http://maven.springframework.org/release/org/springframework/spring

<http://www.springbyexample.org/>

<http://www.springbyexample.org/examples/>

<https://github.com/spring-by-example/spring-by-example>

<http://www.tutorialspoint.com/spring/spring_transaction_management.htm>

<https://github.com/spring-projects/spring-petclinic/blob/master/src/main/java/org/springframework/samples/petclinic/repository/jdbc/JdbcPetRepositoryImpl.java>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.sg/2013/03/5-good-books-to-learn-spring-framework-mvc-java-programmer.html>

5 books to learn Spring framework and Spring MVC for Java Programmers

Spring and Spring MVC is one of the most popular Java framework and most of new Java projects uses Spring these days. Java programmer often ask questions like *which books is good to learn Spring MVC* or *What is the best book to learn Spring framework* etc. Actually, there are many books to learn Spring and Spring MVC but only certain books can be considered good because of  there content, examples or the way they explained concept involved in Spring framework. Similar to  [Top 5 books on Java programming](http://javarevisited.blogspot.com/2013/01/top-5-java-programming-books-best-good.html) we will some good books on Spring in this article, which not only help beginners to start with Spring but also teaches some best practices. In order to learn a new technology or a new framework, probably best way is to start looking documentation provided and Spring Framework is no short on this. Spring  provides great, detailed documentation to use various features of Spring framework but despite of that nothing can replace a good book. Luckily both Spring and Spring MVC got couple of good titles which not only explains concepts like [Dependency Injection and Inversion of Control](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) which is core to spring framework but also gives coverage to other important aspect of Spring. Following are some of the good books available on Spring and Spring MVC which can help you to learn Spring.

### Top 5 Books on Spring Framework and Spring MVC

Here is my list of top 5 books to learn Spring MVC and Spring framework. Let me know if you come across any other great book on Spring, which is worth adding into this list.

[**ExpertSpring MVC and Web Flow**](http://www.amazon.com/Expert-Spring-Flow-Experts-Voice/dp/159059584X?tag=javamysqlanta-20)

Expert Spring MVC and Web Flow by Seth Ladd, Darren Davison, Steven Devijver, Colin Yates is one of my favorite book on Spring MVC and arguably one of the best book in Spring MVC. It covers both Spring MVC and web flow in depth and explains each concept with simple explanation. I highly recommend this book to any beginner which is learning Spring MVC framework. There chapter on Spring fundamentals is also one of the best way to learn [dependency injection and inversion of control in Spring](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) and I myself learned DI and IOC from that chapter. This is the Spring book I recommend to any Java web developer who is familiar with Java web technology or any MVC framework like Struts. Only missing point is that this book only covers Spring MVC and web flow and does not cover whole Spring framework. Also, in my opinion there chapter on Spring Fundamentals is one of the best way to start with Spring framework.

[**SpringRecipes – A problem solution approach**](http://www.amazon.com/Spring-Recipes-Problem-Solution-Approach-Experts/dp/1430224991?tag=javamysqlanta-20)

This is another good book on Spring Framework which I like most. This book is collection of Spring recipes or How to do in Spring Framework. In every Spring recipes you learn some new concept and it also helps to learn Spring fundamental e.g. there recipes help me to learn [when to use ApplicationContext and BeanFactory](http://javarevisited.blogspot.ca/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html) and  [Constructor vs Setter Injection](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html). Key highlight of this book is, It’s **problem solution approach**. Since it’s teaching style is different than any conventional book, it’s a good supplement along with Spring documentation. This books also provide excellent coverage of many spring technologies e.g. [Spring Security](http://javarevisited.blogspot.sg/2011/11/ldap-authentication-active-directory.html), Spring JDBC, Spring and [EJB](http://javarevisited.blogspot.com/2012/03/top-10-ejb-interview-question-and.html), JMX, Email and have a chapter on scripting as well. If you like books on problem solution approach than you will enjoy reading Spring Recipes, not the best book on Spring but still a good one and will definitely made to any list of *top 10 books on Spring framework*.

[**ProfessionalJava Development with the Spring Framework**](http://www.amazon.com/Professional-Java-Development-Spring-Framework/dp/0764574833?tag=javamysqlanta-20)

Main highlight of this book is that one of it’s author is [Rod Johnson](http://en.wikipedia.org/wiki/Rod_Johnson_%28programmer%29), who is also created Spring framework. So you get his view on Spring and How spring should be used used, what are best practices to follow on Spring e.g. [When to use Setter Injection and Constructor Injection](http://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html). This book provide good coverage of Spring framework including Spring core, Spring MVC, Spring ORM support etc. Also examples in this book is easy to understand and it also focus on [Unit tests](http://javarevisited.blogspot.com/2012/08/best-practices-to-write-junit-test.html) which is good practice. Though I don’t rate this book too high, like if your focus is Spring MVC than Expert Spring MVC and Web flow is the best Spring book to follow. If you are looking an overview on Spring features, than Spring Documentation is best book to read. As I said positive point of this book is knowing Spring from author Rod Johnson himself. Once you have basic knowledge of Spring framework, you can read this book to get authors view.

**Pro Spring 3.0**

Pro Spring is one of the best book to learn Spring Framework from start. This book is massive and tries to cover most of the Spring concept e.g. Spring fundamentals, [JDBC Support](http://javarevisited.blogspot.com/2012/06/jdbc-database-connection-pool-in-spring.html), Transaction support, Spring AOP, Spring Web MVC, Spring Testing etc. Good point about this book is that it’s conventional and easy to read, it explains concept, followed with good example, which is good way to learn. What is worrying is sheer size, I haven’t completed this book till date and only refer with some topic. Good point is that this book covers **Spring 3.1** which is the latest stable version. As I said this is one of the *most comprehensive book on Spring framework* and any one who wants to learn Spring framework by following just one book, Pro Spring 3.0 is a good choice.

**Spring Documentation**

Spring framework documentation is located on Springsource website, here is the link for Spring documentation for Spring framework 3.1 in HTML format [http://static.springsource.org/spring/docs/3.1.0.M2/spring-framework-reference/html/.](http://static.springsource.org/spring/docs/3.1.0.M2/spring-framework-reference/html/) Though this is not a book, Spring tutorials and Spring documentation are another two source of learning Spring framework, which I highly recommend. Main reason for that is they are free and highly comprehensive and has lot of examples to support various concept and feature. Also one of the **best part of reference documentation is that they are updated with the latest Spring release available**. Updating books with every new version of Spring is rather difficult than updating documentation. Spring documentation combine with any Spring book is best way to learn Spring framework. For learning Spring MVC, you can combine Spring documentation with earlier spring book, **Expert Spring MVC and Web Flow**.

### Spring in Action

Lots of my readers suggested  Spring in Action from manning, as one of the best book to learn Spring. Seems like a worth reading book. I have seen it's content briefly and it does cover both Spring and Spring MVC. So if you are looking for common book for complete Spring framework, Spring in Action is another one.

These are some of the best books to learn Spring framework and Spring MVC. Spring documentation is special  because of update and new releases of Spring Framework. Given popularity of Spring Framework for new Java development work, every Java developer should make effort to learn Spring framework.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://java67.blogspot.sg/2012/08/spring-interview-questions-answers.html>

Spring MVC Framework Interview Questions Answers

**Spring Interview Questions Answers**  
Spring Interview Question is one of the first thing Java programmer should prepare before appearing on any Java or Spring interview. With growing popularity of Spring framework in Java world, [Interview questions on Spring](http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html) are getting more important day by day. For those who don't know, Spring framework is one of the most popular Java programming framework, which not only provided dependency Injection and Inversion of Control but also provides useful API to Java programmers. Spring framework is divided into many different modules e.g. Spring MVC, Spring Integration, Spring Batch, Spring LDAP, Spring Security and several other modules. Based upon your work experience, Interviewer can asked questions from core Spring or these modules. That's why, while preparing Spring interview question focus on Spring core, [Spring Security](http://javarevisited.blogspot.sg/2011/11/ldap-authentication-active-directory.html) and Spring API, these are main areas from where interviewer ask questions. If you have working experience in Spring based Java projects, you can easily answer most of these Spring questions asked in [Java interview](http://javarevisited.blogspot.sg/2011/04/top-20-core-java-interview-questions.html), but if you don't have any prior experience and just learning Spring, you need to not only prepare answers of these Spring questions but also some follow-up questions, which may stem from your answers.

## Spring Questions and Answers

[](http://4.bp.blogspot.com/-Uv_sJCDUbRw/UCOoHFznRsI/AAAAAAAAAck/prQO446sWAg/s1600/spring_thumbnail.PNG)Anyway let's see frequently asked questions on Spring framework on Java interviews :  
 **1) What is spring framework? Why Java programmer should use Spring framework**  
Very common Spring interview question, Spring is a framework which helps Java programmer in development. Spring provides dependency Injection and IOC container, Spring MVC flow and several useful API for Java programmer.  
 **2) What is default scope of bean in Spring framework ?**  
default scope of bean is [Singleton](http://javarevisited.blogspot.sg/2011/03/10-interview-questions-on-singleton.html), you can read this article which explains about all possible scope of a spring bean : [What is bean scope in Spring](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html)  
  
**3) Does Spring singleton beans are thread-safe ?**  
No, Spring singleton beans are not thread-safe. Singleton doesn't mean bean would be [thread-safe](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html).  
  
**4) What is dependency Injection?**  
Dependency Injection is one of the design pattern, which allows to inject dependency on Object, instead of object resolving dependency.  
  
5) What is Inversion of Control concept, how does Spring support IOC?  
  
6) What is Spring MVC ? Can you explain How one request is processed ?  
  
7) How to you create controller in Spring ?  
  
  
**8) What is view Resolver pattern ? how it work in Spring MVC**  
View Resolver patter 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 view, which is than passed to View Resolver for selecting appropriate view.Spring MVC framework supplies inbuilt view resolver for selecting views.  
  
  
**9) What is Spring Security ?**  
Spring security is a project under spring framework umbrella, which provides support for security requirements of enterprise Java projects. Spring Security formerly known as aegis security provides out of box support for creating login screen, remember me cookie support, securing URL, authentication provider to authenticate user from database, LDAP and in memory, concurrent active session management support and many more. In order to use Spring security in a Spring MVC based project, you need to include spring-security.jar and configure it in application-Context-security.xml file, you can name it whatever you want, but make sure to supply this to ContextLoaderListener, which is responsible for creating Spring context and initializing dispatcher servlet.  
  
**10) How do you control concurrent Session on Java web application using Spring Security.**  
You can use Spring Security to control number of active session in Java web application. Spring security framework provides this feature out of box and when enabled , a user can only have one active session at a time. See this Spring Security example to learn more about [How to control concurrent user session using Spring security](http://javarevisited.blogspot.sg/2012/03/spring-security-example-tutorial-how-to.html)  
  
 **11) What types of dependency injection is supported by Spring Framework? When do you use Setter and Constructor Injection, pros and cons?**  
There are 2 types of dependency injection supported by Spring, constructor based injection and setter based injection. Both types have their own advantages and disadvantages, you should use Constructor injection when object's dependencies are not optional and they must be initialized with their dependencies. Also use constructor injection if order of initialization or dependency matters because in Setter based injection you cannot impose any order. Use setter injection when dependencies are optional. See [difference between setter and constructor injection in Spring](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html) for more detailed answer.  
  
  
12) What is difference between ApplicationContext and BeanFactory in Spring framework?  
  
13) How do you call stored procedure by using Spring framework?  
  
14) What does JdbcTemplate and JmsTemplate class offer in Spring?  
  
15) Can we use more than one configuration file for our Spring project?  
  
16) Explain Spring MVC flow with a simple example e.g. starting from Container receives request and forward to your Java application ?  
  
17) What is difference in Spring MVC and Spring core?  
  
18) Can you use Spring MVC framework along with Struts ? I have an existing Java MVC application which is based in Struts, Can I migrate that to use Spring MVC ? How ?  
  
19) What is advantage of Spring MVC framework over Struts 1.0 or Struts 2.0 ? is it worth to convert an existing Struts application to Spring MVC ?  
  
20) How does Spring resolves view returned by ModelAndView class ?  
  
Some Spring MVC questions are tricky e.g. Struts and Spring integration and can be only answered by experienced Java program with 2 to 4 year experience in Spring MVC framework.  
  
**21) IF User checked in CheckBox and got validation error on otherfields and than he unchecked the CheckBox, what would be selection status in command object in Spring MVC ? How do you fix this issue?**  
Since during HTTP post, if checkbox is unchecked than HTTP does include a request parameter for checkbox, which means updated selection won't be picked up. you can use hidden form field, starting with \_ to fix this in Spring MVC. quite tricky question to answer if you are not aware of HTTP POST behavior and Spring MVC.  
  
**22) What are different implementation of View interface you have used in Spring MVC?**  
ULBased View e.g. JSP , JSTLView,  
  
**23) How to escape HTML special characters using Spring MVC?**  
There are some methods in Spring tag library, can't remember now.  
  
These were some of the **Core Spring framework and MVC Interview questions** from my collection, I have given short answers for most of these Spring interview question. I suggest to research more or read along those Spring question to prepare for follow up Spring interview questions.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html

Top 10 Spring Interview Questions Answers J2EE

**Spring framework interview questions** is in rise on J2EE and core Java interviews,  As Spring is the best framework available for Java application development and now *Spring IOC container* and Spring MVC framework are used as de-facto framework for all new Java development. With this popularity interview questions from spring framework is top on any list of  core Java Interview questions. I thought to put together some spring interview questions and answers which has appeared on many Java and J2EE interviews and useful for practicing before appearing on any Java Job interview. This list of Spring interview questions and answers contains questions from Spring fundamentals e.g. Spring IOC and [dependency Injection](http://javarevisited.blogspot.sg/2012/06/20-design-pattern-and-software-design.html), **Spring MVC framework**, Spring Security, Spring AOP etc, because of length of this post I haven't included Spring interview questions from Spring JDBC and JMS which is also a popular topic in core Java and [J2EE interviews](http://javarevisited.blogspot.sg/2011/09/servlet-interview-questions-answers.html). I suggest to prepare those as well. Any way these Spring questions are not very difficult and based on fundamentals e.g. [What is default scope of Spring bean](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html) and mostly asked during first round or telephonic round of Java interview. Although you can find answers of these **Spring interview questions** by doing Google but I have also included some answers for quick reference. As I said Spring  and Spring MVC is fantastic Java framework and if you are not using it than start using it, these questions will give you some head start as well.

## Top Spring Interview Questions and Answers

Now let’s start with questions, these **Spring interview Questions** are not very tricky or tough and based upon primary concepts of spring framework. If you are developing application using Spring framework than you may be, already familiar with many of these Java and Spring interview questions and answers. Nevertheless it’s a good recap before appearing to any Spring and Java interview.  
  
  
**Spring Security Interview questions Answer**   
Some of the reader requested to provide Spring Security interview questions and answer, So i though to update this article with few of Spring security question I came across. Here are those:  
  
**How do you setup LDAP Authentication using Spring Security?**  
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 configuration in Spring config file you can have this feature enable. See [How to perform LDAP authentication in Java using Spring Security](http://javarevisited.blogspot.sg/2011/11/ldap-authentication-active-directory.html) for detailed code explanation and sample.  
  
**How do you control concurrent Active session using Spring Security?**  
Another Spring interview question which is based upon 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. See this spring security example of [how to control concurrent session in Java and Spring](http://javarevisited.blogspot.sg/2012/03/spring-security-example-tutorial-how-to.html) for exact details.

**Question1: What is IOC or inversion of control?**

Answer: This *Spring interview question* is first step towards Spring framework and many interviewer starts Spring interview from this question. As the name implies **Inversion of 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 container to create 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** because all object dependency (resources) is injected into it by 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 annotation it can be done automatically by Spring, refereed as auto-wiring of bean in Spring.

**Question 2: Explain 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 container.

1. Container will look the bean definition inside 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 method destroy() 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.

**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.

XmlBeanFactory is one of the implementation of bean Factory which we have used in our project.

**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 : This one is very popular spring interview question and often asks in entry level interview. ApplicationContext is preferred way of using spring because of functionality provided by it and 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 | Doesn’t support. |
| Support internationalization (I18N) messages | It’s not |
| Support application life-cycle events, and validation. | Doesn’t support. |
| Support  many enterprise services such JNDI access, EJB integration, remoting | Doesn’t support. |

**Question 5: What are different modules in spring?**

Answer : spring have 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

**Question 6: What is difference between singleton and prototype bean?**

Ans: This is another popular *spring interview questions* and an important concept to understand. Basically a bean has scopes which defines 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 7: 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.  Two type of transaction management is supported by spring

1. Programmatic transaction management

2. Declarative transaction management.

**Question 8: 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 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

**Question 9: 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 10: What is 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. Join point is an opportunity within 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 point cut is something that defines at what join-points an advice should be applied

These spring interview Questions and answers are not very difficult and focused on spring fundamentals rather than focusing on advanced feature of session management, spring security, authentication etc. we will cover of those question on some other interview article. I would also suggest that share some spring questions asked to you guys during interview and than I can put together those with there answers for quick reference of everybody.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html>

Inversion of Control and Dependency Injection design pattern with real world Example - Spring tutorial

Inversion of Control and Dependency Injection is a core design pattern of Spring framework. IOC and DI design pattern is also a popular [design pattern interview question in Java](http://javarevisited.blogspot.sg/2012/06/20-design-pattern-and-software-design.html). As name suggest Inversion of control pattern Inverts responsibility of managing life cycle of object e.g. creating object, setting there dependency etc from application to framework, which makes writing Java application even more easy. Programmer often confused between IOC and DI, well both words used interchangeably in Java world but **Inversion of Control** is more general concept and **Dependency Injection** is a concrete design pattern. Spring framework provides two implementation of IOC container in form of [Application Context and BeanFactory](http://javarevisited.blogspot.ca/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html) which manages life-cycle of bean used by Java application. As you may know necessity is mother of invention, it benefit to first understand problem solved by IOC and Dependency Injection design pattern. This makes your understanding more clear and concrete. We have touched basics of Dependency Injection and Inversion of control in our article [10 OOPS and SOLID design principles for Java programmer](http://javarevisited.blogspot.de/2012/03/10-object-oriented-design-principles.html) and this Java article tries to explain it by taking a real life example of Service based architecture popular in enterprise Java development. In this Spring or design pattern tutorial we will first see normal implementation of AutditService class, a class in this example which provides auditing in enterprise Java application and than use of dependency Injection. This will allow us to find out problems and how they are solved by *Dependency injection design pattern*. . Also there are multiple way to inject dependency in spring e.g. Setter Injection or Constructor Injection, which uses setter method and constructor for injecting dependency, see [Setter injection vs Constructor injection](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html)  to find out when to use them.

## Inversion of Control and Dependency Injection design pattern

Any way let’s back to core *concept of Inversion of Control and dependency Injection* design pattern. Look at below implementation of an AuditService whose job is to store every audit messages into database. This is one of the simplest kind of auditing Service required in Enterprise Java application.

/\*\*  
 \* Java Service class which provides auditing functionality by storing  
 \* auditing message into persistent.  
 \*/   
**public** **class** AuditServiceImpl **implements** AuditService{  
  
    **private** AuditDAO auditDao = **new** AuditDAO();  
        
    @**Override**  
    **public** **boolean** audit (**String** message) {  
       **return** auditDao.store(message);  
    }  
    
}

In first glance this implementation looks perfect but there are three major problem with this implementation:

1) Every AuditServiceImpl has its **own copy of AuditDAO** which is an **expensive** object as it wraps a [database connection](http://javarevisited.blogspot.sg/2012/06/jdbc-database-connection-pool-in-spring.html) with in. It make no sense to create separate instances of AuditDAO, if you can share one between multiple AuditService.

2) AuditServiceImpl is **closely coupled with AuditDAO** as its creating instance of AuditDAO using new() operator. If you change the [constructor](http://javarevisited.blogspot.sg/2012/12/what-is-constructor-in-java-example-chainning-overloading.html) of AuditDAO this code will be broken. Because of this coupling its difficult to replace AuditDAO with better implementation.

3) There is **no easy way to test audit()** method which is **dependent on auditDAO**. Since you can not mock AuditDAO you have to rely on actual implementation and if AuditDAO is an environmental dependent object which it is as it connect to different database on different environment, your [Junit test](http://javarevisited.blogspot.sg/2012/08/best-practices-to-write-junit-test.html) case may pass in some environment and may fail in other environment.

## What is Dependency Injection concept:

Dependency Injection is a design pattern on which dependency of object (in this case AuditDAO is a dependency for AuditServiceImpl Object) is injected by framework rather than created by [Object](http://javarevisited.blogspot.ca/2012/12/what-is-object-in-java-or-oops-example.html) itself. Dependency Injection reduces coupling between multiple object as its dynamically injected by framework. One of the implementation of DI is Inversion of Control (IOC) on which framework like Spring controls object’s dependency. There are mainly two types of Dependency Injection: [Constructor Injection and Setter Injection](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html).

In Constructor Injection, dependency of Object is injected using [constructor](http://javarevisited.blogspot.sg/2012/12/what-is-constructor-in-java-example-chainning-overloading.html), while in Setter Injection, Dependency is provided by [setter method](http://javarevisited.blogspot.sg/2012/12/getter-and-setter-method-vs-public-modifier-field-java.html). Both has there pros and cons. Constructor DI allows object to be created in complete state and follows principle of fully functional object while Setter DI allows object to be created without its dependency. which may result in **incomplete object** if dependency is not available. This answers one of the famous [spring interview question](http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html) "when do you use Setter injection and Constructor Injection in Spring". Another benefit of Setter Dependency Injection is readability, since Spring is configured with **xml configuration file** and setter injection is provided with bean property which is much easier to read and understand than constructor injection which doesn't state the property.

**AuditServiceImpl written using Dependency Injection**

Now we will see How Dependency Injection solves all three problems we have listed with above implementation of AuditService. here is a new implementation of AuditService with setter dependency injection.

**public** **class** AuditServiceImpl **implements** AuditService{  
  
    **private** AuditDAO auditDao;  
  
    **public** **void** setAuditDao(AuditDAO AuditDao) {  
        **this**.AuditDao = AuditDao;  
    }  
    
    @**Override**  
    **public** **boolean** audit (**String** message) {  
       **return** auditDao.store(message);  
    }  
    
}

1. Since AuditDAO is injected here its possible to share single AuditDAO (an expensive object) between multiple AuditService.

2. Since AuditServiceImpl is not creating instance of AuditDAO its no more coupled with AuditDAO and work with any implementation of AuditDAO, thanks to another famous object oriented design principle [“program for interface than implementation"](http://javarevisited.blogspot.de/2012/03/10-object-oriented-design-principles.html).

3. Because AuditDAO is injected by DI at runtime its easy to test audit() method by providing a mock AuditDAO class. This not only makes testing easier but also independent of environmental changes as you are not using actual implementation of AuditService.

This was the exact way I learn **Dependency Injection** and **Inversion Of Control design principles**. It always help first to understand problem and than solution to related each other. From above learning we can easily derive *advantages or benefits of Dependency Injection* in Java application:

**1) Reduce coupling**

both constructor and setter dependency injection reduce coupling. like in above example coupling between AuditService and AuditDAO is reduced by using Dependency Injection.

**2) Improves testability**

Dependency Injection allows to replace actual object with mock object which improves testability by writing simple JUnit tests which uses mock object.

**3) Flexibility**

This is another advantage which comes as side benefit of reduced coupling, because of DI you can replace non performance implementation with better one later.

That’s all on **What is Inversion of control and Dependency Injection design pattern**. We have tried to learn this pattern with a real life example and compares a class which is written using principle of IOC and DI and without that. IOC and DI easily bring quality in coding. We have seen clear benefits in terms of reduce coupling, improved testability and Flexibility to change implementation. It’s always good to write code which follows principle of Inversion of Control and dependency Injection and Spring framework by default ensures that.

Other **design pattern and Spring tutorials** from Javarevisited

[Decorator design pattern in Java with real life example](http://javarevisited.blogspot.sg/2011/11/decorator-design-pattern-java-example.html)

[How to implement LDAP authentication in Java using Spring security](http://javarevisited.blogspot.de/2011/11/ldap-authentication-active-directory.html)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html>

What is Bean scope in Spring MVC framework with Example

**Bean scope in Spring framework** or Spring MVC are scope for a bean managed by **Spring IOC container**. As we know that Spring is a framework which is based on **Dependency Injection** and **Inversion of Control** and provides bean management facilities to Java application. In Spring managed environment bean (Java Classes) are created and wired by Spring framework. Spring allows you to define how those beans will be created and scope of bean is one of those details. This Spring tutorial is next on my earlier post on Spring like [how to implement LDAP authentication using Spring security](http://javarevisited.blogspot.com/2011/11/ldap-authentication-active-directory.html) and  [How to limit number of user session in web application](http://javarevisited.blogspot.com/2012/03/spring-security-example-tutorial-how-to.html), if you haven’t read them already you may find them useful.

In spring framework bean declared in ApplicationContext.xml can reside in five scopes:

1) Singleton (default scope)

2) prototype

3) request

4) session

5) global-session

Singleton and prototype are two common bean scope which is available on all Spring Application Context while request, session and global session bean scope are only available on Web aware application Context like WebApplicationContext.

[bean scope in Spring 2.5 and spring 3.0](http://javarevisited.blogspot.com/2011/11/struts-interview-questions-answer-j2ee.html)**Singleton bean scope is default scope for bean** declared in Spring and applicable when you don't specify scope attribute while specifying bean details in ApplicationContext.xml or Spring configuration file. Singleton bean scope is like [Singleton pattern in Java](http://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html) where only one instance of bean is created per Spring container. So no matter how many times you call getBean() method, **same bean instance** will be returned if its bean scope is declared as Singleton. While in case of prototype bean scope, every getBean() call creates a new instance of Spring bean. [Difference between Singleton and prototype bean scope](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html) is also a popular Spring question.

On the other hand **request bean scope** allows each HTTP request to have its own instance of bean created and supplied by Spring framework, while Session bean scope allows Web application to have bean instance per session basis. both of these bean scope are available on WebApplicationContext or any web aware application context. Last one which is **global session bean scope** is only applicable on *portlet* aware bean scope and allows bean instance per global session. In short Singleton vs prototype is an important which clearly segregate one instance to multiple instances of bean.

## How to specify Bean Scope in Spring Framework

In order to specify bean scope you can either use Annotation on Spring or you can define it on Application Context, for example in below Spring configuration file AuditService is configured as Singleton using singleton bean scope and PaymentService as prototype bean scope.

//bean configured on singleton bean scope  
**<bean** id="auditService" class="com.sample.service.impl.AuditServiceImpl"  scope="singleton"**/>**

Since singleton is also default scope in spring framework, following declaration is exactly same and creates bean on singleton scope.

**<bean** id="auditService" class="com.sample.service.impl.AuditServiceImpl" **/>**

Though I prefer explicit declaration to make bean scope loud and clear. Now every time you call getBean("auditService") it will return same instance of AuditService.

AuditService auditService = ApplicationContext.getBean("auditService");

//bean configured on prototype bean scope  
**<bean** id="auditService" class="com.sample.service.impl.AuditServiceImpl"  scope="prototype"**/>**

In case of prototype beans cope every call to getBean("auditServie") will return different instance of AuditServiceImpl class. If you want to use **Annotation to define bean scope** than you can use @Scope("singleton") or @Scope("prototype") on Bean class. you will also need to enable component scanning in Order to let Spring knows about bean scope. which you can do it spring 2.5 as <context:component-scan base-package="com.sample.service.impl" />. Bean scope has not been changed from various spring version and so far two most used spring version spring 2.5 and spring 3.0 has only five bean scope.

Bean Scope in Spring 2.5 and Spring 3.0 is similar, all default scopes are still supported in spring 3.0 with addition of few new scopes like thread scope or [SimpleThreadScope](http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/context/support/SimpleThreadScope.html) which is a scope backed by thread. You can also register your own custom scope using CustomScopeConfigurer utility., there is no new scope for bean is introduced on spring 3.0

That’s about what is bean scope in Spring framework. Since bean creation is managed by Spring IOC container its worth remember that how to specify scope for a particular Bean and what is default scope of Bean which is Singleton to avoid any assumption and code accordingly.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.sg/2012/03/spring-security-example-tutorial-how-to.html>

Spring Security Example Tutorial - How to limit number of User Session in Java J2EE

**Spring security** can limit number of session a user can have. If you are developing web application specially **secure web application** in Java J2EE then you must have come up with requirement similar to online banking portals have e.g. **only one session per user** at a time or **no concurrent session per user**. You can also implement this functionality without using spring security but with Spring security its just piece of cake with coffee :). **Spring Security** provides lots of Out of Box functionality a secure enterprise or web application needed like authentication, authorization, session management, password encoding, secure access, session timeout etc. In our spring security example we have seen [how to do LDAP Authentication in Active directory using spring security](http://javarevisited.blogspot.com/2011/11/ldap-authentication-active-directory.html) and in this spring security example we will see how to limit number of session user can have in Java web application or restricting concurrent user session.

## Spring Security Example: Limit Number of User Session

[spring security example - limit number of session in java J2EE](http://2.bp.blogspot.com/-wrzDeQGAe1I/TWu8pLuLr4I/AAAAAAAAADE/V017G-6Q61w/s1600/java_logo_50_50.jpg)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.

### Dependency

This code has dependency on spring-security framework. You need to download spring security jar like spring-security-web-3.1.0.jar and add into [application classpath](http://javarevisited.blogspot.com/2011/01/how-classpath-work-in-java.html).

This **simple example of spring security** shows power of spring security, a small piece of xml snippet can add very useful and handy **security feature** in your Java web application. I recommend using spring security for your new or existing Java web application created using Servlet JSP.

That’s all on how to limit number of user session using [spring security](http://static.springsource.org/spring-security/site/) in Java web application. Let me know if you face any issue while implementing this security feature in your project.

Other **Java tutorials** you may like

[Top 10 Spring question and answer asked in Interview](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html)

[Top 10 Struts interview question and answer for Beginners and Intermediate Java programmer](http://javarevisited.blogspot.com/2011/11/struts-interview-questions-answer-j2ee.html)

[How to add certificate in keystore in Java – Example Tutorial](http://javarevisited.blogspot.com/2012/03/add-list-certficates-java-keystore.html)

[How to setup error page in JSP – Example Tutorial](http://javarevisited.blogspot.com/2012/01/error-page-in-java-web-application.html)

[Top 10 EJB Interview question and answer](http://javarevisited.blogspot.com/2012/03/top-10-ejb-interview-question-and.html)

[Difference between truncate and delete in SQL](http://javarevisited.blogspot.com/2011/10/how-to-use-truncate-and-delete-command.html)

[How to manage transaction in database](http://javarevisited.blogspot.com/2011/11/database-transaction-tutorial-example.html)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html>

Difference between Setter vs Constructor Injection in Spring

**Spring Setter vs Constructor Injection**

Spring supports two types of dependency Injection, using setter method e.g. setXXX() where XXX is dependency or via constructor argument. First way of dependency injection is known as **setter injection** while later is known as **constructor injection**. Both approaches of Injecting dependency on Spring bean has there pros and cons, which we will see in this Spring framework article. *Difference between Setter Injection and Constructor Injection in Spring* is also a popular [Spring framework interview question](http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html).Some time interviewer also ask as When do you use Setter Injection over Constructor injection in Spring or simply benefits of using setter vs constructor injection in Spring framework. Points discussed in this article not only help you to understand Setter vs Constructor Injection but also Spring's dependency Injection process. By the way if you are new in Spring framework and learning it, you may want to take a look at my list of [5 good books to learn Spring framework](http://javarevisited.blogspot.sg/2013/03/5-good-books-to-learn-spring-framework-mvc-java-programmer.html). That will certainly help on  your learning process. Since Spring is now a must have skill for Java programmers, it worth putting time and effort to learn this powerful framework

## Difference between Setter and Constructor Injection in Spring framework

As I said earlier Spring supports both setter and constructor Injection which are two standard way of injecting dependency on beans managed by IOC constructor. Spring framework doesn't support Interface Injection on which dependency is injected by implementing a particular interface. In this section we will see couple of difference between setter and constructor Injection, which will help you decide when to use setter Injection over constructor Injection in Spring and vice-versa.

1) Fundamental difference between setter and constructor injection, as there name implies is How dependency is injected. Setter injection in Spring uses setter methods like setDependency() to inject dependency on any bean managed by Spring's IOC container. On the other hand constructor injection uses [constructor](http://javarevisited.blogspot.sg/2012/01/what-is-constructor-overloading-in-java.html) to inject dependency on any Spring managed bean.

2) Because of using setter method, setter Injection in more readable than constructor injection in Spring configuration file usually applicationContext.xml . Since setter method has name e.g. setReporotService() by reading Spring XML config file you know which dependency you are setting. While in constructor injection, since it uses index to inject dependency, its not as readable as setter injection and you need to refer either Java documentation or code to find which index corresponds to which property.

3) Another difference between setter vs constructor injection in Spring and one of the drawback of setter injection is that it does not ensures [dependency Injection](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html). You can not guarantee that certain dependency is injected or not, which means you may have an object with incomplete dependency. On other hand constructor Injection does not allow you to construct object, until your dependencies are ready.

4) One more drawback of setter Injection is Security. By using setter injection, you can [override](http://javarevisited.blogspot.in/2011/12/method-overloading-vs-method-overriding.html) certain dependency which is not possible which is not possible with constructor injection because every time you call constructor, a new object is gets created.  
  
  
5) One of our reader Murali Mohan Reddy, pointed out one more difference between Setter and Constructor injection in Spring, where later can help, if there is a circular dependency between two object A and B.

If Object A and B are dependent each other i.e A is depends ob B and vice-versa. Spring throws ObjectCurrentlyInCreationException while creating objects of A and B bcz A object cannot be created until B is created and vice-versa. So spring can resolve circular dependencies through setter-injection. Objects constructed before setter methods invoked.

See comment section for more inputs from other readers.

## When to use Setter Injection over Constructor Injection in Spring

Setter Injection has upper hand over Constructor Injection in terms of readability. Since for configuring Spring we use [XML files](http://javarevisited.blogspot.in/2011/12/parse-xml-file-in-java-example-tutorial.html), readability is much bigger concern. Also drawback of setter Injection around ensuring mandatory dependency injected or not can be handled by configuring Spring to check dependency using "dependency-check" attribute of tag or tag. Another worth noting point to remember while comparing Setter Injection vs Constructor Injection is that, once number of dependency crossed a threshold e.g. 5 or 6 its handy manageable to passing dependency via constructor. Setter Injection is preferred choice when number of dependency to be injected is lot more than normal, if some of those arguments is optional than using [Builder design pattern](http://javarevisited.blogspot.in/2012/06/builder-design-pattern-in-java-example.html) is also a good option.

In Summary both Setter Injection and Constructor Injection has there own advantage and disadvantage. Good thing about Spring is that it doesn't restrict you to use either Setter Injection or Constructor Injection and you are free to use both of them in one Spring configuration file. Use Setter injection when number of dependency is more or you need readability. Use Constructor Injection when Object must be created with all of its dependency.  
  
  
**Further Reading on Spring Framework**

* [Spring Recipes: A Problem-Solution Approach](http://www.amazon.com/Spring-Recipes-Problem-Solution-Approach-Experts/dp/1430224991?tag=javamysqlanta-20) By Gary Mak
* [Expert Spring MVC and Web Flow](http://www.amazon.com/Expert-Spring-Flow-Experts-Voice/dp/159059584X?tag=javamysqlanta-20)  By Seth Ladd, Darren Davison, Steven Devijver,  and Colin Yates

Other **Spring tutorials** from Javarevisited Blog

* How to setup LDAP authentication in Java using Spring Security? ([solution](http://javarevisited.blogspot.in/2012/03/spring-security-example-tutorial-how-to.html" \t "_blank))
* How to limit maximum number of concurrent active session in Java web app? ([demo](http://javarevisited.blogspot.in/2012/03/spring-security-example-tutorial-how-to.html" \t "_blank))
* How to convert ArrayList to delimited String in Java using Spring? ([solution](http://javarevisited.blogspot.in/2012/08/convert-collection-to-string-in-java.html" \t "_blank))
* How to get ServletContext object in Spring controller? ([example](http://javarevisited.blogspot.sg/2012/03/how-to-get-servletcontext-in-servlet.html" \t "_blank))
* 10 example of display tag in JSP and spring ([examples](http://javarevisited.blogspot.sg/2011/09/displaytag-examples-tutorial-jsp-struts.html))
* What is default bean scope in Spring MVC framework? ([answer](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html" \t "_blank))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://javarevisited.blogspot.com.by/2011/11/ldap-authentication-active-directory.html>

LDAP Active Directory Authentication in Java Spring Security Example Tutorial

**LDAP authentication** is one of the most popular authentication mechanism around the world for enterprise application and **Active directory** (an LDAP implementation by Microsoft for Windows) is another widely used ldap server. In many project we need to *authenticate against active directory using ldap* by credentials provided in login screen. Some time this simple task gets tricky because of various issues faced during implementation and integration *and no standard way of doing ldap authentication*. Java provides ldap support but in this article I will mostly talk about **spring security** because its my preferred Java framework for authentication, authorization and security related stuff. you can do same thing in Java by writing your own program for doing **LDAP search** and than **LDAP bind** but as I said its much easier and cleaner when you use spring security for LDAP authentication.

Along with LDAP Support, Spring Security also provides several other feature which is required by enterprise java application including SSL Security, encryption of passwords and session timeout facilities.

### LDAP Authentication Basics

Before getting deep into LDAP authentication on Active Directory, let's get familiar with some LDAP term because most of the time user is doing it first time and they are not very familiar with typical LDAP glossary such as Dn , Ou , Bind or search etc.

**Dn - Distinguished name**, a unique name which is used to find user in LDAP server e.g. Microsoft Active Directory.

**Ou - Organization Unit**

**Bind - LDAP Bind** is an operation in which ldap clients sends bindRequest to ldap user including username and password and if LDAP server able to find user and password correct, it allows access to ldap server.

**Search - LDAP search** is an operation which is performed to retrieve Dn of user by using some user credential.

**Root** - LDAP directory's top element, like Root of a tree.

**BaseDn** - a branch in LDAP tree which can be used as base for ldap search operation e.g. dc=Microsoft,dc=org"

If you want to know more about LDAP check this link it has detailed information on LDAP.

## LDAP Authentication Active Directory Spring

### Active Directory Authentication in Spring Security

There are two ways to implement active directory authentication using LDAP protocol in spring security, first way is **programmatic and declarative** way which requires some coding and some configuration and second way is an out of box solution from spring security which just require to configure ActireDirectoryAuthentication provider and you are done. we will see both approach but I suggest using second one because of its simplicity and easy to use feature.

### Active Directory Authentication using LDAP in Spring Security -1

**Configuration**

Add following configuration into your spring application-context.xml file, I would suggest to put this configuration in a separate **application-context-security.xml** file along with other security related stuff.

**1) Configuring LDAP Server**

<s:ldap-server url="ldap://stockmarket.com"   **//ldap url**

port="389"                              //ldap port

manager-dn="serviceAcctount@sotckmarket.com" //manager username

manager-password="AD83DgsSe"/>             //manager password

This configuration is self explanatory but briefly few lines about manager-dn and password, **Ldap authentication on active directory** or any other ldap directory is performed in two steps first an LDAP search is performed to locate Dn(Distinguised Name) of user and than this Dn is used to perform LDAP Bind , if bind is successful than usre authentication is successful other wise it fails. Some people **prefer remote compare of password  than LDAP bind**, but LDAP bind is what you mostly end of doing. most of Active directory doesn't allow Anonymous Search operation , so to **perform an ldap search** your service must have an LDAP account which is what we have provided here in manager-dn and manager-password.

In Summary now LDAP login will be done on these step

1) Your Service or application bind itself with LDAP using manager-dn and manager-password.

2) LDAP search for user to find UserDn

3) LDAP bind using UserDn

**2) Configuring LDAP Authentication Provider**

<s:authentication-manager erase-credentials="true">

<s:ldap-authentication-provider

user-search-base="dc=stockmarketindia,dc=trader"

user-search-filter="userPrincipalName={0}"

/>

<s:authentication-provider ref="springOutOfBoxActiveDirecotryAuthenticationProvider"/>

</s:authentication-manager>

This section specifies various **authentication provider in spring-security** here you can see your LDAP authentication provider and we are using **userPrincipalName** to search user inside Microsoft Active directory.

Now small piece of coding to pass userPrincipalName and authenticate user.

public boolean login(String username, String password) {

AndFilter filter = new AndFilter();

ldapTemplate.setIgnorePartialResultException(true); **// Active Directory doesn’t transparently handle referrals. This fixes that.**

filter.and(new EqualsFilter("userPrincipalName", username));

return ldapTemplate.authenticate("dc=stockmarketindia,dc=trader", filter.toString(), password);

}

line 2 is very important in this program because I spent whole day figuring out when my application was repeatedly throwing **javax.naming.PartialResultException: Unprocessed Continuation Reference(s)**

you can also use **sAMAccountName** for searching user, *both userPrincipalName and sAMAccountName are unique in Active directory*. What is important here is that it has to be full name e.g. name@domain like [jimmy@stockmarket.com](mailto:jimmy@stockmarket.com).

authenticate() method will return true or false based on result of bind operation.

### Active Directory Authentication using LDAP in Spring Security - 2

Second approach is much simpler and cleaner because it comes out of box , you just need to configure ldap server url and domain name and it will work like cream.

<s:authentication-manager erase-credentials="true">

<s:authentication-provider ref="ldapActiveDirectoryAuthProvider"/>

</s:authentication-manager>

<bean id="ldapActiveDirectoryAuthProvider" class="org.springframework.security.ldap.authentication.ad.ActiveDirectoryLdapAuthenticationProvider">

<constructor-arg value="stockmarket.com" />  **//your domain**

<constructor-arg value="ldap://stockmarket.com/" />  **//ldap url**

</bean>

Done. This configuration will both authenticate and load all the granted authority from LDAP like group which you are member of. This is integrated with spring security login element also.

**Dependency**

This example is based on spring security 3.0 and I was using spring-ldap-1.3.1.RELEASE-all.jar and spring-security-ldap-3.1.0.RC3.jar

**Errors during LDAP authentication**

you need to be very lucky to complete LDAP authentication against Active directory without any error or exception, here I am listing down some common error which I encountered and there solution for quick reference.

**1) javax.naming.PartialResultException: Unprocessed Continuation Reference(s); remaining name 'dc=company,dc=com'**

This error comes because Microsoft Active Directory doesn't handle referrals properly and to fix this set this property

ldapTemplate.setIgnorePartialResultException(true);

**2) javax.naming.NameNotFoundException: [LDAP: error code 32 - No Such Object]; remaining name ''**

This error solved with some trial and error and mainly came **due to invalid format of username**. it solved by providing full name e.g. jemmy@stockmarket.com

**Tools**

**LDAP Browser**: Having some tools to look data inside LDAP directory is best it gives you some visibility as well as means to browse data in LDAP. it's called as LDAP browser and there are lot of open source LDAP browser available in web e.g. jexplorer. you can browse and see data inside

Active directory by using LDAP browser.

### Ldap Active directory Authentication over SSL

This works perfectly to implement ldap authentication against microsoft active directory. but one thing you might want to put attention is that with ldap username and password travel to ldap server as clear text and anyone who has access to ldap traffic can sniff user credential so its not safe. one solution is to use ldaps( LDAP over ssl) protocol which will encrypt the traffic travels between ldap client and server. this is easy to do in spring-security what you need to change is the url instead of "**ldap://stockmarket.com/**" you need ot use ""**ldaps://stockmarket.com/**". actually **port for ldap is 339** and for **ldaps is 636** but that's been taken care by spring in second approach, in first approach you need to provide this information.

What problem you may face is **"unable to find valid certification path to requested target"**

exception  as shown below:

javax.net.ssl.SSLHandshakeException:

sun.security.validator.ValidatorException: PKIX path building failed:

sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

Reason of this Exception is simple, **Certificate returns during SSL handshake is not signed by any trustsed Certification Authority(CA) which is configured in you JRE keysotre** e.g Verisign, Thwate, goDaddy or entrust etc. Instead Server is sending certificate which is not known to JRE.

To solve this problem you need to add certificates returned by Server into JRE's keystore.

**What I did to solve the problem**

Nothing surprising , I use an open source program called **InstallCert.java** , just run with your ldap server and port and it will try to connect LDAP server using SSL and first throw same "PKIX path building failed" and then Certificates returned by LDAP server. It will then ask you to add Certificate into keystore just give certificate number as appeared in your screen and it will then add those certificate into "**jssecacerts**" inside **C:\Program Files\Java\jdk1.6.0\jre\lib\security** folder. Now re-run the program that error must be disappeared and

it will print:

"Loading KeyStore jssecacerts...

Opening connection to stockmarket.com:636...

Starting SSL handshake...

No errors, certificate is already trusted

you are done, now if you try authenticating against ldaps you will be succeed.

There are many other approach to perform **ldap authentication against active directory** even without spring security by using Java. but I found spring-security very helpful so consider using it for your security requirement. let me know if you face any issue during ldap login and I will try my best to help you.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) What is Spring IOC Container ?

Spring IOC container is an implementation of Dependency Injection design pattern provided by Spring framework. Instead of creating objects in application code using new() keyword, we use ApplicationContext.getBean("nameOfBean") to get reference of Objects. Spring framework creates objects, manages object and set the dependency automatically so that you can get Object in proper state

2) Does Singleton from Spring Container is thread safe?  
3) Why Spring MVC is better than Struts?  
4) What is Spring Integration?  
5) How to call remote method by RMI using Spring  
6) What scheduling feature Spring framework provides?  
7) How do you make a Singleton bean to lazy load in ApplicationContext which loads all Singleton beans eagerly during startup?  
8) Does Spring Security part of Spring framework?  
9) How to configure Spring using Annotation  
10) Which version of Spring have you used recently and what is difference you observed from previous spring version.

What is benefit of using JdbcTemplate ? Why should you use JdbcTemplate in Spring ?  
How do you handle SQLException in Spring?  
How to you setup JDBC connection pool in Spring etc.  
interview questions from JDBC in Spring is not that many, but you should be familiar with JdbcTemplate.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

How would you inject a prototype bean into a singleton bean from a method?   
  
I answered that I would create the prototype bean using the application context giving the bean id inside the method of the singleton class.   
But apparently there is another way of doing this:  
1. Use element when creating the Singleton class  
2. using the element when creating the Prototype class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*