**Core Spring Interview Questions**

**Question 1: What is IOC or inversion of control?**([answer](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html))

Answer: This *Spring interview question* is the first step towards the Spring framework and many interviewers start Spring interview from this question. As the name implies **Inversion of the control** means now we have inverted the control of creating the object from our own using new operator to container or framework.  
  
Now it’s the responsibility of the container to create an object as required. We maintain one XML file where we configure our components, services, all the classes, and their property. We just need to mention which service is needed by which component and container will create the object for us.  
  
This concept is known as [dependency injection](http://javarevisited.blogspot.com/2013/06/spring-helloworld-example-in-java-using-3.0-dependency-injection.html) because all object dependency (resources) is injected into it by the framework.

Example:

<bean id="createNewStock"

class="springexample.stockMarket.CreateNewStockAccont">

<property name="newBid"/>

</bean>

In this example, CreateNewStockAccont class contain getter and setter for newBid and container will instantiate newBid and set the value automatically when it is used. This whole process is also called wiring in Spring and by using [annotations](http://www.java67.com/2018/11/top-10-spring-framework-annotations-for-java-developers.html) it can be done automatically by Spring, referred to as auto-wiring of bean in Spring.

**Question 2: Explain the Spring Bean-LifeCycle.**

Ans: Spring framework is based on IOC so we call it as IOC container also So Spring beans reside inside the IOC container. Spring beans are nothing but Plain old java object (POJO).

Following steps explain their life cycle inside the container.

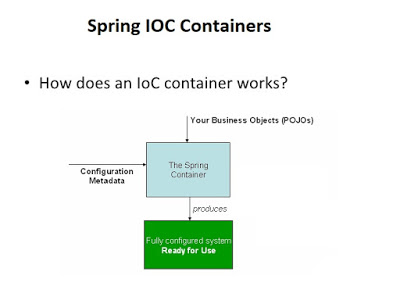
1. The container will look the bean definition inside the configuration file (e.g. bean.xml).

2 using reflection container will create the object and if any property is defined inside the bean definition then it will also be set.

3. If the bean implements the BeanNameAware interface, the factory calls setBeanName() passing the bean’s ID.  
  
4. If the bean implements the BeanFactoryAware interface, the factory calls setBeanFactory(), passing an instance of itself.  
  
5. If there are any BeanPostProcessors associated with the bean, their post- ProcessBeforeInitialization() methods will be called before the properties for the Bean are set.

6. If an init() method is specified for the bean, it will be called.  
7. If the Bean class implements the DisposableBean interface, then the destroy() method will be called when the Application no longer needs the bean reference.

8. If the Bean definition in the Configuration file contains a 'destroy-method' attribute, then the corresponding method definition in the Bean class will be called.  
  
These were just some of the Spring Fundamentals I can cover here if you are interested to learn more I suggest you take a look at [**Spring Master Class - Beginner to Expert**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F), an end-to-end course to learn Spring.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F)

**Question 3: What is Bean Factory, have you used XMLBeanFactory?**

Ans: BeanFactory is factory Pattern which is based on IOC [design principles](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html).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.XmlBeanFactory is used to create bean instance defined in our XML file.**

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

Or

ClassPathResource resorce = **new** ClassPathResource("beans.xml");

XmlBeanFactory factory = **new** XmlBeanFactory(resorce);

**Question 4: What are the difference between BeanFactory and ApplicationContext in Spring?**([answer](http://javarevisited.blogspot.com/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html))

Answer: This one is a very popular Spring interview question and often asks in an entry-level interview. ApplicationContext is the preferred way of using spring because of the functionality provided by it and the interviewer wanted to check whether you are familiar with it or not.

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

**Question 5: What is AOP?**

Answer: The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. AOP is a programming technique that allows a developer to modularize crosscutting concerns, that cuts across the typical divisions of responsibility, such as **logging and transaction management.**  
  
Spring AOP, aspects are implemented using regular classes or regular classes annotated with the @Aspect annotation. You can also check out these [**30+ Spring MVC interview questions**](http://java67.com/2012/08/spring-interview-questions-answers.html) for more focus on Java web development using the Spring MVC framework.

**Question 6: Explain Advice?**

Answer: It’s an implementation of aspect; advice is inserted into an application at join points. Different types of advice include “around,” “before” and “after” advice

**Question 7: What are the joint Point and point cut?**

Ans: This is not really a spring interview questions I would say an AOP one.  Similar to [Object-oriented programming](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html), AOP is another popular programming concept which complements OOPS. A join point is an opportunity within the code for which we can apply an aspect. In Spring AOP, a join point always represents a method execution.

**Pointcut**: a predicate that matches join points. A pointcut is something that defines what join-points advice should be applied.  
  
Here are few more Spring fundamental interview questions for practice  
  
  
  
**Question 8: Difference between the setter and constructor injection in Spring?** ([answer](http://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html))  
Setter injection is more flexible than constructor injection because you must remember the type and order of constructor parameter. Also, constructor injection is generally used to inject the mandatory dependency, while setter can be used to inject the optional dependency.

**Question 9: Difference between Factory Pattern and Dependency Injection in Java?**([answer](http://javarevisited.blogspot.com/2015/06/difference-between-dependency-injection.html))

Even though both allow you to reduce coupling in code, dependency injection is much more flexible and easier to test than Factory pattern.

**Questions 10. Difference between @Autowired and @ Inject annotation in Spring?** ([answer](https://javarevisited.blogspot.com/2017/04/difference-between-autowired-and-inject-annotation-in-spring-framework.html#axzz5nFQiRhkv))

**Question 11: What are the different modules in spring?**

Answer: spring has seven core modules

1.      The Core container module

2.      Application context module

3.      AOP module (Aspect Oriented Programming)

4.      JDBC abstraction and DAO module

5.      O/R mapping integration module (Object/Relational)

6.      Web module

7.      MVC framework module

**Spring MVC Interview Questions Answers**

**Questions 12: What is the difference between @Controller and @RestController in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
Even though both are used to indicate that a Spring bean is a Controller in Spring MVC setup, @RestController is better when you are developing [RESTful web services](https://javarevisited.blogspot.com/2018/02/top-5-restful-web-services-with-spring-courses-for-experienced-java-programmers.html" \t "_blank) using Spring MVC framework. It's a combination of @Controller + @ResponseBody annotation which allows the controller to directly write the response and bypassing the view resolution process, which is not required for RESTful web service.  
  
It also instructs DispatcherServlet to use different HttpMessageConverters to represent the response in the format client is expecting e.g. HttpMessageJackson2Convert to represent response in JSON format and JAXB based message converts to generate XML response.  
  
You can further see  **[REST with Spring](http://www.baeldung.com/rest-with-spring-course?utm_source=javarevisited&utm_medium=web&utm_campaign=rws&affcode=22136_bkwjs9xa" \t "_blank)** course by Baeldung to learn more about developing RESTful Web Services using Spring 4 and Spring 5.

[](http://www.baeldung.com/rest-with-spring-course?utm_source=javarevisited&utm_medium=web&utm_campaign=rws&affcode=22136_bkwjs9xa)

**Question 13: What is the difference between a singleton and prototype bean?**

Ans: This is another popular *spring interview questions* and an important concept to understand. Basically, a bean has scopes which define their existence on the application.

**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.

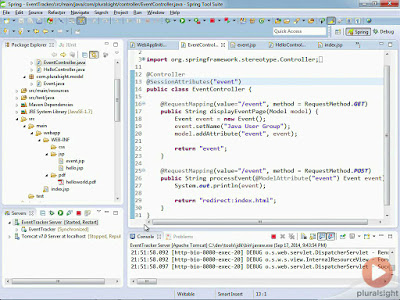
<bean id="createNewStock" class="springexample.stockMarket.CreateNewStockAccont"

singleton=”false”>

<property name="newBid"/>

</bean>

**Question 14: What is the role of DispatcherServlet in Spring MVC?**([answer](http://www.java67.com/2017/06/what-is-use-of-dispatcherservlet-in-spring-mvc.html))  
The DispatcherServlet is very important from Spring MVC perspective, it acts as a FrontController i.e. all requests pass through it. It is responsible for routing the request to the controller and view resolution before sending the response to the client.  
  
When Controller returns a Model or View object, it consults all the view resolvers registered to find the correct type of ViewResolver which can render the response for clients.  
  
In case of RESTful Web Services, the DispatcherServlet is also responsible for using HttpMessageConverter to represent the response in the JSON, XML, or TEXT format, depending on the content negotiation between Client and Server like if client sends request with HTTP accept header as "application/json" then DispatcherServlet will ask the HttpMessageJackson2Converter to convert the response into JSON format.  
  
You can further see the free [**Introduction to Spring MVC**](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspringmvc-intro) course from Pluralsight to learn more about Spring MVC and DispatcherServlet.

[](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspringmvc-intro)

**Question 15: How to call the stored procedure from Java using Spring Framework?** ([answer](http://javarevisited.blogspot.com/2013/04/spring-framework-tutorial-call-stored-procedures-from-java.html))  
Spring Framework provides excellent support to call stored procedures from Java application. In fact, there are multiple ways to call stored procedures in Spring Framework, e.g. you can use one of the query() method from JdbcTemplate to call stored procedures, or you can extend [abstract class](http://javarevisited.blogspot.com/2010/10/abstraction-in-java.html) StoredProcedure to call stored procedures from Java. In this Java Spring tutorial, we will see the second approach to call stored procedure. It's more [object-oriented](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html), but at the same time requires more coding. StoredProcedure class allows you to declare IN and OUT parameters and call stored procedure using it's various execute() method, which has protected access and can only be called from a subclass.  
  
  
I personally prefer to implement StoredProcedure class as [Inner class](http://javarevisited.blogspot.sg/2012/12/inner-class-and-nested-static-class-in-java-difference.html), if it's tied up with one of [DAO Object](http://javarevisited.blogspot.com/2013/01/data-access-object-dao-design-pattern-java-tutorial-example.html), e.g. in this case it nicely fits inside EmployeeDAO. Then you can provide a convenient method to wrap stored procedure calls.   
  
In order to demonstrate, how to call stored procedures from spring-based application, we will first create a simple stored proc using MySQL database, as shown below.

By the way, if you are new to Spring framework then I also suggest you join a comprehensive and up-to-date course to learn Spring in depth. If you need recommendations, I highly suggest you take a look at [**Spring Framework 5: Beginner to Guru**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-framework-5-beginner-to-guru%2F), one of the comprehensive and hands-on course to learn modern Spring.  
  
It' also most up-to-date and covers Spring 5. It's also very affordable and you can buy in just $10 on Udemy sales which happen every now and then.   
  
**Question 16: How to Setup JDBC Database connection pool in Spring Web application?** ([answer](http://javarevisited.blogspot.com/2012/06/jdbc-database-connection-pool-in-spring.html))

**Spring framework** provides a convenient JdbcTemplate class for performing all Database related operations. if you are not using Hibernate than using Spring's JdbcTemplate is a good option. JdbcTemplate requires a DataSource which is javax.sql.DataSource implementation and you can get this directly using [spring bean](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html) configuration or by using **JNDI** if you are using the [J2EE web server or application server](http://javarevisited.blogspot.sg/2012/05/5-difference-between-application-server.html) for managing Connection Pool.   
  
See *How to setup JDBC connection Pool in tomcat and Spring* for JNDI based connection pooling for more details. In order to the setup Data source you will require the following configuration in your applicationContext.xml (spring configuration) file:

//Datasource connection settings in Spring  
**<bean** id="springDataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close" **>**  
   **<property** name="url" value="jdbc:oracle:thin:@localhost:1521:SPRING\_TEST" **/>**  
   **<property** name="driverClassName" value="oracle.jdbc.driver.OracleDriver" **/>**  
   **<property** name="username" value="root" **/>**  
   **<property** name="password" value="root" **/>**  
   **<property** name="removeAbandoned" value="true"**/>**  
   **<property** name="initialSize" value="20" **/>**  
   **<property** name="maxActive" value="30" **/>**  
**</bean>**  
  
//Dao class configuration in spring  
 **<bean** id="EmployeeDatabaseBean" class="com.test.EmployeeDAOImpl"**>**  
    **<property** name="dataSource" ref="springDataSource"**/>**  
 **</bean>**

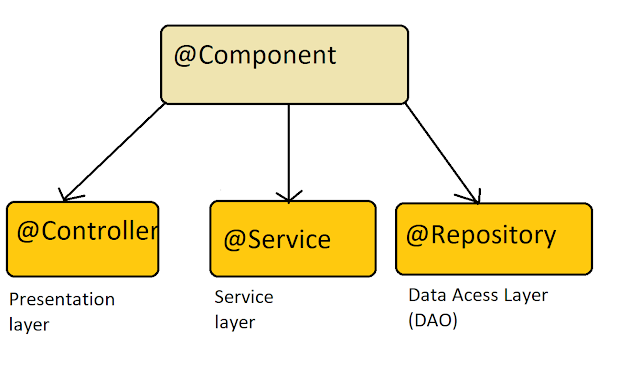
Below configuration of DBCP connection pool will create 20 database connection as initialSize is 20 and goes up to 30 Database connection if required as maxActive is 30. you can customize your database connection pool by using different properties provided by Apache DBCP library.   
  
The above example is creating a connection pool with Oracle 11g database and we are using oracle.jdbc.driver.OracleDriver comes along with **ojdbc6.jar or ojdbc6\_g.jar,** to learn more about [how to connect Oracle database from Java program](http://javarevisited.blogspot.sg/2012/04/java-program-to-connect-oracle-database.html) see the link.

**Questions 17: Difference between @ReqeustParam and @PathVariable in Spring MVC?** ([answer](https://javarevisited.blogspot.com/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html))  
The Spring MVC framework, one of the most popular frameworks for developing a web application in Java world also provides several useful annotations to extract data from the incoming request and mapping the request to the controller, like @RequestMapping, @RequestParam, and @PathVariable. Even though both @RequestParam and @PathVariable is used to extract values from the HTTP request, there is a subtle difference between them, which makes them a useful question from an interview and [spring certification](http://javarevisited.blogspot.sg/2017/06/2-books-to-prepare-for-spring-certification-exam.html#axzz4pbqSY8Ua) point of view. We'll examine the subtle difference between @RequestParam and @PathVaraible in this article.  
  
  
As the name suggests, @RequestParam is used to get the request parameters from URL, also known as query parameters, while @PathVariable extracts values from URI.  
  
For example, if the incoming HTTP request to retrieve a book on topic "Java" is http://localhost:8080/shop/order/1001/receipts?date=12-05-2017, then you can use the @RequestParam annotation to retrieve the query parameter date and you can use @PathVariable to extract the orderId i.e. "1001" as shown below:  
  
@RequestMapping(value="/order/{orderId}/receipts", method = RequestMethod.GET)  
public List listUsersInvoices(                               @PathVariable("orderId") int order,  
 @RequestParam(value = "date", required = false) Date dateOrNull) {  
...  
}  
  
The required=false denotes that the query parameter can be optional, but the [URL](http://www.java67.com/2013/01/difference-between-url-uri-and-urn.html) must have the same URI.  
  
By the way, if you are new to Spring framework then I also suggest you join a comprehensive and up-to-date course to learn Spring in depth. If you need recommendations, I highly suggest you take a look at [**Spring Framework 5: Beginner to Guru**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-framework-5-beginner-to-guru%2F), one of the comprehensive and hands-on course to learn modern Spring. It' also most up-to-date and covers Spring 5.  
  
Read more: <https://javarevisited.blogspot.com/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html#ixzz6QdvYrvAD>

**Questions 18: Difference between @Component, @Service, @Controller, and @Repositoring annotation in Spring MVC?** ([answer](https://javarevisited.blogspot.com/2017/11/difference-between-component-service.html))

Before you learn the difference between @Component, @Service, @Controller, and @Repository annotations in the [Spring framework](https://javarevisited.blogspot.com/2018/06/top-6-spring-framework-online-courses-Java-programmers.html), it's important to understand the role of @Component annotation in Spring. During the initial release of Spring, all beans are used to be declared in an XML file. For a large project, this quickly becomes a massive task, and Spring guys recognize the problem rather quickly. In later versions, they provide annotation-based dependency injection and Java-based configuration. From Spring 2.5 annotation-based [dependency injection](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) was introduced, which automatically scans and registers classes as Spring bean which is annotated using @Component annotation.  
  
This means you don't declare that bean using the <bean> tag and inject the dependency, it will be done automatically by Spring. This functionality was enabled and disabled using <context:component-scan> tag.  
  
Now that you know what does @Component annotation does let's see what does @Service, @Controller, and @Repository annotation do.  
  
They are nothing but the specialized form of @Component annotation for certain situations. Instead of using @Component on a controller class in Spring MVC, we use @Controller, which is more readable and appropriate.  
  
By using that annotation we do two things, first, we declare that this class is a Spring bean and should be created and maintained by [Spring ApplicationContext](http://javarevisited.blogspot.sg/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html), but also we indicate that its a controller in MVC setup. This latter property is used by web-specific tools and functionalities.

For example, DispatcherServlet will look for @RequestMapping on classes that are annotated using @Controller but not with @Component.  
  
This means @Component and @Controller are the same with respect to bean creation and dependency injection but later is a specialized form of former. Even if you replace @Controller annotation with @Compoenent, Spring can automatically detect and register the controller class but it may not work as you expect with respect to request mapping. You can further see, [**Spring Master Class**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F) course on Udemy for more details about these annotations.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F)

The same is true for @Service and @Repository annotation, they are a specialization of @Component in service and persistence layer. A Spring bean in the service layer should be annotated using @Service instead of @Component annotation and a spring bean in the persistence layer should be annotated with @Repository annotation.  
  
By using a specialized annotation we hit two birds with one stone. First, they are treated as Spring bean, and second, you can put special behavior required by that layer.  
  
For example, @Repository's not only helping in annotation based configure but also catch Platform-specific exceptions and re-throw them as one of Spring’s unified [unchecked exception](https://javarevisited.blogspot.sg/2011/12/checked-vs-unchecked-exception-in-java.html).  
  
Though for that you also need to declare org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor as Spring bean in your application context.  
  
This bean post-processor adds an advisor to any bean that’s annotated with @Repository so that any platform-specific exceptions are caught and then rethrown as one of Spring’s unchecked data access exceptions. You can also see [**Spring Framework 5: Beginner to Guru**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-framework-5-beginner-to-guru%2F) on Udemy for more details.  
  
This is also one of the frequently asked Spring Interview Question and a popular concept from the Spring certification perspective. You will find a couple of questions based on these annotations and their usage in the Spring professional certification exam too.  
  
Read more: <https://javarevisited.blogspot.com/2017/11/difference-between-component-service.html#ixzz6QdwCVxre>

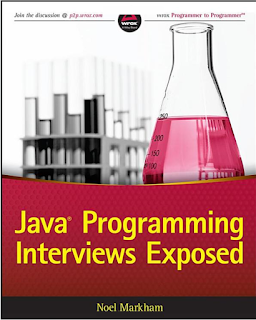
**Question 19: What type of transaction Management Spring support?**

Ans: This spring interview question is a 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 types of transaction management is supported by spring:

1. Programmatic transaction management

2. Declarative transaction management.

If you need more questions, you can also check out the [**Java Programming Interview exposed**](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20) book from Wrox publication, apart from various Java topics it also contains some really good Spring framework, Hibernate, and Spring MVC questions with detailed explanation.

[](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20)

Also, Hibernate is mostly used in Spring, so don't forget to prepare some [Hibernate interview questions](http://javarevisited.blogspot.sg/2013/05/10-hibernate-interview-questions-answers-java-j2ee-senior.html) along with Spring.

**Spring Security Interview Questions Answer**

Some of the readers requested to provide Spring Security interview questions and answer, So I thought to update this article with a few of Spring security question I came across.  
  
Here are those:  
  
  
**20. How do you control the concurrent Active session using Spring Security? (**[**answer**](http://javarevisited.blogspot.sg/2012/03/spring-security-example-tutorial-how-to.html)**)**  
Another Spring interview question which is based on Out of box feature provided by Spring framework. You can easily control How many active session a user can have with a Java application by using Spring Security.  
  
Apart from that Spring Security also provides the "remember me" feature which you can use to provide easier access for your users by remembering their login details on their personal computer. You can further see [**Learn Spring Security**](http://www.baeldung.com/learn-spring-security-course?utm_source=javarevisited&utm_medium=web&utm_campaign=lss&affcode=22136_bkwjs9xa) course by Eugen Paraschiv to learn more about advanced details of Spring Security.

**21. How do you set up LDAP Authentication using Spring Security? (**[**answer**](http://javarevisited.blogspot.sg/2011/11/ldap-authentication-active-directory.html)**)**  
This is a very popular Spring Security interview question as Spring provides out of the box support to connect Windows Active Directory for LDAP authentication and with few configurations in Spring config file you can have this feature enabled.  
  
  
**22.  How to implement Role Based Access Control (RBAC) using Spring Security?**([answer](http://javarevisited.blogspot.com/2013/07/role-based-access-control-using-spring-security-ldap-authorities-mapping-mvc.html))  
Spring Security provides a couple of ways to implement Role based access control like by using GrantedAuthority. See the article to learn more about it.