**Java Interview Checklist**

1. Quick Brief Introduction on work Experience.

* Like Skill set ,year of experience
* Area of interest Front end, Business Layer or DB
* Role Lead or developer.

1. Last Application

* Technology used
* Architecture.

1. Core Java

* Basic OOPS concept
* Annotation
* Java collections. (what to use if we need to store unique custom object)
* Class loader and its type.
* Error & Exception.
* JVM argument for out of memory

1. Web Component basic & end to end workflow.

* Any frame work like Struts,JSF,Spring

1. EJB, Webservice, JMS, Hibernate, JNDI, LDAP, JDBC, Hibernate
2. Design Pattern

* Singleton

1. Application Server Basic

* Namespace Binding, Datasource, Connection Factory, Connection Pool
* Class loader order in J2EE.

1. IDE Expertise
2. Development Process Followed

* Test driven development, Junit, Build Process, Test Env
* Tools for source code management. Git
* Tool for monitoring Application health like (Splunk,Wily)

1. Building and deploying application.

* Like Jenkin,Hudson etc

1. Performance Testing Experience.
2. Individual

* Knowledge/understanding of the business.
* Ability to communicate and interact effectively with the business users/customers.
* Ability to persuade/convince business with alternativesolutions.
* Ability to communicate effectively with your fellow developers, immediate and senior management.
* Ability to work in a team as well as independently.
* Problem solving/analytical skills.
* Organizational skills.
* Ability to cope with difficult situations like stress due to work

load, deadlines etc and manage or deal with difficult people.

1. RTR Specific Question

* Any exp on xml, How many type of xml parser you know
* How can delete, add or update node in xml what api you have used.
* Transformation Using xsl
* INS Domain Knowledge.
* Service Call , Junit etc
* Hibernate transaction management.
* Testing automation via Jenkin.

Sample Feed Back

Hi,

We have interviewed **ABC** this morning. He is okay technically and needs training on Front-end technologies like JSP, JSF, Portals etc.. He is well aware of middleware and worked on frameworks and tools that are used in Kemper.

ABC has experience in coding using core Java, Spring, MDBs, JMS, Git, Stash, Control-M, Splunk, Webshepre Application Server console.

A very plus point and value add for **ABC** is: he already worked in Kemper and he knows the important teams and key people (Distributed Systems, Architects, Bert Miranda’s group, Gaurav) in this account. This is very critical to EWS team and I believe this would help him to get acquainted with the client very quickly.

I checked with XYZ and we both believe that **ABC** is a good match for LMN team, but he would definitely need training in Front-end technologies.

**RTR Xml Specific**

1. Worked on XML processing like add delete or update node What API used. Example

javax.xml.parsers.DocumentBuilder;

javax.xml.parsers.DocumentBuilderFactory;

javax.xml.parsers.ParserConfigurationException;

javax.xml.transform.TransformerException;

org.apache.xpath.CachedXPathAPI;

org.apache.xpath.XPathAPI;

org.w3c.dom.Document;

org.w3c.dom.Element;

org.w3c.dom.NamedNodeMap;

org.w3c.dom.Node;

org.w3c.dom.NodeList;

Node node = XPathAPI.selectSingleNode(xmlDocument,"xpath goes here");

String info = node.getFirstChild().getNodeValue();

1. Type of XML parser and Transformation of xml from client format to Custom format using xsl.

Or what approach you would use Example.

javax.xml.transform.Transformer;

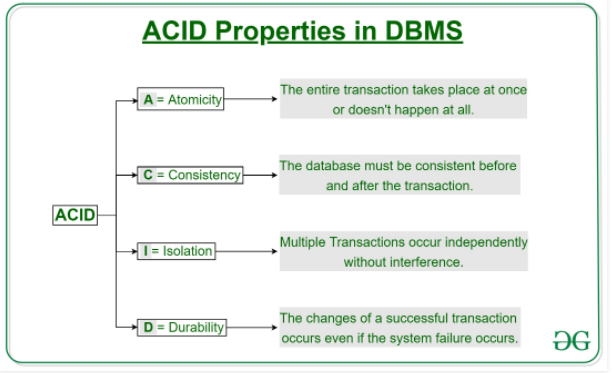
TransformerFactory factory = TransformerFactory.newInstance();

Transformer transformer = factory.newTransformer(xslSource.xsl);

Document customerFormarxml = transformer.transform(sourceFormatxmlDocument);

1. ACID Properties in DBMS (**Atomicity, Consistency,** **Isolation & Durability)**

[**transaction**](https://www.geeksforgeeks.org/sql-transactions/) is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.   
In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.



**Spring Question Answer**

1. What is Spring Frame Work how this is different from other like Struts or JSF.

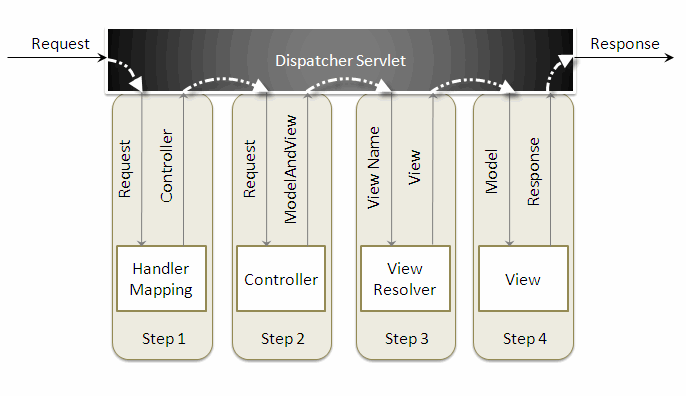
Some Module of Spring like AOP, IOC Container, Security, Transaction Mgmnt etc.

Reference : <https://www.javacodegeeks.com/java-spring-tutorial>

1. What is Spring Frame Work WorkFlow <https://dzone.com/tutorials/java/spring/spring-mvc-tutorial-1.html>

spring mvc helps in building flexible and loosely coupled web applications. the model-view-controller design pattern helps in seperating the business logic, presentation logic, and navigation logic. models are responsible for encapsulating the application data. the views render a response to the user with the help of the model object. controllers are responsible for receiving the request from the user and calling the back-end services.

the figure below shows the flow of requests in the spring mvc framework.



when a request is sent to the spring mvc framework the following sequence of events happen.

* the dispatcherservlet first receives the request.
* the dispatcherservlet consults the handlermapping and invokes the controller associated with the request.
* the controller processes the request by calling the appropriate service methods and returns a modelandview object to the dispatcherservlet . the modelandview object contains the model data and the view name.
* the dispatcherservlet sends the view name to a viewresolver to find the actual view to invoke.
* now, the dispatcherservlet will pass the model object to the view to render the result.
* the view , with the help of the model data, will render the result back to the user.

1. Some of the spring core framework [annotations](https://www.journaldev.com/721/java-annotations) are:

<https://www.journaldev.com/16966/spring-annotations>

@Configuration @ComponentScan@Component @Controller

@Transactionalis the spring declarative transaction management annotation,

@EnableWebSecurity is used with @Configuration class to have the Spring Security configuration defined

@Controller annotation is used to indicate that it’s a web controller class.

@RequestMapping is used with classes and methods to redirect the client

1. What is AOP, Example of Spring AOP if have used any where.

Declarative transactions separates transaction management code from the business logic. Spring supports declarative transactions using transaction advice (using AOP) via XML configuration in the spring context or with **@Transactional** annotation.

Reference : <https://www.javacodegeeks.com/2011/01/aspect-oriented-programming-spring-aop.html>

1. Transaction Management in Spring Example of Spring Annotation what annotation to be used.

Reference : <https://www.javacodegeeks.com/2016/05/understanding-transactional-annotation-spring.html>

1. Spring Dependency Injection and Inversion of Control

Reference : <https://www.javacodegeeks.com/2019/02/spring-dependency-injection-inversion-control.html>

**Inversion of Control** is.  
The Inversion of control is a term used in Object Oriented Programming, by which the control of an object or set of objects is given to a framework or a container provided by the framework.

**The Dependency Injection** is a term used in Object Oriented Programming, by which Objects will focus on doing the assigned functionality and utilising other objects. The necessary configurations and initialisations will not be handled by the objects. However, the Objects will provide a way to initialise them and their dependencies by field assignment, field setters or constructors. This way, the external entities can initialise the things and not the actual objects.

In a Spring based application, Inversion of Control Container (IoC container) does the dependency injection. We will see that in coming section. First, let’s see why do we even need such a container.

**Inversion of Control Container (IoC Container)**

Spring provides an IoC Container to solve the problem. This container instantiates all the objects, and while doing so it also resolves their dependencies. The class ApplicationContext represents the Spring IOC Container. The Application context is responsible for instantiating, configuring and wiring the beans.  
*Remember, Beans are nothing but Java objects registered with Spring’s Application Context.*

To configure, instantiate or write beans the Application Context needs some instructions. These instructions can be provided in the form of XML configurations, Java Annotations or Code.

**Spring Dependency Injection**

In **Spring** each object is a bean. Each object has an id, or name. An **ApplicationContext** keeps track of all such beens and ids. When a bean is requested, by a consumer, the Application Context returns an instance of the bean. Look at the below code to understand bean creation and wiring in detail

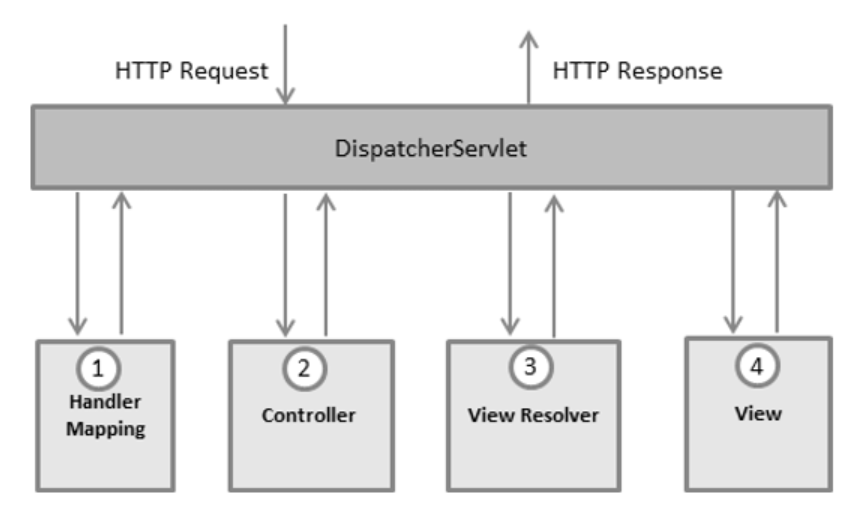
* By DI, the responsibility of creating objects is shifted from our application code to the Spring container; this phenomenon is called IOC.
* Dependency Injection can be done by setter injection or constructor injection.
* Spring helps in the creation of loosely coupled applications because of **Dependency Injection**.
* In Spring, objects define their associations (dependencies) and do not worry about how they will get those **dependencies**. It is the responsibility of Spring to provide the required dependencies for creating objects.

**For example**: Suppose we have an object Employee and it has a dependency on object Address. We would define a bean corresponding to Employee that will define its dependency on object Address.

When Spring tries to create an Employee object, it will see that Employee has a dependency on Address, so it will first create the Address object (dependent object) and then inject it into the Employee object.

* Inversion of Control (**IOC**) and Dependency Injection (**DI**) are used interchangeably. IOC is achieved through DI. DI is the process of providing the dependencies and IOC is the end result of DI. (**Note:** DI is not the only way to achieve IOC. There are [other ways](https://en.wikipedia.org/wiki/Inversion_of_control#Implementation_techniques) as well.)

**Spring MVC Life Cycle :** <https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm>



**What is Transaction Management**

Sequence of action that will be performed to complete database operation and its management is known as Transaction Management. All these action in combination will be treated as ONE action only. So that DB doesn’t fall in inconsistent mode ever. For more details you can search for ACID property of relation DB.

**Type of Transaction Management**

In J2EE, Transaction Management can be divided in two types.

1. Global Transaction
2. Local Transaction

**Global Transaction**

* Use to work with multiple transaction resources like RDBMS or Message Queue (Pros)
* Managed by Application Server (WebSphere, Weblogic) using JTA (Cons)
* JNDI is required to use JTA
* Code can not be reused as JTA is available at server level(Cons)
* Example of Global Transaction : EJB CMT

**Local Transaction**

* Use to work with specific resource(transaction associated with JDBC)
* Can not work across multiple transaction resource opposite to Global transaction (cons)
* Most of web application uses only single resources hence it is best option to use in normal app.

**Spring Framework Transaction Management**

As you can see above there are some pros and cons associated with both approach. Spring transaction management tries to resolve the problem of both transactions. Consistent programming model approach can be used in any environment. Same code will work for different transactions management in different environment.

**Different Approach for transaction management**

Spring supports two different approach for transaction management.

**Programmatic Transaction Management**

### Ref: <https://www.roytuts.com/programmatic-transaction-management-example-in-spring/>

Here you will write code for transaction management.Spring API dependency. Not good for maintenance. Good for development. Flexibity.

* Allows us to manage transactions through programming in our source code.
* This means hardcoding transaction logic between our business logic.
* We use programming to manage transactions
* Flexible, but difficult to maintain with large amount of business logic. Introduces boilerplate between business logic.
* Preferred when relative less transaction logic is to be introduced.

**Declarative Transaction Management**

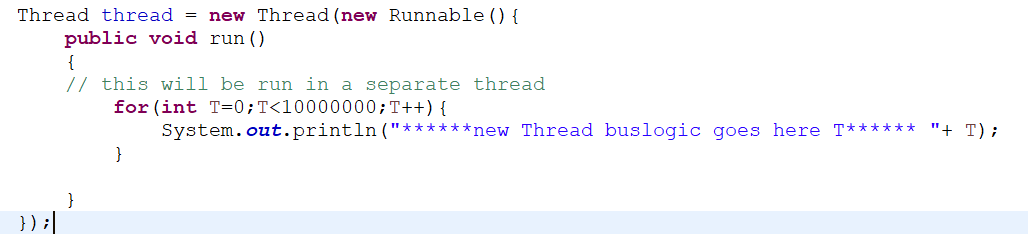
### Ref <https://www.roytuts.com/declarative-transaction-management-example-in-spring/>

Here you will use XML or annotation for transaction management. Less flexible but preferable over programmatic approach. In normal case no code is required for transaction management.

* Allows us to manage transactions through configuration.
* This means separating transaction logic with business logic.
* We use annotations (Or XML files) to manage transactions.
* Easy to maintain. Boilerplate is kept away from business logic.
* Preferred when working with large amount of Transaction logic

**Problem and Solution**

1. What collection you gona use to store a unique Object.
2. You have a sequence of operation to be performed and out of which one is taking more time as it has to send file to other application. Same time we want this to complete in less time. What approach you gona use. JMS or Multithreading
3. How can we initiate some thin in different Thread.
4. [How to call a method with a separate thread in Java](https://stackoverflow.com/questions/3489543/how-to-call-a-method-with-a-separate-thread-in-java) or [How do I execute a method in a new thread using Java?](http://helpdesk.objects.com.au/java/how-do-i-execute-a-method-in-a-new-thread-using-java)



1. Call Soap service in your java program you got the url for same. How would you implement this for QA and PROD Env.
2. Approach to integrate Soap service is Angular App.
3. Do you know concept on Webserver name. Used any clustering env.
4. HTTP Server, Application Server and Load Balancer.
5. What is Singleton Pattern.
6. Why do we use Annotation, What benefit we get have used it any where.
7. What is Reflection. Why do we need Reflection (<https://www.geeksforgeeks.org/reflection-in-java/>)

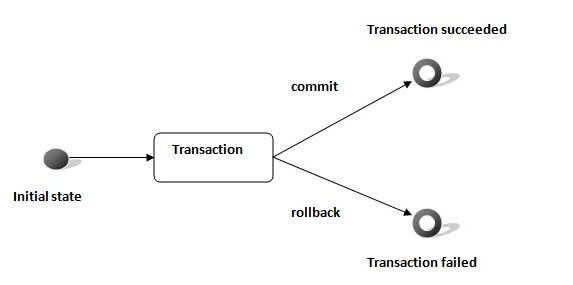
**Reflection** is an API which is used to examine or modify the behavior of methods, classes, interfaces at runtime. ... **Reflection** gives us information about the class to which an object belongs and also the methods of that class which can be executed by using the object

* We can load class from property file using reflection Class.forname(“nameofClass”). No need to compile any code and runtime we can load any class.
* This can be used to inspect class at runtime. Like methods field etc.

**Hibernate Transaction Management Example**

Reference : <https://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-best-practices-tutorial/>

A **transaction** simply represents a unit of work. In such case, if one step fails, the whole transaction fails (which is termed as atomicity). A transaction can be described by ACID properties (Atomicity, Consistency, Isolation and Durability).



**Transaction Interface in Hibernate**

In hibernate framework, we have **Transaction** interface that defines the unit of work. It maintains abstraction from the transaction implementation (JTA,JDBC).

A transaction is associated with Session and instantiated by calling **session.beginTransaction()**.

The methods of Transaction interface are as follows:

1. **void begin()** starts a new transaction.
2. **void commit()** ends the unit of work unless we are in FlushMode.NEVER.
3. **void rollback()** forces this transaction to rollback.
4. **void setTimeout(int seconds)** it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.
5. **boolean isAlive()** checks if the transaction is still alive.
6. **void registerSynchronization(Synchronization s)** registers a user synchronization callback for this transaction.
7. **boolean wasCommited()** checks if the transaction is commited successfully.
8. **boolean wasRolledBack()** checks if the transaction is rolledback successfully.

**Example of Transaction Management in Hibernate**

In hibernate, it is better to rollback the transaction if any exception occurs, so that resources can be free. Let's see the example of transaction management in hibernate.

1. Session session = **null**;
2. Transaction tx = **null**;
4. **try** {
5. session = sessionFactory.openSession();
6. tx = session.beginTransaction();
7. //some action
9. tx.commit();
11. }**catch** (Exception ex) {
12. ex.printStackTrace();
13. tx.rollback();
14. }
15. **finally** {session.close();}

**Core Java Question**

Reference : <https://www.janbasktraining.com/blog/core-java-interview-questions-answers/>

* 1. [Explain the difference between JDK, JRE, and JVM.](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#1)
  2. Describe synchronization with respect to multi-threading.
  3. What are the different ways of implementing thread? Which one is more advantageous?
  4. What if the main() method is declared as private? What happens when the static modifier is removed from the signature of the main() method?
  5. Error & Exception

**Web Component Basics**

* 1. What is web container / ejb container.
  2. Describe the Life-cycle of a Java Servlet

Java Servlet life cycle is managed by the servlet container. When first web request comes in, the container will load the Servlet class, calls its init method, then calls its service method to process the web request. It says there can be only one instance of servlet class. The container creates multiple threads and manages these threads to process multiple web requests (this is what I know from my knowledge). But I want to understand, how multiple threads are running and processing multiple simultaneous web requests given that there is only one instance of the servlet class.

**Answer :** An object instance('s methods) can be called simultaneously by more than one thread. This isn't specific for servlets though and is true in general.

What happens when this occurs? Each thread will still have their own stack, which means, each thread will have a different copy of local variables to work on. As a result there will be no interference between threads and you won't have to worry about concurrent calls. Only when a shared resource, e.g. instance/class variable is accessed, there can be a problem. Same thing if instance/class variable is directly accessed concurrently.

EJBs in contrast do exactly what you seem to suggest. The EJB container makes sure that only one thread enters an EJB instance at a time, and hence an EJB programmer doesn't have to worry about concurrency so long as he/she doesn't break the EJB programming contract. There is no reason why the servlet spec didn't do this. Most likely no body came up with it during the meetings? It does have the advantage though that you can use a more efficient concurrency management than EJB's "one thread per instance".