [What is Aspect, Advice, Pointcut, JointPoint and Advice Arguments in AOP?](#)
11. [What is the difference between Spring AOP and AspectJ AOP?](#)



13. What is a Spring Bean?
14. What is the importance of Spring bean configuration file?
15. What are different ways to configure a class as Spring Bean?
16. What are different scopes of Spring Bean?
17. What is Spring Bean life cycle?
18. How to get ServletContext and ServletConfig object in a Spring Bean?
19. What is Bean wiring and @Autowired annotation?
20. What are different types of Spring Bean autowiring?
21. Does Spring Bean provide thread safety?
22. What is a Controller in Spring MVC?
23. What's the difference between @Component, @Repository & @Service annotations in Spring?
24. What is DispatcherServlet and ContextLoaderListener?
25. What is ViewResolver in Spring?
26. What is a MultipartResolver and when it's used?
27. How to handle exceptions in Spring MVC Framework?
28. How to create ApplicationContext in a Java Program?
29. Can we have multiple Spring configuration files?
30. What is ContextLoaderListener?
31. What are the minimum configurations needed to create Spring MVC application?
32. How would you relate Spring MVC Framework to MVC architecture?
33. How to achieve localization in Spring MVC applications?
34. How can we use Spring to create Restful Web Service returning JSON response?
35. What are some of the important Spring annotations you have used?
36. Can we send an Object as the response of Controller handler method?
37. How to upload file in Spring MVC Application?
38. How to validate form data in Spring Web MVC Framework?
39. What is Spring MVC Interceptor and how to use it?
40. What is Spring JdbcTemplate class and how to use it?
41. How to use Tomcat JNDI DataSource in Spring Web Application?
42. How would you achieve Transaction Management in Spring?
43. What is Spring DAO?
44. How to integrate Spring and Hibernate Frameworks?
45. What is Spring Security?
46. How to inject a java.util.Properties into a Spring Bean?

Celebrities Who Have Had Strange Pets



48. What are some of the best practices for Spring Framework?

## Spring Interview Questions and Answers

1.

### What is Spring Framework?

Spring is one of the most widely used Java EE framework. Spring framework core concepts are "Dependency Injection" and "Aspect Oriented Programming".

Spring framework can be used in normal java applications also to achieve loose coupling between different components by implementing dependency injection and we can perform cross-cutting tasks such as logging and authentication using spring support for aspect-oriented programming.

I like spring because it provides a lot of features and different modules for specific tasks such as Spring MVC and Spring JDBC. Since it's an open source framework with a lot of online resources and active community members, working with the Spring framework is easy and fun at the same time.

**Recommended Read:** [Spring Framework](#)

2.

### What are some of the important features and advantages of Spring Framework?

Spring Framework is built on top of two design concepts – Dependency Injection and Aspect Oriented Programming.

Some of the features of spring framework are:

- Lightweight and very little overhead of using framework for our development.
- Dependency Injection or Inversion of Control to write components that are independent of each other, spring container takes care of wiring them together to achieve our work.
- Spring IoC container manages Spring Bean life cycle and project specific configurations such as JNDI lookup.
- Spring MVC framework can be used to create web applications as well as restful web services capable of returning XML as well as JSON response.
- Support for transaction management, JDBC operations, File uploading, Exception Handling etc with very little configurations, either by using annotations or by spring bean configuration file.

Some of the advantages of using Spring Framework are:

- Reducing direct dependencies between different components of the application, usually Spring IoC container is responsible for initializing resources or beans and inject them as dependencies.
- Writing unit test cases are easy in Spring framework because our business logic doesn't have direct dependencies with actual resource implementation classes. We can easily write a test configuration and inject our mock beans for testing purposes.
- Reduces the amount of boiler-plate code, such as initializing objects, open/close resources. I like JdbcTemplate class a lot because it helps us in removing all the boiler-plate code that comes with JDBC programming.
- Spring framework is divided into several modules, it helps us in keeping our application lightweight. For example, if we don't need Spring transaction management features, we don't need to add that dependency on our project.
- Spring framework support most of the Java EE features and even much more. It's always on top of the new technologies, for example, there is a Spring project for Android to help us write better code for native [Android](#) applications. This makes spring framework a complete package and we don't need to look after the different framework for different requirements.

13.

### What do you understand by Dependency Injection?

Dependency Injection design pattern allows us to remove the hard-coded dependencies and make our application loosely coupled. We can implement dependency injection pattern to move the dependency resolution from compile-time to runtime.

Some of the benefits of using Dependency Injection are Separation of Concerns, Boilerplate Code reduction, Configurable components.

Read more at [Dependency Injection Tutorial](#). We can also use [Google Guice for Dependency Injection](#) to automate the process. We are looking for more than just dependency injection and that's why Spring is the top choice for this.

Celebrities Who Have Had Strange Pets



14.

## How do we implement DI in Spring Framework?

We can use Spring XML based as well as Annotation-based configuration to implement DI in spring applications. For better understanding, please read [Spring Dependency Injection](#) example where you can learn both the ways with JUnit test case. The post also contains a sample project zip file, that you can download and play around to learn more.

15.

## What are the new features in Spring 5?

Spring 5 brought a massive update to Spring framework. Some of the new features in Spring 5 are:

- 16. Spring 5 runs on Java 8+ and supports Java EE 7. So we can use lambda expressions and Servlet 4.0 features. It's good to see Spring trying to support the latest versions.
- 17. Spring Framework 5.0 comes with its own Commons Logging bridge; spring-jcl instead of standard Commons Logging.
- 18. Support for providing spring components information through index file "META-INF/spring.components" rather than classpath scanning.
- 19. Spring WebFlux brings reactive programming to the Spring Framework.
- 20. Spring 5 also supports Kotlin programming now. This is a huge step towards supporting functional programming, just as Java is also moving towards functional programming.
- 21. Support for JUnit 5 and parallel testing execution in the Spring TestContext Framework.

You can read about these features in more detail at [Spring 5 Features](#).

22.

## What is Spring WebFlux?

Spring WebFlux is the new module introduced in Spring 5. Spring WebFlux is the first step towards the reactive programming model in spring framework.

Spring WebFlux is the alternative to the Spring MVC module. Spring WebFlux is used to create a fully asynchronous and non-blocking application built on the event-loop execution model.

You can read more about it at [Spring WebFlux Tutorial](#).

23.

## What are the benefits of using Spring Tool Suite?

We can install plugins into Eclipse to get all the features of Spring Tool Suite. However, STS comes with Eclipse with some other important kinds of stuff such as Maven support, Templates for creating different types of Spring projects and tc server for better performance with Spring application.

I like STS because it highlights the Spring components and if you are using AOP pointcuts and advice, then it clearly shows v pointcut. So rather than installing everything on our own, I prefer using STS when developing Spring-based applications.

Celebrities Who Have Had Strange Pets



## name some of the important Spring Modules?

Some of the important Spring Framework modules are:

- **Spring Context** – for dependency injection.
- **Spring AOP** – for aspect oriented programming.
- **Spring DAO** – for database operations using DAO pattern
- **Spring JDBC** – for JDBC and DataSource support.
- **Spring ORM** – for ORM tools support such as Hibernate
- **Spring Web Module** – for creating web applications.
- **Spring MVC** – Model-View-Controller implementation for creating web applications, web services etc.
- 

32.

## What do you understand by Aspect Oriented Programming?

Enterprise applications have some common cross-cutting concerns that are applicable to different types of Objects and application modules, such as logging, transaction management, data validation, authentication etc. In Object Oriented Programming, modularity of application is achieved by Classes whereas in AOP application modularity is achieved by Aspects and they are configured to cut across different classes methods.

AOP takes out the direct dependency of cross-cutting tasks from classes that are not possible in normal object-oriented programming. For example, we can have a separate class for logging but again the classes will have to call these methods for logging the data. Read more about Spring AOP support at [Spring AOP Example](#).

33.

## What is Aspect, Advice, Pointcut, JointPoint and Advice Arguments in AOP?

**Aspect:** Aspect is a class that implements cross-cutting concerns, such as transaction management. Aspects can be a normal class configured and then configured in Spring Bean configuration file or we can use Spring AspectJ support to declare a class as Aspect using `@Aspect` annotation.

**Advice:** Advice is the action taken for a particular join point. In terms of programming, they are methods that gets executed when a specific join point with matching pointcut is reached in the application. You can think of Advices as [Spring interceptors](#) or [Servlet Filters](#).

**Pointcut:** Pointcut are regular expressions that are matched with join points to determine whether advice needs to be executed or not. Pointcut uses different kinds of expressions that are matched with the join points. Spring framework uses the AspectJ pointcut expression language for determining the join points where advice methods will be applied.

**Join Point:** A join point is a specific point in the application such as method execution, exception handling, changing object variable values etc. In Spring AOP a join point is always the execution of a method.

**Advice Arguments:** We can pass arguments in the advice methods. We can use `args()` expression in the pointcut to be applied to any method that matches the argument pattern. If we use this, then we need to use the same name in the advice method from where the argument type is determined.

These concepts seems confusing at first, but if you go through [Spring Aspect, Advice Example](#) then you can easily relate to them.

34.

## What is the difference between Spring AOP and AspectJ AOP?

AspectJ is the industry-standard implementation for Aspect Oriented Programming whereas Spring implements AOP for some cases. Main differences between Spring AOP and AspectJ are:

- Spring AOP is simpler to use than AspectJ because we don't need to worry about the weaving process.
- Spring AOP supports AspectJ annotations, so if you are familiar with AspectJ then working with Spring AOP is easier.
- Spring AOP supports only proxy-based AOP, so it can be applied only to method execution join points. AspectJ support a
- One of the shortcomings of Spring AOP is that it can be applied only to the beans created through Spring Context.

Celebrities Who Have Had Strange Pets



39.

## What is Spring IoC Container?

**Inversion of Control** (IoC) is the mechanism to achieve loose-coupling between Objects dependencies. To achieve loose coupling and dynamic binding of the objects at runtime, the objects define their dependencies that are being injected by other assembler objects. Spring IoC container is the program that injects dependencies into an object and makes it ready for our use.

Spring Framework IoC container classes are part of `org.springframework.beans` and `org.springframework.context` packages and provides us different ways to decouple the object dependencies.

Some of the useful ApplicationContext implementations that we use are;

- `AnnotationConfigApplicationContext` : For standalone java applications using annotations based configuration.

- `ClassPathXmlApplicationContext` : For standalone java applications using XML based configuration.

- `FileSystemXmlApplicationContext` : Similar to `ClassPathXmlApplicationContext` except that the xml configuration file can be loaded from anywhere in the file system.

- `AnnotationConfigWebApplicationContext` and `XmlWebApplicationContext` for web applications.

44.

## What is a Spring Bean?

Any normal java class that is initialized by Spring IoC container is called Spring Bean. We use Spring `ApplicationContext` to get the Spring Bean instance.

Spring IoC container manages the life cycle of Spring Bean, bean scopes and injecting any required dependencies in the bean.

45.

## What is the importance of Spring bean configuration file?

We use Spring Bean configuration file to define all the beans that will be initialized by Spring Context. When we create the instance of Spring `ApplicationContext`, it reads the spring bean XML file and initializes all of them. Once the context is initialized, we can use it to get different bean instances.

Apart from Spring Bean configuration, this file also contains spring MVC interceptors, view resolvers and other elements to support annotations based configurations.

46.

## What are different ways to configure a class as Spring Bean?

There are three different ways to configure Spring Bean.

**47. XML Configuration:** This is the most popular configuration and we can use bean element in context file to configure a Spring Bean. For example:

```
<bean name="myBean" class="com.journaldev.spring.beans.MyBean"></bean>
```

**48. Java Based Configuration:** If you are using only annotations, you can configure a Spring bean using `@Bean` annotation. This annotation is used with `@Configuration` classes to configure a spring bean. Sample configuration is:

```
@Configuration
@ComponentScan(value="com.journaldev.spring.main")
public class MyConfiguration {

    @Bean
    public MyService getService(){
        return new MyService();
    }
}
```

To get this bean from spring context, we need to use following code snippet:

Celebrities Who Have Had Strange Pets



```
MyConfiguration.class);
MyService service = ctx.getBean(MyService.class);
```

49. **Annotation Based Configuration:** We can also use `@Component`, `@Service`, `@Repository` and `@Controller` annotations with classes to configure them to be as spring bean. For these, we would need to provide base package location to scan for these classes. For example:

```
<context:component-scan base-package="com.journaldev.spring" />
```

50.

## What are different scopes of Spring Bean?

There are five scopes defined for Spring Beans.

- 51. **singleton:** Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn't have shared instance variables otherwise it might lead to data inconsistency issues because it's not thread-safe.
- 52. **prototype:** A new instance will be created every time the bean is requested.
- 53. **request:** This is same as prototype scope, however it's meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
- 54. **session:** A new bean will be created for each HTTP session by the container.
- 55. **global-session:** This is used to create global session beans for Portlet applications.

Spring Framework is extendable and we can create our own scopes too, however most of the times we are good with the scopes provided by the framework.

To set spring bean scopes we can use "scope" attribute in bean element or `@Scope` annotation for annotation based configurations.

56.

## What is Spring Bean life cycle?

Spring Beans are initialized by Spring Container and all the dependencies are also injected. When the context is destroyed, it also destroys all the initialized beans. This works well in most of the cases but sometimes we want to initialize other resources or do some validation before making our beans ready to use. Spring framework provides support for post-initialization and pre-destroy methods in spring beans.

We can do this by two ways – by implementing `InitializingBean` and `DisposableBean` interfaces or using **init-method** and **destroy-method** attribute in spring bean configurations. For more details, please read [Spring Bean Life Cycle Methods](#).

57.

## How to get ServletContext and ServletConfig object in a Spring Bean?

There are two ways to get Container specific objects in the spring bean.

- 58. Implementing Spring `*Aware` interfaces, for these `ServletContextAware` and `ServletConfigAware` interfaces, for complete example of these aware interfaces, please read [Spring Aware Interfaces](#)
- 59. Using `@Autowired` annotation with bean variable of type `ServletContext` and `ServletConfig`. They will work only in servlet container specific environment only though.

```
@Autowired
ServletContext servletContext;
```

Celebrities Who Have Had Strange Pets



60.

## What is Bean wiring and @Autowired annotation?

Usually, it's best practice to do the explicit wiring of all the bean dependencies, but the Spring framework also supports autowiring. We can use `@Autowired` annotation with fields or methods for **autowiring byType**. For this annotation to work, we also need to enable annotation-based configuration in spring bean configuration file. This can be done by `context:annotation-config` element.

For more details about `@Autowired` annotation, please read [Spring Autowire Example](#).

61.

## What are different types of Spring Bean autowiring?

There are four types of autowiring in Spring framework.

62. **autowire byName**

63. **autowire byType**

64. **autowire by constructor**

65. autowiring by `@Autowired` and `@Qualifier` annotations

Prior to Spring 3.1, **autowire by autodetect** was also supported that was similar to autowire by constructor or byType. For more details about these options, please read [Spring Bean Autowiring](#).

66.

## Does Spring Bean provide thread safety?

The default scope of Spring bean is singleton, so there will be only one instance per context. That means that all the having a class level variable that any thread can update will lead to inconsistent data. Hence in default mode spring beans are not thread-safe.

However, we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It's a design decision and based on the project requirements.

67.

## What is a Controller in Spring MVC?

Just like MVC design pattern, Controller is the class that takes care of all the client requests and send them to the configured resources to handle it. In Spring MVC, `org.springframework.web.servlet.DispatcherServlet` is the front controller class that initializes the context based on the spring beans configurations.

A Controller class is responsible to handle a different kind of client requests based on the request mappings. We can create a controller class by using `@Controller` annotation. Usually, it's used with `@RequestMapping` annotation to define handler methods for specific URI mapping.

68.

## What's the difference between `@Component`, `@Controller`, `@Repository` & `@Service` annotations in Spring?

`@Component` is used to indicate that a class is a component. These classes are used for auto-detection and configured as bean when annotation based configurations are used.

`@Controller` is a specific type of component, used in MVC applications and mostly used with RequestMapping annotation.

`@Repository` annotation is used to indicate that a component is used as repository and a mechanism to store/retrieve/search data. We can apply this annotation with DAO pattern implementation classes.

`@Service` is used to indicate that a class is a Service. Usually, the business facade classes that provide some services are annotated with `@Service`.

We can use any of the above annotations for a class for auto-detection but different types are provided so that you can easily identify the role of the class.

Celebrities Who Have Had Strange Pets





69.

## What is DispatcherServlet and ContextLoaderListener?

**DispatcherServlet** is the front controller in the Spring MVC application and it loads the spring bean configuration file and initialize all the beans that are configured. If annotations are enabled, it also scans the packages and configure any bean annotated with **@Component**, **@Controller**, **@Repository** or **@Service** annotations.

**ContextLoaderListener** is the listener to start up and shut down Spring's root **WebApplicationContext**. Its important functions are to tie up the lifecycle of **ApplicationContext** to the lifecycle of the **ServletContext** and to automate the creation of **ApplicationContext**. We can use it to define shared beans that can be used across different spring contexts.

70.

## What is ViewResolver in Spring?

**ViewResolver** implementations are used to resolve the view pages by name. Usually we configure it in the spring bean configuration file. For example:

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <beans:property name="prefix" value="/WEB-INF/views/" />
  <beans:property name="suffix" value=".jsp" />
</beans:bean>
```

**InternalResourceViewResolver** is one of the implementation of **ViewResolver** interface and we are providing the view pages directory and suffix location through the bean properties. So if a controller handler method returns "home", view resolver will use view page located at `/WEB-INF/views/home.jsp`.

71.

## What is a MultipartResolver and when it's used?

**MultipartResolver** interface is used for uploading files – **CommonsMultipartResolver** and **StandardServletMultipartResolver** are two implementations provided by spring framework for file uploading. By default there are no multipart resolvers configured but to use them for uploading files, all we need to define a bean named "multipartResolver" with type as MultipartResolver in spring bean configurations.

Once configured, any multipart request will be resolved by the configured MultipartResolver and pass on a wrapped HttpServletRequest. Then it's used in the controller class to get the file and process it. For a complete example, please read [Spring MVC File Upload Example](#).

72.

## How to handle exceptions in Spring MVC Framework?

Spring MVC Framework provides the following ways to help us achieving robust exception handling.

**73. Controller Based** – We can define exception handler methods in our controller classes. All we need is to annotate these methods with **@ExceptionHandler** annotation.

**74. Global Exception Handler** – Exception Handling is a cross-cutting concern and Spring provides **@ControllerAdvice** annotation that we can use with any class to define our global exception handler.

**75. HandlerExceptionResolver implementation** – For generic exceptions, most of the times we serve static pages. Spring Framework provides **HandlerExceptionResolver** interface that we can implement to create global exception handler. The reason behind this handler is that Spring framework also provides default implementation classes that we can define in our spring bean configurations to achieve robust exception handling benefits.

For a complete example, please read [Spring Exception Handling Example](#).

Celebrities Who Have Had Strange Pets



## How to create ApplicationContext in a Java Program?

There are following ways to create spring context in a standalone java program.

77. **AnnotationConfigApplicationContext:** If we are using Spring in standalone java applications and using annotations for Configuration, then we can use this to initialize the container and get the bean objects.
78. **ClassPathXmlApplicationContext:** If we have spring bean configuration xml file in standalone application, then we can use this class to load the file and get the container object.
79. **FileSystemXmlApplicationContext:** This is similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from anywhere in the file system.

80.

### Can we have multiple Spring configuration files?

For Spring MVC applications, we can define multiple spring context configuration files through `contextConfigLocation`. This location string can consist of multiple locations separated by any number of commas and spaces. For example;

```
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-
jdbc.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

We can also define multiple root level spring configurations and load it through context-param. For example;

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml /WEB-INF/spring/root-security.xml</param-value>
</context-param>
```

Another option is to use import element in the context configuration file to import other configurations, for example:

```
<beans:import resource="spring-jdbc.xml"/>
```

81.

### What is ContextLoaderListener?

ContextLoaderListener is the listener class used to load root context and define spring bean configurations that will be visible to all other contexts. It's configured in web.xml file as:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

82.

### What are the minimum configurations needed to create Spring MVC app?

Celebrities Who Have Had Strange Pets



Add `spring-context` and `spring-webmvc` dependencies in the project.

- Configure `DispatcherServlet` in the web.xml file to handle requests through spring container.
- Spring bean configuration file to define beans, if using annotations then it has to be configured here. Also we need to configure view resolver for view pages.
- Controller class with request mappings defined to handle the client requests.
- 

Above steps should be enough to create a simple Spring MVC Hello World application.

87.

## How would you relate Spring MVC Framework to MVC architecture?

As the name suggests Spring MVC is built on top of **Model-View-Controller** architecture. `DispatcherServlet` is the Front Controller in the Spring MVC application that takes care of all the incoming requests and delegate it to different controller handler methods.

The model can be any Java Bean in the Spring Framework, just like any other MVC framework Spring provides automatic binding of form data to java beans. We can set model beans as attributes to be used in the view pages.

View Pages can be JSP, static HTMLs etc. and view resolvers are responsible for finding the correct view page. Once the view page is identified, control is given back to the `DispatcherServlet` controller. `DispatcherServlet` is responsible for rendering the view and returning the final response to the client.

88.

## How to achieve localization in Spring MVC applications?

Spring provides excellent support for localization or i18n through resource bundles. Basis steps needed to make our application localized are:

89. Creating message resource bundles for different locales, such as `messages_en.properties`, `messages_fr.properties` etc.
90. Defining `messageSource` bean in the spring bean configuration file of type `ResourceBundleMessageSource` or `ReloadableResourceBundleMessageSource`.
91. For change of locale support, define `localeResolver` bean of type `CookieLocaleResolver` and configure `LocaleChangeInterceptor` interceptor. Example configuration can be like below:

```
<beans:bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
<beans:property name="basename" value="classpath:messages" />
<beans:property name="defaultEncoding" value="UTF-8" />
</beans:bean>

<beans:bean id="localeResolver"
class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
<beans:property name="defaultLocale" value="en" />
<beans:property name="cookieName" value="myAppLocaleCookie"></beans:property>
<beans:property name="cookieMaxAge" value="3600"></beans:property>
</beans:bean>

<interceptors>
<beans:bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
<beans:property name="paramName" value="locale" />
</beans:bean>
</interceptors>
```

92. Use `spring:message` element in the view pages with key names, `DispatcherServlet` picks the corresponding value and renders the page in corresponding locale and return as response.

For a complete example, please read [Spring Localization Example](#).

93.

## How can we use Spring to create Restful Web Service returning JSON re

We can use Spring Framework to create Restful web services that returns JSON data. Spring provides integration with [Jackson](#) to return JSON response in restful web service.

Celebrities Who Have Had Strange Pets



94. Adding Jackson JSON dependencies, if you are using Maven it can be done with following code:

```
<!-- Jackson -->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>${jackson.databind-version}</version>
</dependency>
```

95. Configure `RequestMappingHandlerAdapter` bean in the spring bean configuration file and set the messageConverters property to MappingJackson2HttpMessageConverter bean. Sample configuration will be:

```
<!-- Configure to plugin JSON as request and response in method handler -->
<beans:bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
  <beans:property name="messageConverters">
    <beans:list>
      <beans:ref bean="jsonMessageConverter"/>
    </beans:list>
  </beans:property>
</beans:bean>

<!-- Configure bean to convert JSON to POJO and vice versa -->
<beans:bean id="jsonMessageConverter"
class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">
</beans:bean>
```

96. In the controller handler methods, return the Object as response using `@ResponseBody` annotation. Sample code:

```
@RequestMapping(value = EmpRestURIConstants.GET_EMP, method = RequestMethod.GET)
public @ResponseBody Employee getEmployee(@PathVariable("id") int empId) {
    logger.info("Start getEmployee. ID="+empId);

    return empData.get(empId);
}
```

97. You can invoke the rest service through any API, but if you want to use Spring then we can easily do it using RestTemplate class.

For a complete example, please read [Spring Restful Webservice Example](#).

98.

## What are some of the important Spring annotations you have used?

Some of the Spring annotations that I have used in my project are:

- **@Controller** – for controller classes in Spring MVC project.

- **@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through [Spring MVC RequestMapping Annotation Examples](#)

- **@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.

- **@PathVariable** – for mapping dynamic values from the URI to handler method arguments.

- **@Autowired** – for autowiring dependencies in spring beans.

- **@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

- **@Service** – for service classes.

- **@Scope** – for configuring scope of the spring bean.

- **@Configuration**, **@ComponentScan** and **@Bean** – for java based configurations.

- AspectJ annotations for configuring aspects and advices, **@Aspect**, **@Before**, **@After**, **@Around**, **@Pointcut** etc.

Celebrities Who Have Had Strange Pets



109.

## Can we send an Object as the response of Controller handler method?

Yes we can, using `@ResponseBody` annotation. This is how we send JSON or XML based response in restful web services.

110.

## How to upload file in Spring MVC Application?

Spring provides built-in support for uploading files through `MultipartResolver` interface implementations. It's very easy to use and requires only configuration changes to get it working. Obviously we would need to write controller handler method to handle the incoming file and process it. For a complete example, please refer [Spring File Upload Example](#).

111.

## How to validate form data in Spring Web MVC Framework?

Spring supports JSR-303 annotation based validations as well as provide `Validator` interface that we can implement to create our own custom validator. For using JSR-303 based validation, we need to annotate bean variables with the required validations.

For custom validator implementation, we need to configure it in the controller class. For a complete example, please read [Spring MVC Form Validation Example](#).

112.

## What is Spring MVC Interceptor and how to use it?

Spring MVC Interceptors are like Servlet Filters and allow us to intercept client request and process it. We can intercept client request at three places – **preHandle**, **postHandle** and **afterCompletion**.

We can create spring interceptor by implementing `HandlerInterceptor` interface or by extending abstract class `HandlerInterceptorAdapter`.

We need to configure interceptors in the spring bean configuration file. We can define an interceptor to intercept all the client requests or we can configure it for specific URI mapping too. For a detailed example, please refer [Spring MVC Interceptor Example](#).

113.

## What is Spring JdbcTemplate class and how to use it?

Spring Framework provides excellent integration with JDBC API and provides `JdbcTemplate` utility class that we can use to avoid boiler-plate code from our database operations logic such as Opening/Closing Connection, `ResultSet`, `PreparedStatement` etc.

For `JdbcTemplate` example, please refer [Spring JDBC Example](#).

114.

## How to use Tomcat JNDI DataSource in Spring Web Application?

For using servlet container configured JNDI `DataSource`, we need to configure it in the spring bean configuration file and then inject it to spring beans as dependencies. Then we can use it with `JdbcTemplate` to perform database operations.

Sample configuration would be:

```
<beans:bean id="dbDataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
  <beans:property name="jndiName" value="java:comp/env/jdbc/MyLocalDB"/>
</beans:bean>
```

For complete example, please refer [Spring Tomcat JNDI Example](#).

Celebrities Who Have Had Strange Pets



Spring framework provides transaction management support through Declarative Transaction Management as well as programmatic transaction management. Declarative transaction management is most widely used because it's easy to use and works in most of the cases.

We use annotate a method with `@Transactional` annotation for Declarative transaction management. We need to configure the transaction manager for the DataSource in the spring bean configuration file.

```
<bean id="transactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>
```

116.

## What is Spring DAO?

Spring DAO support is provided to work with data access technologies like JDBC, Hibernate in a consistent and easy way. For example we have `JdbcDaoSupport`, `HibernateDaoSupport`, `JdoDaoSupport` and `JpaDaoSupport` for respective technologies.

Spring DAO also provides consistency in exception hierarchy and we don't need to catch specific exceptions.

117.

## How to integrate Spring and Hibernate Frameworks?

We can use Spring ORM module to integrate Spring and Hibernate frameworks if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using `HibernateTemplate` or `HibernateDaoSupport` classes and better to use DAO pattern with dependency injection for the integration.

Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for Hibernate boiler-plate code for transaction management.

For better understanding you should go through following tutorials:

[Spring Hibernate Integration Example](#)

[Spring MVC Hibernate Integration Example](#)

.

120.

## What is Spring Security?

Spring security framework focuses on providing both authentication and authorization in java applications. It also takes care of most of the common security vulnerabilities such as CSRF attack.

It's very beneficial and easy to use Spring security in web applications, through the use of annotations such as `@EnableWebSecurity`. You should go through the following posts to learn how to use the Spring Security framework.

[Spring Security in Servlet Web Application](#)

[Spring MVC and Spring Security Integration Example](#)

.

123.

## How to inject a java.util.Properties into a Spring Bean?

We need to define propertyConfigurer bean that will load the properties from the given property file. Then we can use Spring Beans to inject dependencies. For example;

```
<bean id="propertyConfigurer"
      class="org.springframework.context.support.PropertySourcesPlaceholderConfigurer">
  <property name="location" value="/WEB-INF/application.properties" />
</bean>

<bean class="com.journaldev.spring.EmployeeDaoImpl">
```

Celebrities Who Have Had Strange Pets



If you are using annotation to configure the spring bean, then you can inject property like below.

```
@Value("${maxReadResults}")  
private int maxReadResults;
```

124.

## Name some of the design patterns used in Spring Framework?

Spring Framework is using a lot of design patterns, some of the common ones are:

- 125. Singleton Pattern: Creating beans with default scope.
- 126. Factory Pattern: Bean Factory classes
- 127. Prototype Pattern: Bean scopes
- 128. Adapter Pattern: Spring Web and Spring MVC
- 129. Proxy Pattern: Spring Aspect Oriented Programming support
- 130. Template Method Pattern: JdbcTemplate, HibernateTemplate etc
- 131. Front Controller: Spring MVC DispatcherServlet
- 132. Data Access Object: Spring DAO support
- 133. Dependency Injection and Aspect Oriented Programming

134.

## What are some of the best practices for Spring Framework?

Some of the best practices for Spring Framework are:

- 135. Avoid version numbers in schema reference, to make sure we have the latest configs.
- 136. Divide spring bean configurations based on their concerns such as spring-jdbc.xml, spring-security.xml.
- 137. For spring beans that are used in multiple contexts in Spring MVC, create them in the root context and initialize with listener.
- 138. Configure bean dependencies as much as possible, try to avoid autowiring as much as possible.
- 139. For application-level properties, the best approach is to create a property file and read it in the spring bean configuration file.
- 140. For smaller applications, annotations are useful but for larger applications, annotations can become a pain. If we have all the configuration in XML files, maintaining it will be easier.
- 141. Use correct annotations for components for understanding the purpose easily. For services use @Service and for DAO beans use @Repository.
- 142. Spring framework has a lot of modules, use what you need. Remove all the extra dependencies that get usually added when you create projects through Spring Tool Suite templates.
- 143. If you are using Aspects, make sure to keep the join point as narrow as possible to avoid advice on unwanted methods. Consider custom annotations that are easier to use and avoid any issues.
- 144. Use dependency injection when there is an actual benefit, just for the sake of loose-coupling don't use it because it's hard

That's all for Spring Framework interview questions. I hope these questions will help you in coming Java EE interview. I will keep adding more questions as I find them. If you know some more questions that should be part of the list, make sure to add a comment for it and I will include it.

Celebrities Who Have Had Strange Pets





Facebook

Twitter

WhatsApp

Reddit

LinkedIn

Email

Spring MVC Interceptor  
HandlerInterceptorAdapter,  
HandlerInterceptor Example  
PREV

Spring Security Example  
Tutorial

NEXT

Pankaj

I love Open Source technologies and writing about my experience about them is my passion.

Follow Author



## Comments

Prabir says:

Hi I have one Question :-

In Spring Framework's declarative transaction implementation, is it sufficient to annotate the classes with the @Transactional annotation

- a. No.
- b. Yes
- c. No. has to be added in configuration.
- d. It is necessary to add @Enable TransactionManagement in the configuration.

October 21, 2020 at 11:47 am

Reply

Rahul Dholu says:

How to resolve Circular Dependency? When it occurs?

July 6, 2020 at 2:01 am

Reply

Rahul Dholu says:

Please Add the question, Architecture of Spring Framework and Spring MVC?

June 23, 2020 at 10:09 am

Reply

Mohaseen says:

Excellent questions and answers.

June 13, 2020 at 5:29 pm

Reply

Vikas says:

Great article! It is really helpful.

Link to Spring Aware Interfaces is incorrect in point #18.

Please keep up the good work!

January 16, 2020 at 5:10 am

Reply

Myvin Barboza says:

Pankaj YOU ROCK !!!!!

kiran says:

Useful spring question and answers.

Celebrities Who Have Had Strange Pets







I have just passed a spring interview today with the help of this article that summarized everything to me truly thanks from another continent

Reply

Akash Chaudhari says:

January 17, 2019 at 10:16 am

Please add composite key in many interviews they r asking about composite key how retrieve and how to add primary keys.

Reply

Dipanshu Asri says:

November 5, 2018 at 2:20 am

Hi Pankaj,  
What happens when we create an inner bean of prototype scope in a singleton scope bean?

Reply

Swasti says:

April 25, 2019 at 2:24 am

Dipanshu , It is possible and as you mentioned the inner bean will be instantiate at the time of required (called) only.

Reply

sreenath sandra says:

August 6, 2018 at 4:12 am

Good explanation

Reply

Arun Sundaramoorthy says:

July 27, 2018 at 7:11 pm

WOW, such a nice tutorial for a person who has to start on Spring projects.

Reply

Yogesh Katkar says:

April 3, 2018 at 10:07 pm

good tutorial to learn

Reply

Sandeep says:

January 12, 2018 at 10:55 pm

Thanks a lot , Its really helpful

Reply

Gundamaiah says:

November 15, 2017 at 2:25 am

In question no.44, for property injection using annotations we need to use @PropertySource also to read the properties.

Reply

Gundamaiah says:

November 15, 2017 at 2:26 am

@PropertySource to load the properties.

Reply

Kumudam says:

August 27, 2017 at 10:51 am

Thanks a lot for sharing wonderful post

Reply

gauri says:

May 26, 2017 at 9:44 pm

hi i purchased spring+hibernate course from udemy in 15 dollar and it is really awsm for fresher as well as experienced person.

Reply

Arul says:

August 21, 2017 at 12:38 am

Hi dude,  
Could you share that Spring and hibernate notes.

Reply

Abhi says:

December 3, 2017 at 8:54 am

You can get that from Udemy. You cant share 100's of videos. It is just 10 -15 dollars.

Reply

Manisha says:

Could you please send me the name/link of the tutorial available on Udemy?  
Thanks

Ron says:

10\$ only on Udemy

Celebrities Who Have Had Strange Pets





Configure bean dependencies as much as possible, try to avoid autowiring as much as possible.  
Why should avoid auto wiring please elaborate more on this.

Reply

Debashis Das says:

November 12, 2018 at 3:15 am

You should always avoid autowiring by type/name since it is lookup-based and can be expensive in a large enterprise-level application. Also if you use @Autowired it makes testing difficult in frameworks like Mockito. Try to use @Autowired by constructor level.

Reply

Samba siva says:

January 24, 2017 at 11:51 pm

Nice startup place to start preparing for interviews.

Reply

Suraj says:

September 9, 2016 at 5:53 am

Nice work...

Reply

anreddy says:

August 25, 2016 at 11:44 am

Nice blog.. Thanks alot

Reply

Alex says:

July 22, 2016 at 12:27 pm

Wow, excellent selection of interview questions. I would like to add to the question "What's the difference between @Component, @Controller, @Repository & @Service annotations in Spring?" by say that the Repository annotation confers special behavior to all beans it marks. The PersistenceExceptionTranslationPostProcessor automatically applies persistence exception translation to any bean marked with @Repository. I have post more about this here: <https://readlearncode.com/2016/02/13/insights-from-stackoverflow-most-voted-for-spring-4-questions/#1>

Reply

Annalakshmi says:

June 13, 2016 at 8:23 am

Hi, Do you have any spring-jdbc interview questions and examples latest (4.\* version)?

Reply

Annalakshmi says:

June 9, 2016 at 12:17 pm

Thanks Pankaj, Could you please add Spring and (other) REST API integration examples also.

Reply

Pankaj says:

June 9, 2016 at 12:34 pm

Let me know which REST APIs integration you are looking for?

Reply

Annalakshmi says:

June 13, 2016 at 8:24 am

Jersey or Apache CXF with Spring.

Reply

Champ says:

April 6, 2016 at 10:55 am

very nice collection of spring quetions.

Reply

Rajesh Ingole says:

February 15, 2016 at 1:28 am

In the interview given on 14th Feb 2016 I came across below new question :  
1. What is the difference between beanfactory and applicationcontext in spring  
2. Write a code for implementing inheritance in the Spring

Reply

Lokesh says:

March 16, 2016 at 3:06 am

1. Its not a difference but you can say a relationship between them is, BeanFactory + Decorator Design Pattern = ApplicationContext, and both are the java interfaces and their relation is ApplicationContext extends BeanFactory.  
2. implementing inheritance in spring is inheriting the common properties of beans using , this will inherit all the proprieties of the parent bean into the new bean

Reply

anonoumus says:

Good explanation

Ganesh says:

Thanks for the aritcle.  
I had some questions.

Celebrities Who Have Had Strange Pets





Q. what happens if we autowire all the objects:

Please help me on this questions

Reply

upendra says:

November 6, 2016 at 12:26 pm

Hi Ganesh

@Autowired doesnot allow the programmer to enable the restrictio on specific property. it allow the programmer to enable restriction on simple, object, all

Reply

Prestigio says:

December 13, 2015 at 9:18 am

Thanks Pankaj for all the questions and answers, it's so easy and benifits .

Reply

Pankaj says:

December 13, 2015 at 9:26 am

Thanks for the appreciation, make sure to subscribe to newsletter where I share exclusive tips and free eBooks.

Reply

bala says:

September 17, 2015 at 7:33 am

Hello sir i have been following your site since 6 months awesome site sir.and also interview questions collections also great.thank u very much sir.

Reply

Abhideep Bakshi says:

June 4, 2015 at 4:25 am

"singleton: Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn't have shared instance variables otherwise it might lead to data inconsistency issues because it's not thread-safe."

But If you ceate a instance of same class twice in applicationcontext.xml with diff. id and apply session scope that dosen't mean that only one instance of that class will be created.

ie.

The singleton scope mean only one instance of the bean will create for every request.

Nice blogs on spring. Thanks a lot

Reply

Abhideep Bakshi says:

June 4, 2015 at 4:27 am

like :- bean id="abc" class="ABC" scope="singleton" \bean

bean id="abc1" class="ABC" scope="singleton" \bean

it works in Spring 3.0.5

Reply

Jai says:

March 24, 2015 at 12:21 pm

I have been involved in Spring development from last 5 years. But when it comes to interview questions, it becomes difficult to arrange your words and answer them in the best way. Thanks a lot for all the efforts that you take to write your knowledge down.

Reply

prasanna says:

March 8, 2015 at 1:28 pm

Answers for all questions are not visible in the page. Please fix the problem and let us know.

Reply

Prakash says:

January 22, 2015 at 4:21 am

It is very easy to understand and prepare for interview

Reply

Sai says:

January 16, 2015 at 5:00 am

Questions after 13 is are not visible on this page. this seems to be the issue with other topics as well. Please fix this that would be really helpful.

Reply

Sadhna says:

December 15, 2014 at 10:10 am

Nice blogs on spring . Thanks a lot.

Reply

Jaya says:

Your collection is awesome and very helpful and updated to latest technology.

Thanks for helping us!!!

YOGESH says:

where is other questions answer. all questions are not answered

Celebrities Who Have Had Strange Pets





I can't see questions-answers after 13 on this page. However I can see all in view-source of this page.  
I suspect some javascript or css hiding rest of the . Please check.

Reply

Vasant Talpada says:

October 15, 2014 at 2:05 am

Hi Pankaj,  
Sorry Pankaj, but I didn't see your request for facebook or G+ like after 13th onwards.  
I have +1 you on G+ and I can see all answers now. Thanks.

Reply

Purvash says:

October 10, 2014 at 12:44 pm

Supurb work

Reply

hemant chauhan says:

October 6, 2014 at 12:51 am

Hi JournalDev,  
Please provide answer of all questions

Reply

Pravindrasingh says:

September 27, 2014 at 1:47 pm

Would you post about the difference between Abstract factory and Spring DI?

Reply

ASHOK says:

September 23, 2014 at 5:15 am

Thank you very much Pankaj for the questions and answers, most of them are accurate and clear.

Reply

Ravi says:

August 31, 2014 at 11:08 pm

Hi Pankaj, I have a question  
Can Spring Controller instance's method return the below:  
1) null  
2) SpEL  
3) com.u2ware.springfield.view.jstl.JstlViewResolver

Reply

rakesh says:

August 14, 2014 at 8:28 pm

thanks its very helpful add more questions and answers  
when to use constructor injection and when to use setter injection with differnt scenarios?

Reply

sheetal says:

August 7, 2014 at 4:09 am

Can you please explain the difference between IOC and DI?

Reply

Amol Waghule says:

August 28, 2014 at 10:31 am

sheetal the ioc it means inverce of control it means mange the dependancy in beetween multiple calssses  
supoes compny is a class which has multiple classes like dept,employee,custom so here mange the the depency in between them , another one thing is that spring is  
ioc frame work its taking care of creation of object

Reply

xyz says:

November 19, 2014 at 11:45 pm

" RIP English "

Reply

Jack says:

July 19, 2014 at 6:42 pm

Well done! would you please consider questions about different type of testings as well such as performance testing and methods to pass such testings.

Reply

Sachin Kakkar says:

Hi Pankaj,  
I have couple of questions:  
1. how many mappings we are having in spring MVC (e.g. requestMapping, resourceMapping etc.) and what is their use  
2. Can we have empty requestMapping declaration  
`@requestMapping()`  
`public void defaultMethod(){`  
`System.out.println("I am Sachin");`

Celebrities Who Have Had Strange Pets

