Spring Interview Questions

[](http://2.bp.blogspot.com/-f6WlzIi6PU0/VVDX_DRwxiI/AAAAAAAAAJE/R0wR332y5_4/s1600/SpringInterviewQuestions.png)

1. [What is Dependency Injection?](http://www.javainterview.in/p/spring-interview-questions.html#question1)
2. [Why is Spring one of the most popular Java related frameworks?](http://www.javainterview.in/p/spring-interview-questions.html#question2)
3. [What are the different modules in Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html#question3)
4. [Can you give an overview of a web application that is implemented using Spring related Modules?](http://www.javainterview.in/p/spring-interview-questions.html#question4)
5. [What is the simplest way of ensuring that we are using single version of all Spring related dependencies?](http://www.javainterview.in/p/spring-interview-questions.html#question5)
6. [What are the major features in different versions of Spring ?](http://www.javainterview.in/p/spring-interview-questions.html#question6)
7. [What are the latest specifications supported by Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html#question7)
8. [Can you describe some of the new features in Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html#question8)
9. [What is auto-wiring?](http://www.javainterview.in/p/spring-interview-questions.html#question9)
10. [What would happen in Spring Container finds multiple bean definitions matching the property to be auto wired?](http://www.javainterview.in/p/spring-interview-questions.html#question10)
11. [How is Spring’s singleton bean different from Gang of Four Singleton Pattern?](http://www.javainterview.in/p/spring-interview-questions.html#question11)
12. [How do you represent stateful bean in Spring?](http://www.javainterview.in/p/spring-interview-questions.html#question12)
13. [How do you use values defined in a property file in an application context xml?](http://www.javainterview.in/p/spring-interview-questions.html#question13)
14. [How is validation done using Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html#question14)
15. [How do you implement cross cutting concerns in a web application?](http://www.javainterview.in/p/spring-interview-questions.html#question15)
16. [What is an Aspect and Pointcut in AOP?](http://www.javainterview.in/p/spring-interview-questions.html#question16)
17. [What are the different types of AOP advices?](http://www.javainterview.in/p/spring-interview-questions.html#question17)
18. [How do you define transaction management for Spring – Hibernate integration?](http://www.javainterview.in/p/spring-interview-questions.html#question18)
19. [How do you choose the framework to implement AOP - Spring AOP or AspectJ?](http://www.javainterview.in/p/spring-interview-questions.html#question19)
20. [What are the different mock objects provided by Spring test framework?](http://www.javainterview.in/p/spring-interview-questions.html#question20)
21. [What are the utility methods available to test JDBC classes?](http://www.javainterview.in/p/spring-interview-questions.html#question21)
22. [How do you setup a Session Factory to integrate Spring and Hibernate?](http://www.javainterview.in/p/spring-interview-questions.html#question22)
23. [How do you implement caching with Spring framework?](http://www.javainterview.in/p/spring-interview-questions.html#question23)
24. [What are the important features of Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question31)
25. [What are the important concepts related to setting up a Job in Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question32)
26. [What are the different ItemReader and ItemWriter implementations available with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question33)
27. [How do you start running a Spring Batch Job?](http://www.javainterview.in/p/spring-interview-questions.html#question34)
28. [How do you configure parallel execution of steps with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html#question35)

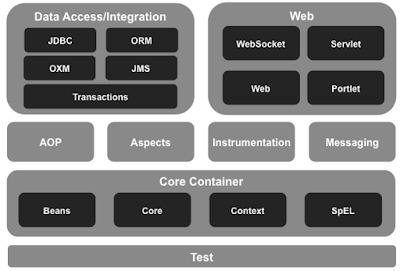
[What is Dependency Injection?](http://www.javainterview.in/p/spring-interview-questions.html)

Most java classes are dependant on a large number of other classes. For example,a ClientService will depend on ClientDO to get the data from database. Before Spring Framework, ClientService class would directly create an instance of ClientDO class by using code like “new ClientDO()” (in constructor, for example). This introduces tight coupling between ClientService and ClientDO. If I want ClientService to use some other data instead of that provided by ClientDO, we need to reimplement ClientService.  
This is where Spring framework comes in. It removes tight coupling between classes. In above example, Spring framework would inject ClientDO into ClientService class. This concept is called Dependency Injection or Inversion of Control.  
If needed ( when writing a unit test), we can ask Spring framework to inject a different class (which has same interface as ClientDO).

[Why is Spring one of the most popular Java related frameworks?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring framework enables development of loosely coupled classes based on well defined interfaces (Dependency Injection and IOC). It makes writing testable code simple.  
Beauty of Spring framework is that it provides great integration support with other non Spring open source frameworks – In case you would want to integrate with frameworks like AspectJ (instead of Spring AOP) or Struts 2(instead of Spring MVC).  
Spring framework is built upon loosely coupled Spring modules enabling us to pick and choose the specific aspects of the Spring module we would like to use.  
This flexibility is what makes Spring framework one of most popular Java frameworks.

[What are the different modules in Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html)

[](http://1.bp.blogspot.com/-ZOTeCgnbWfY/VVDYU2rEUxI/AAAAAAAAAJM/ENNZTy3RLZ0/s1600/SpringModules.png)

The Core package provides

* IoC
* Dependency Injection

BeanFactory, an implementation of the factory pattern, helps us decouple configuration and injection of dependencies from program logic.  
The DAO package provides an abstraction over JDBC to simplify writing code that interacts with database. Declarative transaction management is an addition feature provided by DAO package.  
The ORM package provides integration for Spring with most popular JPA implementations (Hibernate etc) and Query Mapping (iBatis).  
Spring AOP package provides a basic AOP implementation featuring definition of interceptors and pointcuts. Cross cutting concerns like security and transaction management can be implemented using Spring AOP.  
Spring Web package provides

* multipart file-upload functionality
* Integration with Struts and other MVC Frameworks

Spring's MVC package provides a clean implementation of the MVC model for web applications.

[Can you give an overview of a web application that is implemented using Spring related Modules?](http://www.javainterview.in/p/spring-interview-questions.html)

Different Layers of a web application can be implemented using different spring modules. Beauty of Spring framework is that Spring provides great integration support with other non Spring open source frameworks – In case you would want to integrate with frameworks like AspectJ (instead of Spring AOP) or Struts 2(instead of Spring MVC).  
Service & Business Layers

* Core Business Logic using simple POJOs, managed by Spring's IoC container
* Transaction Management using Spring AOP

Integration Layer

* Spring ORM to integrate with Databases (JPA & iBatis).
* Spring JMS to intergrate with external interfaces using JMS
* Spring WS to consume web services.

Web Layer

* Spring MVC to implement MVC pattern.
* Spring WS to expose web services.

[What is the simplest way of ensuring that we are using single version of all Spring related dependencies?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring is made up of a number of small components (spring-core, spring-context, spring-aop, spring-beans and so on). One of the quirks of using Spring framework is the dependency management of these components. Simple way of doing this is using Maven "Bill Of Materials" Dependency.

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-framework-bom</artifactId>

<version>4.1.6.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

All spring dependencies can now be included in this and the child poms without using version.

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

</dependency>

<dependencies>

[What are the major features in different versions of Spring ?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring 2.5 made annotation-driven configuration possible.  
Spring 3.0 made great use of the Java 5 improvements in language.  
Spring 4.0 is the first version to fully support Java 8 features. Minimum version of Java to use Spring 4 is Java SE 6.

[What are the latest specifications supported by Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring Framework 4.0 supports the Java EE 7 specifications

* JMS 2.0
* JTA 1.2
* JPA 2.1
* Bean Validation 1.1
* JSR-236 for Concurrency.

[Can you describe some of the new features in Spring 4.0?](http://www.javainterview.in/p/spring-interview-questions.html)

Following are some of the new features in Spring 4.0 and Spring 4.1

* spring-websocket module provides support for WebSocket-based communication in web applications.
* Spring Framework 4.0 is focused on Servlet 3.0+ environments
* @RestController annotation is introduced for use with Spring MVC applications
* Spring 4.1 introduces @JmsListener annotations to easily register JMS listener endpoints.
* Spring 4.1 supports JCache (JSR-107) annotations using Spring’s existing cache configuration.
* Jackson’s@JsonViewissupporteddirectlyon@ResponseBodyandResponseEntitycontroller methods for serializing different amounts of detail for the same POJO

[What is auto-wiring?](http://www.javainterview.in/p/spring-interview-questions.html)

The Spring container can autowire dependencies into interacting beans. Spring container can resolve dependencies by looking at the other beans defined in the ApplicationContext. This elimates the need for maintaining an xml specifying beans and their dependencies. Autowiring can be done byName (name of the property) and byType (type of the property).

[What would happen in Spring Container finds multiple bean definitions matching the property to be auto wired?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring framework does not resolve this kind of ambiguity. It would throw and exception. The programmer has the choice to remove one of the bean or explicitly wire in a dependency.

[How is Spring’s Singleton bean different from Gang of Four Singleton Pattern?](http://www.javainterview.in/p/spring-interview-questions.html)

The singleton scope is the default scope in Spring. The Gang of Four defines Singleton as having one and only one instance per ClassLoader. However, Spring singleton is defined as one instance of bean definition per container.

[How do you represent Stateful bean in Spring?](http://www.javainterview.in/p/spring-interview-questions.html)

Stateful beans are represented in Spring with a prototype scope. A new instance is created every time a request for bean is made. Example below.  
￼<bean id="state" class="com.foo.SomeState" scope="prototype"/>

[How do you use values defined in a property file in an application context xml?](http://www.javainterview.in/p/spring-interview-questions.html)

Following example shows how to configure a data source using the url, username and password configured in a property file (database-connection.properties)

<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations" value="classpath:database-connection.properties"/>

</bean>

<bean id="dataSource" destroy-method="close" class="org.apache.commons.dbcp.BasicDataSource">

<property name="driverClassName" value="${jdbc.driverClassName}"/>

<property name="url" value="${jdbc.url}"/>

<property name="username" value="${jdbc.username}"/>

<property name="password" value="${jdbc.password}"/>

</bean>

[How is validation done using Spring Framework?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring validator can be used both in web and business layers to validate objects. It is based on the org.springframework.validation.Validator interface having two important methods

* supports(Class) – does this validator support a particular class
* validate(Object, org.springframework.validation.Errors) – validates and sets errors into Errors object

An example is provided below: MessageCodesResolver can be used to convert the message code into proper internationalized text.

public class CarValidator implements Validator {

public boolean supports(Class clazz) {

return Car.class.equals(clazz);

}

public void validate(Object obj, Errors e) {

ValidationUtils.rejectIfEmpty(e, "name", "name.is.empty");

Car c = (Car) obj;

if (c.getUsedYears() < 0) {

e.rejectValue("usedYears", "not.yet.bought");

}

}

}

[How do you implement cross cutting concerns in a web application?](http://www.javainterview.in/p/spring-interview-questions.html)

Functionality spanning multiple layers of an application are called cross cutting concerns. Examples are logging, security, declarative transactions.  
Cross cutting concerns are best implemented using Aspect Oriented Programming (AOP). AOP enables us to apply the cross cutting features across multiple classes.

[What is an Aspect and Pointcut in AOP?](http://www.javainterview.in/p/spring-interview-questions.html)

Two important AOP concepts are

* Aspect : Aspect is the concern that we are trying to implement generically. Ex: logging, transaction management. Advice is the specific aspect of the concern we are implementing.
* Pointcut : An expression which determines what are the methods that the Advice should be applied on.

[What are the different types of AOP advices?](http://www.javainterview.in/p/spring-interview-questions.html)

* Before advice : Executed before executing the actual method.
* After returning advice: Executed after executing the actual method.
* After throwing advice : Executed if the actual method call throws an exception.
* After (finally) advice : Executed in all scenarios (exception or not) after actual method call.
* Around advice : Most powerful advice which can perform custom behavior before and after the method invocation.

[How do you define transaction management for Spring – Hibernate integration?](http://www.javainterview.in/p/spring-interview-questions.html)

First step is to define a a transaction manager in the xml.

<bean id="transactionManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager"

p:sessionFactory-ref="sessionFactory" />

<tx:annotation-driven/>

Next, we can add the @Transactional annotation on the methods which need to part of a transactions.

@Transactional(readOnly = true)

public class CustomerDaoImpl implements CustomerDao {

[How do you choose the framework to implement AOP - Spring AOP or AspectJ?](http://www.javainterview.in/p/spring-interview-questions.html)

Spring AOP is very simple implementation of AOP concepts. Its an ideal fit If the needs of an application are simple like

* Simple method executions and/or
* Advising the execution of operations only on spring beans

AspectJ is a full fledged AOP framework that helps you

* Advise objects not managed by the Spring container .
* Advise join points other than simple method executions (for example, field get or set join points).

[What are the different mock objects provided by Spring test framework?](http://www.javainterview.in/p/spring-interview-questions.html)

* org.springframework.mock.env package : Mock implementations of the Environment and PropertySource abstractions.
* org.springframework.mock.jndi package : Implementation of the JNDI SPI, which can be used to create a simple JNDI environment for test suites.
* org.springframework.mock.web package : Servlet API mock objects to test Spring MVC classes like controllers.

[What are the utility methods available to test JDBC classes?](http://www.javainterview.in/p/spring-interview-questions.html)

JdbcTestUtils provides the following static utility methods.

* countRowsInTable(..) & deleteFromTables(..): counts/deletes the rows in the given table
* countRowsInTableWhere(..) & deleteFromTableWhere(..): counts /deletes the rows in the given table, using the provided WHERE clause
* dropTables(..): drops the specified tables

[How do you setup a Session Factory to integrate Spring and Hibernate?](http://www.javainterview.in/p/spring-interview-questions.html)

Hibernate SessionFactory can be configured as a spring bean.

<bean id="mySessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<property name="dataSource" ref="specificDataSource"/>

<property name="mappingResources">

<list>

<value>tablename.hbm.xml</value>

</list>

</property>

<property name="hibernateProperties">

<value>

hibernate.dialect=org.hibernate.dialect.HSQLDialect

</value>

</property>

</bean>

Datasource can be created from JNDI.

￼<beans>

<jee:jndi-lookup id=" specificDataSource " jndi-name="java:comp/env/jdbc/specificds"/>

</beans>

[How do you implement caching with Spring framework?](http://www.javainterview.in/p/spring-interview-questions.html)

Enabling caching in Spring is all the matter of making the right annotations available in appropriate methods. First, we need to enable caching. This can be done using Annotation or XML based configuration. Below example shows enabling Spring based caching using Annotation.

@Configuration

@EnableCaching

public class AppConfig {

}

Next step is to add the annotations on the method we would like to cache results from.

@Cacheable("players")

public Player findPlayer(int playerId) {

...

}

Spring offers features to control caching

* Conditional Caching : condition="#name.length < 32"
* @CachePut : Update existing cache. For example, when player details cached above are updated.
* @CacheEvict : Remove from cache. For example, the player is deleted.

Spring also provides integration with well known caching frameworks like EhCache. Sample xml configuration shown below.

<bean id="cacheManager"

class="org.springframework.cache.ehcache.EhCacheCacheManager" p:cache-manager-ref="ehcache"/>

<!-- EhCache library setup -->

<bean id="ehcache" class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean" p:config-

location="ehcache.xml"/>

[What are the important features of Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html)

* Restartability : Easy to restart a batch program from where it failed
* Different Readers and Writers : Provides great support to read from JMS, JDBC, Hibernate, iBatis etc. It can write to JMS, JDBC, Hibernate and more.
* Chunk Processing : If we have 1 Million records to process, these can be processed in configurable chunks (1000 at a time or 10000 at a time).
* Easy to implement proper transaction management even when using chunk processing.
* Easy to implement parallel processing. With simple configuration, different steps can be run in parallel.

[What are the important concepts related to setting up a Job in Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html)

A Job in Spring Batch is a sequence of Steps. Each Step can be configured with

* next : next step to execute
* tasklet : task or chunk to execute. A chunk can be configured with a Item Reader, Item Processor and Item Writer.
* decision : Decide which steps need to executed.

[What are the different ItemReader and ItemWriter implementations available with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html)

Important ones are those that allow to read/write from

* Flat File
* Hibernate Cursor
* JDBC
* JMS
* Hibernate Paging
* Stored Procedure

[How do you start running a Spring Batch Job?](http://www.javainterview.in/p/spring-interview-questions.html)

A Job Launcher can be used to execute a Spring Batch Job. A job can be launched/scheduled in a web container as well.  
Each execution of a job is called a Job Instance. Each Job Instance is provided with an execution id which can be used to restart the job (if needed).  
Job can be configured with parameters which can be passed to it from the Job Launcher.

[How do you configure parallel execution of steps with Spring Batch?](http://www.javainterview.in/p/spring-interview-questions.html)

To execute jobs in parallel we can use a split. Example shows configuring a split. In below example, flow1 (step1 and step2 sequence) is executed in parallel with flow2 (step3). Step4 is executed after both flows are complete.

<job id="job1">

<split id="split1" task-executor="taskExecutor" next="step4">

<flow>

<step id="step1" parent="s1" next="step2"/>

<step id="step2" parent="s2"/>

</flow>

<flow>

<step id="step3" parent="s3"/>

</flow>

</split>

<step id="step4" parent="s4"/>

</job>