Spring Interview Questions and Answers

1. 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:

1. 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.
2. Spring Framework 5.0 comes with its own Commons Logging bridge; spring-jcl instead of standard Commons Logging.
3. Support for providing spring components information through index file "META-INF/spring.components" rather than classpath scanning.
4. Spring WebFlux brings reactive programming to the Spring Framework.
5. 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.
6. 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.

2. 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.

1. Spring Security Example: Limit Number of User Session

As I said it's simple and easy when you use spring security framework or library. In fact is all declarative and no code is require to enable concurrent session disable functionality. You will need to include following xml snippet in your Spring Security Configuration file mostly named as applicaContext-security.xml. Here is sample spring security Example of limiting user session in Java web application:

```
<session-management invalid-session-url="/logout.html">
  <concurrency-control max-sessions="1" error-if-maximum-exceeded="true" />
</session-management>
```

As you see you can specify how many concurrent session per user is allowed, most secure system like online banking portals allow just one authenticate session per user. You can even specify a URL where user will be taken if they submit an invalid session identifier can be used to detect session timeout. Session-management element is used to capture session related stuff. Max-session specify how many concurrent authenticated session is allowed and if error-if-maximum-exceeded set to true it will flag error if user tries to login into another session.

2. Factory Pattern vs Dependency Injection

1) Factory pattern adds coupling between object, factory, and dependency. Object not only needs a dependent object to work properly but also a Factory object. While in case of dependency injection, Object just knows the dependency, it doesn't know anything about container or factory

2) As compared to Factory pattern, Dependency injection makes unit testing easier. If you use the factory pattern, you need to create the object you want to test, the factory and the dependent object, of course, you factor can return a mock object, but you need all this just to start with unit testing. On the other hand, if you use dependency injection, you just need to mock the dependency and inject into an object you want to test, no clutter or boilerplate is needed.

3) Dependency injection is more flexible than factory pattern. You can even switch to different DI framework e.g. Spring IOC or Google Guice.

4) One of the drawbacks of Dependency injection as compared to Factory pattern is that you need a container and configuration to inject the dependency, which is not required if you use factory design pattern.

5) Due to low coupling, DI results in much cleaner co than factory pattern. Your object looks like POJO and you also come to know what is mandatory and what is an option by looking which type of dependency injection your class is using.

If an object is injected using Setter injection, which means it's optional and can be injected at any time, while dependencies which are injected using constructor injection means they are mandatory and must be supplied in the order they are declared.

6) Another tricky scenario with using DI is creating an object with too many dependencies and worse if those are injected using constructor injection. That code becomes difficult to read. One solution to that problem is to use Facade pattern and inject dependencies by encapsulating in another object. For example, you can introduce an object say ApplicationSettings which can contain DBSetting, FileSetting and other configuration settings required by an object.

7) You should use Dependency Injection Patterns to introduce loose coupling. Use Factory Patterns if you need to delegate the creation of objects. In short, dependency injection frees your application from factory pattern boilerplate code. All the work which is required to implement a factory is already done by IOC containers like Spring and Google Guice.

8) What is view Resolver pattern ? how it work in Spring MVC
View Resolver pattern is a J2EE pattern which allows a web application to dynamically choose it's view technology e.g. HTML, JSP, Tapestry, JSF, XSLT or any other view technology. In this pattern, View resolver holds mapping of different views, controller return name of the view, which is then passed to View Resolver for selecting an appropriate view.Spring MVC framework supplies inbuilt view resolver for selecting views.

Read more: http://www.java67.com/2012/08/spring-interview-questions-answers.html#ixzz4N46HZ82K

3. Spring LifeCycle

Following is sequence of a bean lifecycle in Spring:
Instantiate - First the spring container finds the bean's definition from the XML file and instantiates the bean..
Populate properties - Using the dependency injection, spring populates all of the properties as specified in the bean definition..
Set Bean Name - If the bean implements BeanNameAware interface, spring passes the bean's id to setBeanName method.
Set Bean factory - If Bean implements BeanFactoryAware interface, spring passes the beanfactory to setBeanFactory method.
Pre Initialization - Also called postprocess of bean. If there are any bean BeanPostProcessors associated with the bean, Spring calls postProcesserBeforeInitialization method.
Initialize beans - If the bean implements IntializingBean,itsafterPropertySet method is called. If the bean has init method declaration, the specified initialization method is called.
Post Initialization - If there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization methods will be called.
Ready to use - Now the bean is ready to use by the application.
Destroy –- If the bean implements DisposableBean , it will call the destroy method .

4. 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 Spring framework is easy and fun at same time.

5. 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 IoCcontainer 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 in 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 different framework for different requirements.

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, extendable and maintainable. 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 and easy unit testing.

Read more at Dependency Injection Tutorial. We can also use Google Guice for Dependency Injectionto automate the process of dependency injection. But in most of the cases we are looking for more than just dependency injection and that's why Spring is the top choice for this.

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 sample project zip file, that you can download and play around to learn more.

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 stuffs such as Maven support, Templates for creating different types of Spring projects and tc server for better performance with Spring applications.

I like STS because it highlights the Spring components and if you are using AOP pointcuts and advices, then it clearly shows which methods will come under the specific pointcut. So rather than installing everything on our own, I prefer using STS when developing Spring based applications.

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.

What do you understand by Aspect Oriented Programming?

Enterprise applications have some common cross-cutting concerns that is applicable for 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 is 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 atSpring AOP Example.

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 is 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 the specific point in the application such as method execution, exception handling, changing object variable values etc. In Spring AOP a join points 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 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.

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 all kinds of pointcuts.
- One of the shortcoming of Spring AOP is that it can be applied only to the beans created through Spring Context.

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 make it ready for our use.

Spring Framework IoCcontainer 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.

What is a Spring Bean?

Any normal java class that is initialized by Spring IoC container is called Spring Bean. We use SpringApplicationContext 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.

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 initialize 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.

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

There are three different ways to configure Spring Bean.

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

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

1. **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:

```
2.  @Configuration
3.  @ComponentScan(value="com.journaldev.spring.main")
4.  publicclassMyConfiguration{
5.
6.      @Bean
7.      publicMyServicegetService(){
8.          returnnewMyService();
```

```
9.                  }
}
```

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

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

10. **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-scanbase-package="com.journaldev.spring"/>
```

What are different scopes of Spring Bean?

There are five scopes defined for Spring Beans.

0. **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.
1. **prototype**: A new instance will be created every time the bean is requested.
2. **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.
3. **session**: A new bean will be created for each HTTP session by the container.
4. **global-session**: This is used to create global session beans for Portlet applications.

What is Spring Bean life cycle?

Spring Beans are initialized by Spring Container and all the dependencies are also injected. When 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.

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

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

0. Implementing Spring *Aware interfaces, for these ServletContextAware and ServletConfigAware interfaces, for complete example of these aware interfaces, please read Spring Aware Interfaces
1. Using @Autowired annotation with bean variable of type ServletContext and ServletConfig. They will work only in servlet container specific environment only though.

```
2. @Autowired

ServletContextservletContext;
```

What is Bean wiring and @Autowired annotation?

The process of injection spring bean dependencies while initializing it called Spring Bean Wiring.

Usually it's best practice to do the explicit wiring of all the bean dependencies, but spring framework also supports autowiring. We can use @Autowired annotation with fields or methods for **autowiringbyType**. 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.

What are different types of Spring Bean autowiring?

There are four types of autowiring in Spring framework.

0. **autowirebyName**
1. **autowirebyType**
2. **autowire by constructor**
3. 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.

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.

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 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.

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 this.

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

What isDispatcherServlet 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'srootWebApplicationContext. It's 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.

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:beanclass="org.springframework.web.servlet.view.InternalResourceViewResolver">
```

```
        <beans:propertyname="prefix"value="/WEB-INF/views/"/>

        <beans:propertyname="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*.

What is a MultipartResolver and when its used?

MultipartResolver interface is used for uploading files – CommonsMultipartResolver andStandardServletMultipartResolver 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.

How to handle exceptions in Spring MVC Framework?

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

0. **Controller Based** – We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.
1. **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.
2. **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 additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

For a complete example, please read Spring Exception Handling Example.

How to create ApplicationContext in a Java Program?

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

0. **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.
1. **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.

2. **FileSystemXmlApplicationContext**: This is similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from anywhere in the file system.

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-context.xml,/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:importresource="spring-jdbc.xml"/>
```

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>
```

What are the minimum configurations needed to create Spring MVC application?

For creating a simple Spring MVC application, we would need to do following tasks.

- 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.

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.

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.

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:

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

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

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

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

20. 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.

How can we use Spring to create Restful Web Service returning JSON response?

We can use Spring Framework to create Restful web services that returns JSON data. Spring provides integration with Jackson JSON API that we can use to send JSON response in restful web service.

We would need to do following steps to configure our Spring MVC application to send JSON response:

0. 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>
```

6. 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:beanclass="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
<beans:propertyname="messageConverters">
<beans:list>
<beans:refbean="jsonMessageConverter"/>
</beans:list>
</beans:property>
</beans:bean>

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

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

```
@RequestMapping(value =EmpRestURIConstants.GET_EMP, method =RequestMethod.GET)
```

```
20.     public@ResponseBodyEmployeegetEmployee(@PathVariable("id")intempId){
21.     logger.info("Start getEmployee. ID="+empId);
22.
23.     returnempData.get(empId);
}
```

24. 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.

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.

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.

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.

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.

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.

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 bolier-plate code from our database operations logic such as Opening/Closing Connection, ResultSet, PreparedStatement etc.

For JdbcTemplate example, please refer Spring JDBC Example.

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 withJdbcTemplate to perform database operations.

Sample configuration would be:

```
<beans:beanid="dbDataSource"class="org.springframework.jndi.JndiObjectFactoryBean">

<beans:propertyname="jndiName"value="java:comp/env/jdbc/MyLocalDB"/>

</beans:bean>
```

For complete example, please refer Spring Tomcat JNDI Example.

How would you achieve Transaction Management in Spring?

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 transaction manager for the DataSource in the spring bean configuration file.

```
<beanid="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<propertyname="dataSource"ref="dataSource"/>

</bean>
```

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.

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.

Also 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

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 following posts to learn how to use Spring Security framework.

- Spring Security in Servlet Web Application
- Spring MVC and Spring Security Integration Example

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 EL support to inject properties into other bean dependencies. For example;

```
<beanid="propertyConfigurer"

class="org.springframework.context.support.PropertySourcesPlaceholderConfigurer">

<propertyname="location"value="/WEB-INF/application.properties"/>

</bean>


<beanclass="com.journaldev.spring.EmployeeDaoImpl">

<propertyname="maxReadResults"value="${results.read.max}"/>

</bean>
```

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

```
@Value("${maxReadResults}")

privateintmaxReadResults;
```

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:

0. Singleton Pattern: Creating beans with default scope.
1. Factory Pattern: Bean Factory classes
2. Prototype Pattern: Bean scopes
3. Adapter Pattern: Spring Web and Spring MVC
4. Proxy Pattern: Spring Aspect Oriented Programming support
5. Template Method Pattern: JdbcTemplate, HibernateTemplateetc
6. Front Controller: Spring MVC DispatcherServlet
7. Data Access Object: Spring DAO support
8. Dependency Injection and Aspect Oriented Programming

What are some of the best practices for Spring Framework?

Some of the best practices for Spring Framework are:

0. Avoid version numbers in schema reference, to make sure we have the latest configs.
1. Divide spring bean configurations based on their concerns such as spring-jdbc.xml, spring-security.xml.

2. For spring beans that are used in multiple contexts in Spring MVC, create them in the root context and initialize with listener.
3. Configure bean dependencies as much as possible, try to avoid autowiring as much as possible.
4. For application level properties, best approach is to create a property file and read it in the spring bean configuration file.
5. 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.
6. Use correct annotations for components for understanding the purpose easily. For services use @Service and for DAO beans use @Repository.
7. Spring framework has a lot of modules, use what you need. Remove all the extra dependencies that gets usually added when you create projects through Spring Tool Suite templates.
8. If you are using Aspects, make sure to keep the join pint as narrow as possible to avoid advice on unwanted methods. Consider custom annotations that are easier to use and avoid any issues.
9. Use dependency injection when there is actual benefit, just for the sake of loose-coupling don't use it because it's harder to maintain.

That's all for Spring Framework interview

49. How do you control concurrent Active session using Spring Security?

In fact is all declarative and no code is require to enable concurrent session disable functionality. You will need to include following xml snippet in your Spring Security Configuration file mostly named as applicaContext-security.xml. Here is sample spring security Example of limiting user session in Java web application:

```
<session-management invalid-session-url="/logout.html">
   <concurrency-control max-sessions="1" error-if-maximum-exceeded="true" />
</session-management>
```

what is Bean Factory, have you used XMLBeanFactory?

Ans: BeanFactory is factory Pattern which is based on IOC design principles.it is used to make a clear separation between application configuration and dependency from actual code. The XmlBeanFactory is one of the implementations of bean Factory which we have used in our project.
The **org.springframework.beans.factory.xml.**XmlBeanFactoryis used to create bean instance defined in our XML file.

```
BeanFactory factory = new XmlBeanFactory(new FileInputStream("beans.xml"));
```

Or

```
ClassPathResourceresorce = new ClassPathResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(resorce);
```

What are the difference between BeanFactory and ApplicationContext in spring?

ApplicationContext.	**BeanFactory**
Here we can have more than one config files possible	In this only one config file or .xml file
Application contexts can publish events to beans that are registered as listeners	Doesn't support.

Support internationalization (I18N) messages	It's not
Support application life-cycle events, and validation.	Doesn't support.
Supports many enterprise services such JNDI access, EJB integration, remoting	Doesn't support.

What is the difference between singleton and prototype bean?

Singleton: means single bean definition to a single object instance per Spring IOC container.
Prototype: means a single bean definition to any number of object instances.
Whatever beans we defined in spring framework are singleton beans. There is an attribute in bean tag named 'singleton' if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default, it is set to true. So, all the beans in spring framework are by default singleton beans.

```
<bean id="createNewStock"
  class="springexample.stockMarket.CreateNewStockAccont" singleton="false">
     <property name="newBid"/>
</bean>
```

What type of transaction Management Spring support?

Ans: This spring interview questions is little difficult as compared to previous questions just because **transaction management** is a complex concept and not every developer familiar with it. Transaction management is critical in any applications that will interact with the database. The application has to ensure that the data is consistent and the integrity of the data is maintained. Following two type of transaction management is supported by spring:
1. Programmatic transaction management
2. Declarative transaction management.

1) What is a spring?

Spring is set to be a framework which helps Java programmer for development of code and it provides IOC container, Dependency Injector, MVC flow and many other APIs for the java programmer.

2) What are Advices in Spring?

It is the execution of an aspect. Advice is like making your application learn a new trick. They are usually introduced at joinpoints.

3) What is the default scope of bean in Spring framework?

The default scope of bean is Singleton for Spring framework.

4) Name the types of transaction management that are supported by Spring?

Transaction management supported by Spring are :

- Declarative transaction management.
- Programmatic transaction management.

5) Is Singleton beans are thread safe in Spring Framework?

No, singleton beans are not thread-safe in Spring framework.

6) What are the benefits of Spring Framework?

Following are the benefits of Spring framework:

- Extensive usage of Components
- Reusability
- Decoupling
- Reduces coding effort by using pattern implementations such as singleton, factory, service locator etc.
- Removal of leaking connections
- Declarative transaction management
- Easy to integrate with third party tools and technologies.

7) What is Bean Factory?

Bean Factory is core of the spring framework and, it is a Lightweight container which loads bean definitions and manages your beans. Beans are configured using XML file and manage singleton defined bean. It is also responsible for life cycle methods and injects dependencies. It also removes adhoc singletons and factories.

8) Define Bean Wiring?

Bean wiring is the creation of associations between application components that are between the beans in a particular spring container.

9) What is called Spring MVC?

A Spring MVC is a single shared controller instance and it is used to handle request type controllers, interceptors which run in the IoC container. It also allows multiple Dispatcher Servlets which can share application context interface but not class based interface.

10) Why Spring framework is needed?

Spring framework is needed because it is –

- Very Light Weight Container
- Framework
- IOC
- AOP

11) Name the various modules used in spring framework?

- AOP module (Aspect Oriented Programming)
- JDBC abstraction and DAO module
- The Core container module
- MVC framework module
- Application context module
- O/R mapping integration module (Object/Relational)
- Web module

12) Explain the RowCallbackHandler in Spring?

The RowCallbackHandler is called for each row in ResultSet and is used to read values from the ResultSet.

13) Define Application context module?

This is a very important module and supplies various necessary services like EJB integration, remoting, JNDI access and scheduling. It transforms spring into a framework. It also broadens the idea of BeanFactory by application of lifecycle events, providing support for internationalization messages and validation.

14) Write about AOP module?

AOP module is utilized for creating aspects for Spring applications. It also enables support for metadata programming in Spring.

15) What is a BeanFactory Interface?

Bean factory interface is used to provide configuration framework for object creation and basic functionality around object management.

16) State the differences between ApplicationContext and BeanFactory in spring?

- ApplicationContext allows more than one config files to exist while BeanFactory only permits one.
- ApplicationContext can print events to beans registered as listeners. This feature is not supported by BeanFactory.
- ApplicationContext also provides support for application of lifecycle events, internationalization messages and validation and also provides services like EJB integration, remoting, JNDI access and scheduling. These features too are not supported by Bean Factory.

17) What is Auto Wiring?

Autowiring is used to build relationships between the collaborating beans. Spring container can automatically resolve collaborators for beans.

18) What are the different Modes of Autowiring?

Autowiring has five different modes:

- no: no autowire
- byName : Autowiring that can be done by property name
- byType : property type as autowired
- constructor: It is similar to byType and it is property is in constructor
- autodetect : Spring is allowed to select autowiring from byType or constructor

19) How to start using spring?

Following steps needs to be done to start with the Spring:

- Download Spring and its dependent file from spring's site.
- Create application context xml to define beans and its dependencies
- Integrate application context xml with web.xml
- Deploy and Run the application

20) What are the methods of bean life cycle?

There are two important methods of Bean life cycle:

- Setup – called when bean is loaded into container
- Teardown – called when bean is unloaded into container

21) What are the different types of events of Listeners?

Following are the different types of events of listeners:

- ContextClosedEvent – This event is called when the context is closed.
- ContextRefreshedEvent – This event is called when context is initialized or refreshed
- RequestHandledEvent – This event is called when the web context handles request

22) Differentiate between singleton and prototype bean?

Singleton means only one bean is defined per object instance while Prototype means one definition to more than one object instances in Spring.

23) What are the types of Dependency Injection?

Two types of dependency injection are supported by spring framework:

- Setter Injection
- Constructor Injection

24) Write about Core container module?

Core container module is responsible for the basic functionality of the spring framework. The whole Spring framework is built with this module as a base.

25) What is AOP module?

This AOP module is used for spring enabled application. Support has been provided AOP alliance to ensure the interoperability between spring and other AOP frameworks.

It instructs spring to add annotations to the source code and tell how to apply aspects.

26) What is AOP Alliance?

AOP alliance is an open-source project which is aimed at promoting adoption of AOP. The AOP alliance's goal is to define a common set of components and interfaces so as to improve interoperability among different AOP implementations.

27) What is called spring configuration file?

Spring configuration file is an XML file and it contains class information. It also describes how these classes are configured and interact with each other.

28) What are different types of Autowire?

There are four different types of Auto wire:

- byName
- byType
- constructor
- autodetect

29) What are the types of the transaction management that is supported by spring?

Following are the types of transaction management that has been supported by spring:

- declarative
- programmatically

30) When are declarative and programmatic transaction management used?

When only a small amount of transactional operations is there, it is advised to use Programmatic transaction management. But if there is a big amount of transactional operations to be taken care of, declarative transaction management is preferred.

31) What is IOC?

IOC (Inversion of Control pattern) is also known as dependency injection. IOC directs the programmers to depict how to create objects instead of actually creating them. But in this design pattern, this control has been given to assembler and assembler will instantiate required class if needed.

32) Write about the different types of Listener related events?

The different types of events related to listeners are:

- ContextRefreshedEvent – This gets called when the context is refreshed or initialized.
- RequestHandledEvent – This gets called when the web context is handling a request.
- ContextClosedEvent – This gets called when the context gets closed.

33) What is an Aspect?

Aspect is also called as logging which is required throughout the application. Logging or aspect is a cross cutting functionality in an application using AOP.

34) What is a Joinpoint?

The point where an aspect can be introduced in the application is known as a joinpoint. This point could be a field being modified, a method being called or even an exception being thrown. At these points, the new aspect's code can be added to introduce a new behavior to the application.

Aspect code can be inserted at this point into normal flow of application to change the current behavior.

35) What is called an Advice?

Advice will tell application on new behavior and it is the implementation of an aspect. It is inserted into an application at the joinpoint.

Advice is the implementation of an aspect. It is something like telling your application of a new behavior. Generally, the advice is inserted into an application at joinpoints.

36) What is a Pointcut?

Pointcut is used to allow where the advice can be applied.

37) What is weaving?

Weaving is used to create new proxy object by applying aspects to target object.

38) What is difference between singleton and prototype bean?

Singleton Bean – Single bean definition to a single object instance per Spring IOC container

Prototype Bean – Single bean definition to any number of object instances per Spring IOC Container

39) In what points, can weaving be applied?

Following are the points where weaving can be applied:

- Compile Time

- Class load Time
- Runtime

40) What are the different types of AutoProxying?

Following are the different types of AutoProxying:

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator
- Metadata autoproxying

41) How can beans be made singleton or prototype?

The bean tag has an attribute called 'singleton'. The bean is singleton if its value is 'TRUE', otherwise the bean is a prototype.

42) What classes are used to Control the database connection?

Following are the classes that are used to control database connection:

- Data Source Utils
- SmartData Source
- AbstractData Source
- SingleConnectionDataSource
- DriverManagerDataSource
- TransactionAwareDataSourceProxy
- DataSource TransactionManager

43) Describe about DAO in Spring framework?

DAO is used to provide integration of Java database connectivity and Object relational mapping objects. DAO is spring framework provides connection for JDBC, hibernate, JDO, JPA, Common client interface and Oracle.

44) What is Autoproxying?

Autoproxying is used to create proxy automatically for the spring users. It provides following two classes to support this automatic proxy creation:

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator

45) What is Metadata Autoproxying?

Metadata Autoproxying can be performed inspiring which can be driven by metadata. This is determined by source level attributes and keeps metadata inside the source code.

This maintains metadata in one place and mainly used for declarative transaction support.

46) What is 'Throws advice' in Spring?

'Throws Advice' define the behavior when an exception occurs. It is an interface and it has no methods which need to be implemented.

A class that implements this interface should have method with this signature:

- Void samplethrow (Throw table t)

- Void samplethrow(Method m, Object[] o, Object target, Throw tablet)

47) What are the various editors used in spring work?

The various custom editors provided by the Spring Framework are:

- PropertyEditor
- URLEditor
- ClassEditor
- CustomDateEditor
- FileEditor
- LocaleEditor
- StringArrayPropertyEditor
- StringTrimmerEditor

48) What are the advantages of spring framework?

Following are the advantages of spring framework:

- Layered Architecture
- Enables Plain Old Java Object (POJO) Programming and it enables continuous integration and testability
- Dependency Injection and Inversion of Control that simplifies JDBC
- Open source framework which can be used for commercial purpose

49) How is Hibernate accessed using the Spring framework?

Hibernate can be accessed in the following two ways:

- By IOC with a Callback and HibernateTemplate.
- By applying an AOP Interceptor and broadening the HibernateDaoSupport.

50) What are the various Channels supported by Spring 2.0?

Following are the channels supported by spring version 2.0:

- Pollable Channel
- Subscribable Channel
- PublishSubscribe Channel
- Queue Channel
- Priority Channel
- Rendezvous Channel
- Direct Channel
- Executor Channel
- Scoped Channel

51) Why is declarative transaction management preferred in Spring?

Declarative transaction management has minimum impact on the application code and, therefore, is an idealistic lightweight container.

52) Explain the concept of a BeanFactory?

BeanFactory applies the idea of a factory pattern that utilizes IOC to separate the application's dependencies and configuration from the actual code.

53) What are the different scopes of spring bean?

Scopes of spring bean are Singleton, prototype, request, session and global session.

54) What are all the ways to access Hibernate by using Spring?

There are two ways to access hibernate using spring:

- Inversion of Control with a Hibernate Template and Callback
- Extending HibernateDAOSupport and Applying an AOP Interceptor node.

55) How struts application can be integrated with spring?

There are two options for struts application that can be integrated with spring:

Configuration of Spring to manage beans using ContextLoader plugin and set their dependencies in a spring context file

Grab spring managed beans explicitly using agetwebapplicationcontext()

56) What is Inversion of control (IOC)?

Inversion of Control (IOC) is also called as dependency Injection which is nothingbut a design pattern that gives control to the assembler of classes. In general, class will instantiate another class if required.

But in this design pattern, this control has been to given to assembler and assembler will instantiate required class if needed.

57) Write the benefits of using IOC?

The major benefits of dependency injection or IOC are that it reduces the amount of coding required for the application. This allows the testing of the application to be done quickly and easily as no JNDI lookup mechanism or singletons are required. IOC containers also support lazy loading and eager installation of services.

58) What is Inner bean? What is the drawback of inner bean?

If a bean element is directly embedded in a property tag while wiring beans, then the bean is called Inner Bean. Its drawback is that it cannot be reprocessed.

59)What are the different types of Injection in spring?

There are three types of Injection in spring:

- Setter Injection
- Constructor Injection
- Getter or Method Injection

60) What are the benefits of spring framework?

Following are the benefits of spring framework:

- Light weight container when compared to j2EE containers
- Built in Web MVC framework
- Creates loosely coupled applications
- Supports aspect oriented programming like logging, transaction and security
- Configuration done in XML format which is easy to write and understand

61) What are the types of Advice?

There are five types of Advice:

- Before Advice

- After returning advice
- After throwing advice
- Finally advice
- Around advice

62) What is called PreparedStatementCreator?

PreparedStatementCreator is one of the most commonly used interfaces for writing data to the database. createPreparedStatement() is a method that can be used to create and return PreparedStatement from the Connection argument, and exception handling is automatically taken care of. When this interface is implemented, a different interface SqlProvider can also be implemented which has a method called getSql(). This method is useful for providing sql strings to the JdbcTemplate. It does not handle SQLExceptions.

63) What is SQLProvider?

SQLProvider has only one method called getSql()and it is implemented using PreparedStatementCreator implementers. It is mainly used for debugging.

64) Write about BatchPreparedStatementSetter?

BatchPreparedStatementSetter is used to update more than a single row in one go, they can use BatchPreparedStatementSetter. This interface provides two methods they are

- setValues(PreparedStatementps, inti) throws SOL exception
- intgetBatchSize

65) What is the better method of using JDBC in Spring?

If JDBC is used with the template class called JdbcTemplate, it gives a better performance.

66) What exceptions do the DAO classes, use in Spring throw?

In spring DAO classes only throws SQLException.

67) Explain the advantages of using DAO module?

The database code can be kept clean and simple by using the DAO module. This helps in preventing problems that rise because of poor handling of closures of database resources. Also, the DAO module utilizes the AOP module to enable objects in the Spring application to use transaction management services.

68) Name the significant ApplicationContext implementations used in the spring framework?

They are:

- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext

69) How is a bean added to a Spring application?

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN">
```

```
4
5 <beans>
6
7 <bean id="foo"/>
8
9 <bean id="bar"/>
10
11 </beans>
```

The bean tag has an ID attribute which stores the bean name and a class attributes which specifies the full class name.

70) What are ORM integration modules?

Object/relational mapping (ORM) tool is supported by Spring over straight JDBC by implementing the ORM module. Spring can join various important ORM frameworks, including JDO, iBATIS SQL Maps and Hibernate.

71) Mention and explain the types of Advice in Spring?

Types of advice are:

- Before advice: Advice that is executed prior to a joinpoint is called the 'before advice'.
- After returning advice: Advice that is executed after the normal completion of a joinpoint is called the 'after returning advice'.
- After throwing advice: Advice that is executed only if a method exits abnormally by throwing an exception, is called the 'after throwing advice'.
- After (finally) advice: Advice that is executed irrespective of how a joinpoint exits is called 'after finally advice'.
- Around advice: Advice that borders a joinpoint, for example, a method invocation, is called an 'around advice'. This can be used to perform special activities before and after the invocation of method.

72) What is the web module?

The web module enables the creation of a web application without XML. The web.xml file needs to be configured for using the web module.

73) What is DataAccessException?

DataAccessException is a RuntimeException. It is an Unchecked Exception. The user cannot be forced to handle these kinds of exceptions.

74) What is XMLBeanFactory?

Spring includes several applications of Bean factory. Out of these, org.springframework.beans.factory.xml.XmlBeanFactory is a very important one. It loads the beans on the basis of the definitions stored in an XML file. For the creation of an XmlBeanFactory, a java.io.InputStream is passed to the constructor. The InputStream provides the XML to the factory. For example, for retrieval of the bean, the getBean() method is called by passing the name of the desired bean.

```
1 MyBean helloBean = (MyBean) factory.getBean("helloBean");
```

75) Name the Exception class which is connected to the exceptions thrown by the applications?

It is the DataAccessException given by org.springframework.dao.DataAccessException

76) Mention the types of IOC (dependency injection)?

The different types of IoC are: –

- Setter Injection: With the help of JavaBeans properties.
- Constructor Injection: Dependencies are given in the form of constructor parameters.
- Interface Injection: With the help of an interface, an Injection is performed.

Out of these three, only construction and setter are being used in Spring.

77) What are the important beans lifecycle methods?

All in all, two bean lifecycle methods are there. The first method is the setup method which is called during the loading of the bean into the container. The second is when the bean is unloaded from the container, and this method is called the teardown.

78) How can the default lifecycle methods of beans be nullified?

The tag, bean, has two useful attributes which can be used to define special initialization and destruction methods.

For Example, two new methods forSetup and forTeardown can be added to the Foo class in the following way:

<beans>

<bean id="bar" init-method=”forSetup”destroy=”forTeardown”/>

</beans>

79) What is a Target?

A target is the class that is advised. This class can either be a class to which we want to add a special behavior to or a third party class. The target class is free to center on its major concern using the AOP concepts, regardless of any advice that is being applied.

80) Explain the term Proxy?

The term proxy refers to an object which is produced the application of an advice to the target object.

81) What is cross cutting concern and concern in spring AOP?

Cross cutting concern: It is a concern which is applicable throughout the application and it affects the entire application. E.g Security, logging and data transfer are the concerns which are needed in almost every module of an application.

Concern: Concern is a behavior that we want to have in a module of an application. Issues in which we are interested defines our concern.

Spring overview

1. What is Spring?

Spring is an open source development framework for Enterprise Java. The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make Java EE development easier to use and promote good programming practice by enabling a POJO-based programming model.

2. What are benefits of Spring Framework?

- **Lightweight:** Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.
- **Inversion of control (IOC):** Loose coupling is achieved in Spring, with the Inversion of Control technique. The objects give their dependencies instead of creating or looking for dependent objects.
- **Aspect oriented (AOP):** Spring supports Aspect oriented programming and separates application business logic from system services.
- **Container:** Spring contains and manages the life cycle and configuration of application objects.
- **MVC Framework:** Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks.
- **Transaction Management:** Spring provides a consistent transaction management interface that can scale down to a local transaction and scale up to global transactions (JTA).
- **Exception Handling:** Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO) into consistent, unchecked exceptions.

3. Which are the spring framework modules?

The basic modules of the Spring framework are :

- Core module
- Bean module
- Context module
- Expression Language module
- JDBC module
- ORM module
- OXM module
- Java Messaging Service(JMS) module
- Transaction module
- Web module
- Web-Servlet module
- Web-Struts module

- Web-Portlet module

Explain the different modules in Spring framework.

Different modules offered by Spring Framework are :

Inversion of control container : Configuration of application components and lifecycle management of java objects.

Aspect-oriented programming : Enables implementation of cross-cutting routines.

Data access : Working with relational database management systems on the java platform using JDBC and object-relational mapping tools.

Transaction management : Unifies several transaction management APIs and coordinates transactions for Java objects.

Model-View-Controller : An HTTP and Servlet-based framework providing hooks for extension and customization.

Remote Access Framework : Configurative RPC-style export and import of java objects over networks supporting RMI, CORBA and HTTP-based protocols including the web services (SOAP).

Batch processing : A framework for high-volume processing featuring reusable functions including logging/tracing, transaction management, job processing statistics, job restart, skip, and resource management.

Authentication and authorization : Configurable security processes that support a range of standards, protocols, tools and practices via the Spring Security sub-project (formerly Acegi).

Remote Management : Configurative exposure and management of java objects for local or remote configuration via JMX.

Messaging : Configurative registration of message listener objects for transparent message consumption from message queues via JMS, improvement of message sending over standard JMS APIs.

Testing : Support classes for writing unit tests and integration tests.

4. Explain the Core Container (Application context) module

This is the basic Spring module, which provides the fundamental functionality of the Spring framework. `BeanFactory` is the heart of any spring-based application. Spring framework was built on the top of this module, which makes the Spring container.

5. BeanFactory – implementation example

A `BeanFactory` is an implementation of the factory pattern that applies Inversion of Control to separate the application's configuration and dependencies from the actual application code.
The most commonly used `BeanFactory` implementation is the `XmlBeanFactory` class.

6. XMLBeanFactory

The most useful one is `org.springframework.beans.factory.xml.XmlBeanFactory`, which loads its beans based on the definitions contained in an XML file. This container reads the configuration metadata from an XML file and uses it to create a fully configured system or application.

7. Explain the AOP module

The AOP module is used for developing aspects for our Spring-enabled application. Much of the support has been provided by the AOP Alliance in order to ensure the interoperability between Spring and other AOP frameworks. This module also introduces metadata programming to Spring.

8. Explain the JDBC abstraction and DAO module

With the JDBC abstraction and DAO module we can be sure that we keep up the database code clean and simple, and prevent problems that result from a failure to close database resources. It provides a layer of meaningful exceptions on top of the error messages given by several database servers. It also makes use of Spring's AOP module to provide transaction management services for objects in a Spring application.

9. Explain the object/relational mapping integration module

Spring also supports for using of an object/relational mapping (ORM) tool over straight JDBC by providing the ORM module. Spring provides support to tie into several popular ORM frameworks, including Hibernate, JDO, and iBATIS SQL Maps. Spring's transaction management supports each of these ORM frameworks as well as JDBC.

10. Explain the web module

The Spring web module is built on the application context module, providing a context that is appropriate for web-based applications. This module also contains support for several web-oriented tasks such as transparently handling multipart requests for file uploads and programmatic binding of request parameters to your business objects. It also contains integration support with Jakarta Struts.

11. Explain the Spring MVC module

MVC framework is provided by Spring for building web applications. Spring can easily be integrated with other MVC frameworks, but Spring's MVC framework is a better choice, since it uses IoC to provide for a clean separation of controller logic from business objects. With Spring MVC you can declaratively bind request parameters to your business objects.

12. Spring configuration file

Spring configuration file is an XML file. This file contains the classes information and describes how these classes are configured and introduced to each other.

13. What is Spring IoC container?

The Spring IoC is responsible for creating the objects, managing them (with dependency injection (DI)), wiring them together, configuring them, as also managing their complete lifecycle.

14. What are the benefits of IOC?

IOC or dependency injection minimizes the amount of code in an application. It makes easy to test applications, since no singletons or JNDI lookup mechanisms are required in unit tests. Loose coupling is promoted with minimal effort and least intrusive mechanism. IOC containers support eager instantiation and lazy loading of services.

Implementation techniques

- Using a factory pattern
- Using a service locator pattern
- Using a dependency injection, for example
  - A constructor injection
  - A parameter injection
  - A setter injection
  - An interface injection
- Using a contextualized lookup
- Using template method design pattern
- Using strategy design pattern

15. What are the common implementations of the ApplicationContext?

The **FileSystemXmlApplicationContext** container loads the definitions of the beans from an XML file. The full path of the XML bean configuration file must be provided to the constructor.
The **ClassPathXmlApplicationContext** container also loads the definitions of the beans from an XML file. Here, you need to set `CLASSPATH` properly because this container will look bean configuration XML file in `CLASSPATH`.
The **WebXmlApplicationContext:** container loads the XML file with definitions of all beans from within a web application.

16. What is the difference between Bean Factory and ApplicationContext?

Application contexts provide a means for resolving text messages, a generic way to load file resources (such as images), they can publish events to beans that are registered as listeners. In addition, operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context. The application context implements `MessageSource`, an interface used to obtain localized messages, with the actual implementation being pluggable.

17. What does a Spring application look like?

- An interface that defines the functions.
- The implementation that contains properties, its setter and getter methods, functions etc.,
- Spring AOP
- The Spring configuration XML file.
- Client program that uses the function

Dependency Injection

18. What is Dependency Injection in Spring?

Dependency Injection, an aspect of Inversion of Control (IoC), is a general concept, and it can be expressed in many different ways.This concept says that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (the IOC container) is then responsible for hooking it all up.

19. What are the different types of IoC (dependency injection)?

- **Constructor-based dependency injection:** Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.
- **Setter-based dependency injection:** Setter-based DI is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

20. Which DI would you suggest Constructor-based or setter-based DI?

You can use both Constructor-based and Setter-based Dependency Injection. The best solution is using constructor arguments for mandatory dependencies and setters for optional dependencies.

21. What are Spring beans?

The Spring Beans are Java Objects that form the backbone of a Spring application. They are instantiated, assembled, and managed by the Spring IoC container. These beans are created with the configuration metadata that is supplied to the container, for example, in the form of XML `<bean/>` definitions.
Beans defined in spring framework are singleton beans. There is an attribute in bean tag named `"singleton"` if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

22. What does a Spring Bean definition contain?

A Spring Bean definition contains all configuration metadata which is needed for the container to know how to create a bean, its lifecycle details and its dependencies.

23. How do you provide configuration metadata to the Spring Container?

There are three important methods to provide configuration metadata to the Spring Container:

- XML based configuration file.
- Annotation-based configuration
- Java-based configuration

24. How do you define the scope of a bean?

When defining a `<bean>` in Spring, we can also declare a scope for the bean. It can be defined through the `scope` attribute in the bean definition. For example, when Spring has to produce a new bean instance each time one is needed, the bean's `scope` attribute to be `prototype`. On the other hand, when the same instance of a bean must be returned by Spring every time it is needed, the the bean `scope` attribute must be set to `singleton`.

25. Explain the bean scopes supported by Spring

There are five scoped provided by the Spring Framework supports following five scopes:

- In **singleton** scope, Spring scopes the bean definition to a single instance per Spring IoC container.
- In **prototype** scope, a single bean definition has any number of object instances.
- In **request** scope, a bean is defined to an HTTP request. This scope is valid only in a web-aware Spring ApplicationContext.
- In **session** scope, a bean definition is scoped to an HTTP session. This scope is also valid only in a web-aware Spring ApplicationContext.
- In **global-session** scope, a bean definition is scoped to a global HTTP session. This is also a case used in a web-aware Spring ApplicationContext.

The default scope of a Spring Bean is `Singleton`.

26. Are Singleton beans thread safe in Spring Framework?

No, singleton beans are not thread-safe in Spring framework.

27. Explain Bean lifecycle in Spring framework

- The spring container finds the bean's definition from the XML file and instantiates the bean.
- Spring populates all of the properties as specified in the bean definition (DI).
- If the bean implements `BeanNameAware` interface, spring passes the bean's id to `setBeanName()` method.
- If Bean implements `BeanFactoryAware` interface, spring passes the `beanfactory` to `setBeanFactory()` method.
- If there are any bean `BeanPostProcessors` associated with the bean, Spring calls `postProcesserBeforeInitialization()` method.
- If the bean implements `IntializingBean`, its `afterPropertySet()` method is called. If the bean has init method declaration, the specified initialization method is called.
- If there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization() methods will be called.
- If the bean implements `DisposableBean`, it will call the `destroy()` method.

28. Which are the important beans lifecycle methods? Can you override them?

There are two important bean lifecycle methods. The first one is `setup` which is called when the bean is loaded in to the container. The second method is the `teardown` method which is called when the bean is unloaded from the container.

The `bean` tag has two important attributes (`init-method` and `destroy-method`) with which you can define your own custom initialization and destroy methods. There are also the correspondiveannotations(`@PostConstruct` and `@PreDestroy`).

29. What are inner beans in Spring?

When a bean is only used as a property of another bean it can be declared as an inner bean. Spring's XML-based configuration metadata provides the use of `<bean/>` element inside the `<property/>` or `<constructor-arg/>` elements of a bean definition, in order to define the so-called inner bean. Inner beans are always anonymous and they are always scoped as prototypes.

30. How can you inject a Java Collection in Spring?

Spring offers the following types of collection configuration elements:

- The `<list>` type is used for injecting a list of values, in the case that duplicates are allowed.
- The `<set>` type is used for wiring a set of values but without any duplicates.
- The `<map>` type is used to inject a collection of name-value pairs where name and value can be of any type.
- The `<props>` type can be used to inject a collection of name-value pairs where the name and value are both Strings.

31. What is bean wiring?

Wiring, or else bean wiring is the case when beans are combined together within the Spring container. When wiring beans, the Spring container needs to know what beans are needed and how the container should use dependency injection to tie them together.

32. What is bean auto wiring?

The Spring container is able to autowire relationships between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for a bean by inspecting the contents of the `BeanFactory` without using `<constructor-arg>` and `<property>` elements.

33. Explain different modes of auto wiring?

The autowiring functionality has five modes which can be used to instruct Spring container to use autowiring for dependency injection:

- **no:** This is default setting. Explicit bean reference should be used for wiring.
- **byName:** When autowiring `byName`, the Spring container looks at the properties of the beans on which `autowire` attribute is set to `byName` in the XML configuration file. It then tries to match and wire its properties with the beans defined by the same names in the configuration file.
- **byType:** When autowiring by `datatype`, the Spring container looks at the properties of the beans on which `autowire` attribute is set to `byType` in the XML configuration file. It then tries to match and wire a property if its type matches with exactly one of the beans name in configuration file. If more than one such beans exist, a fatal exception is thrown.

- **constructor:** This mode is similar to `byType`, but type applies to constructor arguments. If there is not exactly one bean of the constructor argument type in the container, a fatal error is raised.
- **autodetect:** Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowireby `byType`.

34. Are there limitations with autowiring?

Limitations of autowiring are:

- **Overriding:** You can still specify dependencies using `<constructor-arg>` and `<property>` settings which will always override autowiring.
- **Primitive data types:** You cannot autowire simple properties such as primitives, Strings, and Classes.
- **Confusing nature:** Autowiring is less exact than explicit wiring, so if possible prefer using explicit wiring.

35. Can you inject null and empty string values in Spring?

Yes, you can.

Spring Annotations

36. What is Spring Java-Based Configuration? Give some annotation example.

Java based configuration option enables you to write most of your Spring configuration without XML but with the help of few Java-based annotations.
An example is the `@Configuration` annotation, that indicates that the class can be used by the Spring IoC container as a source of bean definitions. Another example is the `@Bean` annotated method that will return an object that should be registered as a bean in the Spring application context.

37. What is Annotation-based container configuration?

An alternative to XML setups is provided by annotation-based configuration which relies on the bytecode metadata for wiring up components instead of angle-bracket declarations. Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration.

38. How do you turn on annotation wiring?

Annotation wiring is not turned on in the Spring container by default. In order to use annotation based wiring we must enable it in our Spring configuration file by configuring `<context:annotation-config/>` element.

39. @Required annotation

This annotation simply indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring. The container throws `BeanInitializationException` if the affected bean property has not been populated.

40. @Autowired annotation

The `@Autowired` annotation provides more fine-grained control over where and how autowiring should be accomplished. It can be used to autowire bean on the setter method just like `@Required` annotation, on the constructor, on a property or pn methods with arbitrary names and/or multiple arguments.

41. @Qualifier annotation

When there are more than one beans of the same type and only one is needed to be wired with a property, the `@Qualifier` annotation is used along with `@Autowired` annotation to remove the confusion by specifying which exact bean will be wired.

Spring Data Access

42. How can JDBC be used more efficiently in the Spring framework?

When using the Spring JDBC framework the burden of resource management and error handling is reduced. So developers only need to write the statements and queries to get the data to and from the database. JDBC can be used more efficiently with the help of a template class provided by Spring framework, which is the `JdbcTemplate` (example here).

43. JdbcTemplate

`JdbcTemplate` class provides many convenience methods for doing things such as converting database data into primitives or objects, executing prepared and callable statements, and providing custom database error handling.

44. Spring DAO support

The Data Access Object (DAO) support in Spring is aimed at making it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows us to switch between the persistence technologies fairly easily and to code without worrying about catching exceptions that are specific to each technology.

45. What are the ways to access Hibernate by using Spring?

There are two ways to access Hibernate with Spring:

- Inversion of Control with a Hibernate Template and Callback.
- Extending `HibernateDAOSupport` and Applying an AOP Interceptor node.

46. ORM's Spring support

Spring supports the following ORM's:

- Hibernate
- iBatis
- JPA (Java Persistence API)
- TopLink

- JDO (Java Data Objects)
- OJB

47. How can we integrate Spring and Hibernate using HibernateDaoSupport?

Use Spring's `SessionFactory` called `LocalSessionFactory`. The integration process is of 3 steps:

- Configure the Hibernate SessionFactory
- Extend a DAO Implementation from `HibernateDaoSupport`
- Wire in Transaction Support with AOP

48. Types of the transaction management Spring support

Spring supports two types of transaction management:

- **Programmatic transaction management:** This means that you have managed the transaction with the help of programming. That gives you extreme flexibility, but it is difficult to maintain.
- **Declarative transaction management:** This means you separate transaction management from the business code. You only use annotations or XML based configuration to manage the transactions.

49. What are the benefits of the Spring Framework's transaction management?

- It provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.
- It provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
- It supports declarative transaction management.
- It integrates very well with Spring's various data access abstractions.

50. Which Transaction management type is more preferable?

Most users of the Spring Framework choose declarative transaction management because it is the option with the least impact on application code, and hence is most consistent with the ideals of a non-invasive lightweight container. Declarative transaction management is preferable over programmatic transaction management though it is less flexible than programmatic transaction management, which allows you to control transactions through your code.

Spring Aspect Oriented Programming (AOP)

51. Explain AOP

Aspect-oriented programming, or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management.

52. Aspect

The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. It ia a module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging. An application can have any number

of aspects depending on the requirement. In Spring AOP, aspects are implemented using regular classes annotated with the `@Aspect` annotation (`@AspectJ` style).

53. What is the difference between concern and cross-cutting concern in Spring AOP

The Concern is behavior we want to have in a module of an application. A Concern may be defined as a functionality we want to implement.
The cross-cutting concern is a concern which is applicable throughout the application and it affects the entire application. For example, logging, security and data transfer are the concerns which are needed in almost every module of an application, hence they are cross-cutting concerns.

54. Join point

The join point represents a point in an application where we can plug-in an AOP aspect. It is the actual place in the application where an action will be taken using Spring AOP framework.

55. Advice

The advice is the actual action that will be taken either before or after the method execution. This is actual piece of code that is invoked during the program execution by the Spring AOP framework.
Spring aspects can work with five kinds of advice:

- **before:** Run advice before the a method execution.
- **after:** Run advice after the a method execution regardless of its outcome.
- **after-returning:** Run advice after the a method execution only if method completes successfully.
- **after-throwing:** Run advice after the a method execution only if method exits by throwing an exception.
- **around:** Run advice before and after the advised method is invoked.

56. Pointcut

The pointcut is a set of one or more joinpoints where an advice should be executed. You can specify pointcuts using expressions or patterns.

57. What is Introduction?

An Introduction allows us to add new methods or attributes to existing classes.

58. What is Target object?

The target object is an object being advised by one or more aspects. It will always be a proxy object. It is also referred to as the advised object.

59. What is a Proxy?

A proxy is an object that is created after applying advice to a target object. When you think of client objects the target object and the proxy object are the same.

60. What are the different types of AutoProxying?

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator

- Metadata autoproxying

61. What is Weaving? What are the different points where weaving can be applied?

Weaving is the process of linking aspects with other application types or objects to create an advised object.
Weaving can be done at compile time, at load time, or at runtime.

62. Explain XML Schema-based aspect implementation?

In this implementation case, aspects are implemented using regular classes along with XML based configuration.

63. Explain annotation-based (@AspectJ based) aspect implementation

This implementation case (`@AspectJ` based implementation) refers to a style of declaring aspects as regular Java classes annotated with Java 5 annotations.

Spring Model View Controller (MVC)

64. What is Spring MVC framework?

Spring comes with a full-featured MVC framework for building web applications. Although Spring can easily be integrated with other MVC frameworks, such as Struts, Spring's MVC framework uses IoC to provide a clean separation of controller logic from business objects. It also allows to declaratively bind request parameters to business objects.

65. DispatcherServlet

The Spring Web MVC framework is designed around a `DispatcherServlet` that handles all the HTTP requests and responses.

66. WebApplicationContext

The `WebApplicationContext` is an extension of the plain `ApplicationContext` that has some extra features necessary for web applications. It differs from a normal `ApplicationContext` in that it is capable of resolving themes, and that it knows which servlet it is associated with.

67. What is Controller in Spring MVC framework?

Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

68. @Controller annotation

The `@Controller` annotation indicates that a particular class serves the role of a controller. Spring does not require you to extend any controller base class or reference the Servlet API.

69. @RequestMapping annotation

`@RequestMapping` annotation is used to map a URL to either an entire class or a particular handler method.