<http://candidjava.com/spring-framework>

**Hibernate Interview Question**

**1.            What is the general flow of Hibernate communication with RDBMS?**

The general flow of Hibernate communication with RDBMS is :

* Load the Hibernate configuration file and create configuration object. It will automatically load all hbm mapping files
* Create session factory from configuration object
* Get one session from this session factory
* Create HQL Query
* Execute query to get list containing Java objects

**2.            What is Hibernate Query Language (HQL)**

Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

**3.            How do you map Java Objects with Database tables?**

* First we need to write Java domain objects (beans with setter and getter).
* Write hbm.xml, where we map java class to table and database columns to Java class variables.

**Example :**

|  |  |  |
| --- | --- | --- |
| 01 | &lt;hibernate-mapping&gt; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | &lt;class name="com.test.User"  table="user"&gt; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | &lt;property  column="USER\_NAME" length="255" | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | name="userName" not-null="true"  type="java.lang.String"/&gt; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &lt;property  column="USER\_PASSWORD" length="255" | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | name="userPassword" not-null="true"  type="java.lang.String"/&gt; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | &lt;/class&gt; | |
| 14 |  |

|  |  |
| --- | --- |
| 15 | &lt;/hibernate-mapping&gt; |

**4.            What’s load() vs. get() :-**

|  |  |
| --- | --- |
| **load()** | **get()** |
| Only use the load() method if you are sure that the object exists. | If you are not sure that the object exists, then use one of the get()methods. |
| load() method will throw an exception if the unique id is not found in the database. | get() method will return null if the unique id is not found in the database. |
| load() just returns a proxy by default and database wonâ€™t be hit until the proxy is first invoked | get() will hit the database immediately. |

Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

**5.            What is the difference between and merge and update ?**

Use update() if you are sure that the session does not contain an already persistent instance with the same identifier, and merge() if you want to merge your modifications at any time without consideration of the state of the session.

**6.            How do you define sequence generated primary key in hibernate?**

Using <generator> tag.

Example:-

<id column=”USER\_ID” name=”id” type=”java.lang.Long”>

<generator class=”sequence”>

<param name=”table”>SEQUENCE\_NAME</param>

<generator>

</id>

**7.            Define cascade and inverse option in one-many mapping?**

cascade – enable operations to cascade to child entities.

cascade=”all|none|save-update|delete|all-delete-orphan”

inverse – mark this collection as the “inverse” end of a bidirectional association.

inverse=”true|false”

Essentially “inverse” indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

**8.            What do you mean by Named-SQL query?**

Named SQL queries are defined in the mapping xml document and called wherever required.

Example:

<sql-query name = “empdetails”>

<return alias=”emp”/>

SELECT emp.EMP\_ID AS {emp.empid},

emp.EMP\_ADDRESS AS {emp.address},

emp.EMP\_NAME AS {emp.name}

FROM Employee EMP WHERE emp.NAME  LIKE :name

</sql-query>

Invoke Named Query :

List people = session.getNamedQuery(“empdetails”)

.setString(“TomBrady”, name)

.setMaxResults(50)

.list();

**9.            How do you invoke Stored Procedures?**

<sql-query name=”selectAllEmployees\_SP” callable=”true”>

<return alias=”emp”>

<return-property name=”empid” column=”EMP\_ID”/>

<return-property name=”name” column=”EMP\_NAME”/>

<return-property name=”address” column=”EMP\_ADDRESS”/>

{ ? = call selectAllEmployees() }

</return>

</sql-query>

**10.          Explain Criteria API**

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like “search” screens where there is a variable number of conditions to be placed upon the result set.

Example :

List employees = session.createCriteria(Employee.class)

.add(Restrictions.like(“name”, “a%”) )

.add(Restrictions.like(“address”, “Boston”))

.addOrder(Order.asc(“name”) )

.list();

**1.         Define HibernateTemplate?**

org.springframework.orm.hibernate.HibernateTemplate is a helper class which provides different methods for querying/retrieving data from the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

**2.         What are the benefits does HibernateTemplate provide?**

The benefits of HibernateTemplate are :

* HibernateTemplate, a Spring Template class simplifies interactions with Hibernate Session.
* Common functions are simplified to single method calls.
* Sessions are automatically closed.
* Exceptions are automatically caught and converted to runtime exceptions.

**3.         How do you switch between relational databases without code changes?**

Using Hibernate SQL Dialects , we can switch databases. Hibernate will generate appropriate hql queries based on the dialect defined.

**4.         If you want to see the Hibernate generated SQL statements on console, what should we do?**

In Hibernate configuration file set as follows:

<property name=”show\_sql”>true</property>

**5.         What are derived properties?**

The properties that are not mapped to a column, but calculated at runtime by evaluation of an expression are called derived properties. The expression can be defined using the formula attribute of the element.

**6.         What is the difference between sorted and ordered collection in hibernate?**

|  |  |
| --- | --- |
| **sorted collection** | **order collection** |
| A sorted collection is sorting a collection by utilizing the sorting features provided by the Java collections framework. The sorting occurs in the memory of JVM which running Hibernate, after the data being read from database using java comparator. | Order collection is sorting a collection by specifying the order-by clause for sorting this collection when retrieval. |
| If your collection is not large, it will be more efficient way to sort it. | If your collection is very large, it will be more efficient way to sort it . |

**7.         what are the Collection types in Hibernate ?**

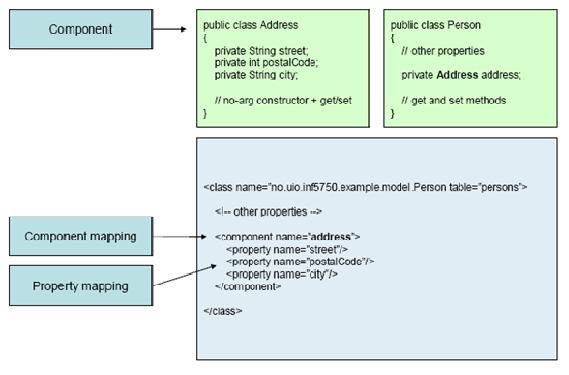
* Bag
* Set
* List
* Array
* Map

**8.         What is Hibernate proxy?**

The proxy attribute enables lazy initialization of persistent instances of the class. Hibernate will initially return CGLIB proxies which implement the named interface. The actual persistent object will be loaded when a method of the proxy is invoked.

**9.         What is component mapping in Hibernate?**

* A component is an object saved as a value, not as a reference
* A component can be saved directly without needing to declare interfaces or identifier properties
* Required to define an empty constructor
* Shared references not supported

**Example:[](http://candidjava.com/wp-content/uploads/2011/12/ComponentMap.jpg)**

**10.       What are the ways to express joins in HQL?**

HQL provides four ways of expressing (inner and outer) joins:-

* An implicit association join
* An ordinary join in the FROM clause
* A fetch join in the FROM clause.
* A theta-style join in the WHERE clause.

**1.         How can Hibernate be configured to access an instance variable directly and not through a setter method ?**

By mapping the property with access=”field” in Hibernate metadata. This forces hibernate to by

pass the setter method and access the instance variable directly while initializing a newly loaded object.

**2.         What is the advantage of Hibernate over jdbc?**

**Hibernate Vs. JDBC :-**

|  |  |
| --- | --- |
| **JDBC** | **Hibernate** |
| With JDBC, developer has to write code to map an object model’s data representation to a relational data model and its corresponding database schema. | Hibernate is flexible and powerful ORM solution to map Java classes to database tables. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this. |
| With JDBC, the automatic mapping of Java objects with database tables and vice versa conversion is to be taken care of by the developer manually with lines of code. | Hibernate provides transparent persistence and developer does not need to write code explicitly to map database tables tuples to application objects during interaction with RDBMS. |
| JDBC supports only native Structured Query Language (SQL). Developer has to find out the efficient way to access database, i.e. to select effective query from a number of queries to perform same task. | Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries. Hibernate also supports native SQL statements. It also selects an effective way to perform a database manipulation task for an application. |
| Application using JDBC to handle persistent data (database tables) having database specific code in large amount. The code written to map table data to application objects and vice versa is actually to map table fields to object properties. As table changed or database changed then itâ€™s essential to change object structure as well as to change code written to map table-to-object/object-to-table. | Hibernate provides this mapping itself. The actual mapping between tables and application objects is done in XML files. If there is change in Database or in any table then the only need to change XML file properties. |
| With JDBC, it is developerâ€™s responsibility to handle JDBC result set and convert it to Java objects through code to use this persistent data in application. So with JDBC, mapping between Java objects and database tables is done manually. | Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance if client application reads same data many times for same write. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code. |
| In JDBC there is no check that always every user has updated data. This check has to be added by the developer. | Hibernate enables developer to define version type field to application, due to this defined field Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table. So if two users retrieve same tuple and then modify it and one user save this modified tuple to database, version is automatically |

**3.         How can a whole class be mapped as immutable?**

Mark the class as mutable=”false” (Default is true),. This specifies that instances of the class are (not) mutable. Immutable classes, may not be updated or deleted by the application.

**4.         What is the use of dynamic-insert and dynamic-update attributes in a class mapping?**

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like “search” screens where there is a variable number of conditions to be placed upon the result set.

•           dynamic-update (defaults to false): Specifies that UPDATE SQL should be generated at runtime and contain only those columns whose values have changed.

•           dynamic-insert (defaults to false):Specifies that INSERT SQL should be generated at runtime and contain only the columns whose values are not null.

**5.         What do you mean by fetching strategy ?**

fetching strategy is the strategy Hibernate will use for retrieving associated objects if the application needs to navigate the association. Fetch strategies may be declared in the O/R mapping metadata, or over-ridden by a particular HQL or Criteria query.

**6.         What is automatic dirty checking?**

Automatic dirty checking is a feature that saves us the effort of explicitly asking Hibernate to update the database when we modify the state of an object inside a transaction.

**7.         What is transactional write-behind?**

Hibernate uses a sophisticated algorithm to determine an efficient ordering that avoids database foreign key constraint violations but is still sufficiently predictable to the user. This feature is called transactional write-behind.

**8.         What are Callback interfaces?**

Callback interfaces allow the application to receive a notification when something interesting happens to an object for example, when an object is loaded, saved, or deleted. Hibernate applications don’t need to implement these callbacks, but they’re useful for implementing certain kinds of generic functionality.

**9.         What are the types of Hibernate instance states ?**

Three types of instance states:

•           Transient -The instance is not associated with any persistence context

•           Persistent -The instance is associated with a persistence context

•           Detached -The instance was associated with a persistence context which has been closed â€“ currently not associated

**10.       What are the differences between EJB 3.0 & Hibernate?**

**Hibernate Vs EJB 3.0 :-**

|  |  |
| --- | --- |
| **Hibernate** | **EJB 3.0** |
| Session- Cache or collection of loaded objects relating to a single unit of work | Persistence Context-Set of entities that can be managed by a given EntityManager is defined by a persistence unit |
| XDoclet Annotations used to support Attribute Oriented Programming | Java 5.0 Annotations used to support Attribute Oriented Programming |
| Defines HQL for expressing queries to the database | Defines EJB QL for expressing queries |
| Supports Entity Relationships through mapping files and annotations in JavaDoc | Support Entity Relationships through Java 5.0 annotations |
| Provides a Persistence Manager API exposed via the Session, Query, Criteria, and Transaction API | Provides and Entity Manager Interface for managing CRUD operations for an Entity |
| Provides callback support through lifecycle, interceptor, and validatable interfaces | Provides callback support through Entity Listener and Callback methods |
| Entity Relationships are unidirectional. Bidirectional relationships are implemented by two unidirectional relationships | Entity Relationships are bidirectional or unidirectional |

**1.            What is ORM?**

ORM stands for object/relational mapping. ORM is the automated persistence of objects in a Java application to the tables in a relational database

**2.            What does ORM consists of?**

An ORM solution consists of the following four pieces:

API for performing basic CRUD operations

API to express queries refering to classes

Facilities to specify metadata

Optimization facilities :

Dirty checking,lazy associations fetching

**3.            Why do you need ORM tools like hibernate?**

The main advantage of ORM like hibernate is that it shields developers from messy SQL. Apart from this,

ORM provides following benefits:

Improved productivity

High-level object-oriented API

Less Java code to write

No SQL to write

Improved performance

Sophisticated caching Lazy loading Eager loading Improved maintainability

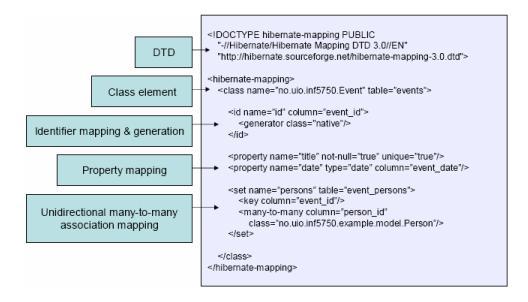
A lot less code to write Improved portability

ORM framework generates database-specific SQL for you

**4.            What does Hibernate simplify?**

Hibernate simplifies: Saving and retrieving your domain objects Making database column and table name changes Centralizing pre save and post retrieve logic Complex joins for retrieving related items Schema creation from object model

**5.            What is the  need for hibernate xml mapping file?**

[](http://candidjava.com/wp-content/uploads/2011/12/Hibernatemapping.jpg)

Hibernate mapping file tells Hibernate which tables and columns to use to load and store objects. Typical mapping file look as follows:

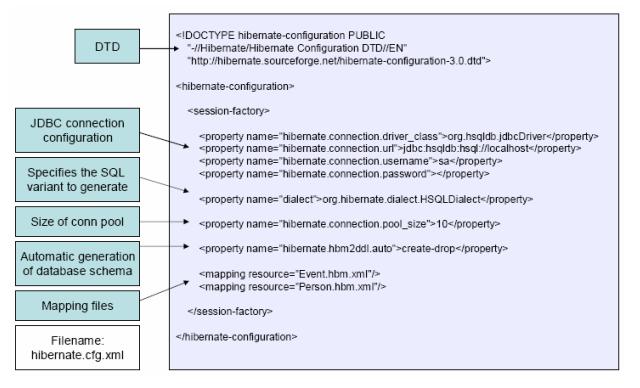
**6.            What are the most common methods of hibernate configuration?**

The most common methods of Hibernate configuration are:

Programmatic configuration XML configuration (hibernate.cfg.xml)

**7.            What are the important tags of hibernate .cfg.xml?**

Following are the important tags of hibernate.cfg.xml:

[](http://candidjava.com/wp-content/uploads/2011/12/HibernateTags.jpg)

**8.            What are the Core interface are of hibernate framework?**

The five core interfaces are used in just about every Hibernate application. Using these interfaces, you can store and retrieve persistent objects and control transactions. Session interface SessionFactory interface Configuration interface Transaction interface Query and Criteria interfaces

**9.            What role does the Session interface play in hibernate?**

The Session interface is the primary interface used by Hibernate applications. It is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It allows you to create query objects to retrieve persistent objects.

Session session = sessionFactory.openSession();

Session interface role: Wraps a JDBC connection Factory for Transaction Holds a mandatory (first-level) cache of persistent objects, used when navigating the object graph or looking up objects by identifier.

**10.          What role does the session Factory interface play in hibernate?**

The application obtains Session instances from a SessionFactory. There is typically a single SessionFactory for the whole application created during application initialization. The SessionFactory caches generate SQL statements and other mapping metadata that Hibernate uses at runtime. It also holds cached data that has been read in one unit of work and may be reused in a future unit of work

SessionFactory sessionFactory = configuration.buildSessionFactory();

**1.            What is a Session Factory? Is it a thread-safe object?**

SessionFactory is Hibernates concept of a single datastore and is threadsafe so that many threads can access it concurrently and request for sessions and immutable cache of compiled mappings for a single database. A SessionFactory is usually only built once at startup. SessionFactory should be wrapped in some kind of singleton so that it can be easily accessed in an application code.

SessionFactory sessionFactory = new Configuration().configure().buildSessionfactory();

**2.            What is a Session? Can you share a session object between different thread?**

Session is a light weight and a non-threadsafe object (No, you cannot share it between threads) that represents a single unit-of-work with the database. Sessions are opened by a SessionFactory and then are closed when all work is complete. Session is the primary interface for the persistence service. A session obtains a database connection lazily (i.e. only when required). To avoid creating too many sessions ThreadLocal class can be used as shown below to get the current session no matter how many times you make call to the currentSession() method.

|  |  |  |
| --- | --- | --- |
| 01 | &amp; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | public class HibernateUtil { | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | &amp; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | public static final ThreadLocal local = new ThreadLocal(); | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &nbsp; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | public static Session currentSession() throws HibernateException { | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | Session session = (Session) local.get(); | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | //open a new session if this thread has no session | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | if(session == null) { | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | session = sessionFactory.openSession(); | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | local.set(session); | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | } | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | return session; | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | } | |
| 28 |  |

|  |  |
| --- | --- |
| 29 | } |

It is also vital that you close your session after your unit of work completes. Note: Keep your Hibernate Session API handy.

**3.            What are the benefits of detached objects?**

Detached objects can be passed across layers all the way up to the presentation layer without having to use any DTOs (Data Transfer Objects). You can later on re-attach the detached objects to another session.

**4.            How does Hibernate distinguish between transient(i.e. newly instantiated ) and detached object?**

” Hibernate uses the versionproperty, if there is one.

” If not uses the identifier value. No identifier value means a new object. This does work only for Hibernate managed surrogate keys. Does not work for natural keys and assigned (i.e. not managed by Hibernate) surrogate keys.

” Write your own strategy with Interceptor.isUnsaved().

**5.            What are the pros and cons of detached objects?**

Pros:

” When long transactions are required due to user think-time, it is the best practice to break the long transaction up into two or more transactions. You can use detached objects from the first transaction to carry data all the way up to the presentation layer. These detached objects get modified outside a transaction and later on re-attached to a new transaction via another session.

Cons

” In general, working with detached objects is quite cumbersome, and better to not clutter up the session with them if possible. It is better to discard them and re-fetch them on subsequent requests. This approach is not only more portable but also more efficient because – the objects hang around in Hibernate’s cache anyway.

” Also from pure rich domain driven design perspective it is recommended to use DTOs (DataTransferObjects) and DOs (DomainObjects) to maintain the separation between Service and UI tiers.

**6.            What are the general considerations or best practices for defining your Hibernate persistent classes?**

* You must have a default no-argument constructor for your persistent classes and there should be getXXX() (i.e accessor/getter) and setXXX( i.e. mutator/setter) methods for all your persistable instance variables.
* You should implement the equals() and hashCode() methods based on your business key and it is important not to use the id field in your equals() and hashCode() definition if the id field is a surrogate key (i.e. Hibernate managed identifier). This is because the Hibernate only generates and sets the field when saving the object.
* It is recommended to implement the Serializable interface. This is potentially useful if you want to migrate around a multi-processor cluster.

* The persistent class should not be final because if it is final then lazy loading cannot be used by creating proxy objects.
* Use XDoclet tags for generating your \*.hbm.xml files or Annotations (JDK 1.5 onwards), which are less verbose than \*.hbm.xml files.

**7.            Explain about the transparent presistence of Hibernate?**

Transparent persistence is provided for Plain old Java objects or POJOs. For proper functioning of theapplications importance should be given to the methods equals () and hash Code methods (). It has a requirement which should be strictly followed in the applications which is a no-argument constructor.

**8.            Explain about the Primary feature of Hibernate?**

Primary feature of hibernate is to java classes to database tables. Data query and retrieval is also possible with Hibernate. Application portability is a key feature in Hibernate it allows developers to port applicationsto almost all SQL databases

**9.            Explain about transaction file?**

Transactions denote a work file which can save changes made or revert back the changes.A transaction can be started by session.beginTransaction() and it uses JDBC connection, CORBA or JTA. When this session starts several transactions may occur.

**10.          What is the effect when a transient mapped object is passed onto a sessions save?**

When a Sessions save () is passed to a transient mapped object it makes the method to become more persistent. Garbage collection and termination of the Java virtual machine stays as long as it is deleted explicitly. It may head back to its transient state.

**1.            Explain about version field?**

Application level data integrity constants are important if you are making changes to offline information which is again backed by database. Higher level locking or versioning protocol is required to support them. Version field usage comes at this stage but the design and implementation process is left to the developer.

**2.            Explain about addClass function?**

This function translates a Java class name into file name. This translated file name is then loaded as an input stream from the Java class loader. This addclass function is important if you want efficient usage of classes in your code.

**3.            Expalin about addjar() and adddirectory() methods?**

These methods are the most convenient to use in hibernate. These methods allow you to load all your Hibernate documents at a time. These methods simplify code configuration, refactoring, layout, etc. These functions help you to add your hibernate mapping to Hibernate initialization files.

**4.            Explain about the id field?**

The id field corresponds to the surrogate key which is generated by the database. These fields are handled by the id field. Name attribute is used to specify the names of the field and it should correspond to the method name of getid. This also should correspond to long type and the values should be stored I the database in the long column.

**5.            What is lazy initialization in hibernate?**

Lazy setting decides whether to load child objects while loading the Parent Object. You need to specify parent class.Lazy = true in hibernate mapping file. By default the lazy loading of the child objects is true. This make sure that the child objects are not loaded unless they are explicitly invoked in the application by calling getChild() method on parent. In this case hibernate issues a fresh database call to load the child when getChild() is actually called on the Parent object. But in some cases you do need to load the child objects when parent is loaded. Just make the lazy=false and hibernate will load the child when parent is loaded from the database. Examples: Address child of User class can be made lazy if it is not required frequently. But you may need to load the Author object for Book parent whenever you deal with the book for online bookshop.

Hibernate does not support lazy initialization for detached objects. Access to a lazy association outside of the context of an open Hibernate session will result in an exception.

**6.            What is DAO ?**

The Java Data Access Object (Java DAO) is an important component in business applications. Business applications almost always need access to data from relational or object databases and the Java platform offers many techniques for accessingthis data. The oldest and most mature technique is to use the Java Database Connectivity (JDBC)API, which provides the capability to execute SQL queries against a databaseand then fetch the results, one column at a time. Although this API provideseverything a developer needs to access data and to persist application state,it is a cumbersome API to develop against – which makes a Java DAO code generator particularly useful.

**7.            Explain the important feature of DAO ?**

The Data Access Object design pattern provides a technique for separating object persistence and data access logic from any particular persistencemechanism or API. There are clear benefits to this approach from anarchitectural perspective. The Java DAO approach provides flexibility to change anapplication’s persistence mechanism over time without the need to re-engineerapplication logic that interacts with the Data Access Object tier. For example, there may beperformance benefits in changing an application’s performance mechanism fromusing Entity Beans to using direct JDBC calls from a session bean, or even amove to an alternative persistence framework, such as Hibernate. Without a Java DAO tierin place, this sort of transition would require extensive re-engineering ofexisting code.

**8.            What is connection pooling?**

Connection pooling is a technique of creating and managing a pool of connections that are ready for use by any thread that needs them.

This technique of pooling connections is based on the fact that most applications only need a thread to have access to a JDBC connection when they are actively processing a transaction, which usually take only milliseconds to complete. When not processing a transaction, the connection would otherwise sit idle. Instead, connection pooling allows the idle connection to be used by some other thread to do useful work.

In practice, when a thread needs to do work against a MySQL or other database with JDBC, it requests a connection from the pool. When the thread is finished using the connection, it returns it to the pool, so that it may be used by any other threads that want to use it.

When the connection is loaned out from the pool, it is used exclusively by the thread that requested it. From a programming point of view, it is the same as if your thread called DriverManager.getConnection() every time it needed a JDBC connection, however with connection pooling, your thread may end up using either a new, or already-existing connection.

**9.            What is the necessity of connection polling?**

Benefits of Connection Pooling:

Connection pooling can greatly increase the performance of your Java application, while reducing overall resource usage. The main benefits to connection pooling are:

Reduced connection creation time:

While this is not usually an issue with the quick connection setup that MySQL offers compared to other databases, creating connections still has networking and JDBC driver overhead that will be avoided if connections are recycled.

Simplified programming model:

When using connection pooling, each individual thread can act as though it has created its own JDBC connection, allowing you to use straight-forward JDBC programming techniques.

Controlled resource usage:

If you don’t use connection pooling, and instead create a new connection every time a thread needs one, your application’s resource usage can be quite wasteful and lead to unpredictable behavior under load.

Remember that each connection to MySQL has overhead (memory, CPU, context switches, etc) on both the client and server side. Every connection limits how many resources there are available to your application as well as the MySQL server. Many of these resources will be used whether or not the connection is actually doing any useful work!

Connection pools can be tuned to maximize performance, while keeping resource utilization below the point where your application will start to fail rather than just run slower.

**10.          What is QBECK (que) QC(qbc)?**

Quality Center is a comprehensive test management tool. It is a web-based tool and supports high level of communication and association among various stakeholders (Business Analyst, Developers , Testers etc. ) , driving a more effective and efficient global application-testing process. Automation Tools like QTP , WinRunner & LoadRunner can be integrated with Quality Center. One can also create reports and graphs for Analysis and Tracking for Test processes.

**1.            What is the purpose of dialect?**

There are two concepts here a Dialect and a Driver.

Driver is like English.

Dialect is the different pronunciations of English.

We all know there are different versions of Oracle… Oracle 9i, Oracle8.

The driver we would use would be a common for all of these.

But the dialect we use is specific to each one of them, which helps Hibernate in generating optimized queries to those specific versions of database.

Also not that this is not mandatory to be given in cfg.xml.

This is the SQL dialect (database type) for the database being used. Any valid Hibernate dialect may be used. A few the more commonly used dialects are:

* net.sf.hibernate.dialect.HSQLDialect
* net.sf.hibernate.dialect.Oracle9Dialect
* net.sf.hibernate.dialect.MySQLDialect
* net.sf.hibernate.dialect.SQLServerDialect
* net.sf.hibernate.dialect.FirebirdDialect

**2.            How to create the table dynamically in hibernate?**

I have managed to setup an application that inserts, retrives, and delete objects in a database. I have also instructed hibernate to update the set of tables when ever it restarts (with hibernate.hbm2ddl.auto=update)

**3.            What is native SQL ?**

They are loosely integrated into ABAP. It allows access to all functions containing programming interface. They are not checked and converted. They are sent directly to the database system. Programs that use Native SQL are specific to the database system for which they were written. For e.g. to create or change table definition in the ABAP.

**4.            What are the different types of mapping?**

Hibernate built-in mapping types usually share the name of the Java type they map; however, there may be more than one Hibernate mapping type for a partucular Java type. The various built-in-mapping types available in Hibernate are:

* Java primitive mapping types
* Date and time mapping types
* Large object mapping types
* Various JDK mapping types

**5.            What is ACID properties?**

The ACID  properties are:

* Atomicity
* Consistency
* Isolation
* Durability

Atomicity – A transaction must be an atomic unit of work; either all of its data modifications are performed or none of them is performed.

Consistency – When completed, a transaction must leave all data in a consistent state. In a relational database, all rules must be applied to the transaction’s modifications to maintain all data integrity. All internal data structures, such as B-tree indexes or doubly-linked lists, must be correct at the end of the transaction.

Isolation – Modifications made by concurrent transactions must be isolated from the modifications made by any other concurrent transactions. A transaction either sees data in the state it was in before another concurrent transaction modified it, or it sees the data after the second transaction has completed, but it does not see an intermediate state. This is referred to as serializability because it results in the ability to reload the starting data and replay a series of transactions to end up with the data in the same state it was in after the original transactions were performed.

Durability – After a transaction has completed, its effects are permanently in place in the system. The modifications persist even in the event of a system failure.

**6.            What is transaction?**

A transaction is a set of rows bound by a commit or rollback of rows. The transaction control transformation is used to commit or rollback a group of rows.

**7.            Types of transaction:**

Long run transaction

Short run transaction

**8.            What is isolation levels?**

The following are the transaction levels or built-in variables:

* TC\_CONTINUE\_TRANSACTION: The Integration Service does not perform any transaction change for this row. This is the default value of the expression.
* TC\_COMMIT\_BEFORE: The Integration Service commits the transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.
* TC\_COMMIT\_AFTER: The Integration Service writes the current row to the target, commits the transaction, and begins a new transaction. The current row is in the committed transaction.
* TC\_ROLLBACK\_BEFORE: The Integration Service rolls back the current transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.
* TC\_ROLLBACK\_AFTER: The Integration Service writes the current row to the target, rolls back the transaction, and begins a new transaction. The current row is in the rolled back transaction.

**9.            What is lock mode?**

setLockMode is a method of Hibernate Criteria API. You can use a lock mode in the following way

|  |  |  |
| --- | --- | --- |
| 1 | Criteria criteria = session.createCriteria(Student.class); | |
| 2 |  |

|  |  |  |
| --- | --- | --- |
| 3 | criteria.add(Restrictions.eq("name", "Ramesh")); | |
| 4 |  |

|  |  |  |
| --- | --- | --- |
| 5 | criteria.setLockMode("this", LockMode.UPGRADE); | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | List list = criteria.list(); |

**10.          Difference between session.load and session.lock**

The load() method is older; get() was added to Hibernate’s API due to user request. The difference is trivial:

The following Hibernate code snippet retrieves a User object from the database:

User user = (User) session.get(User.class, userID);

The get() method is special because the identifier uniquely identifies a single instance of a class. Hence itâ€™s common for applications to use the identifier as a convenient handle to a persistent object. Retrieval by identifier can use the cache when retrieving an object, avoiding a database hit if the object is already cached.

Hibernate also provides a load() method:

User user = (User) session.load(User.class, userID);

If load() can’t find the object in the cache or database, an exception is thrown. The load() method never returns null. The get() method returns

null if the object can’t be found. The load() method may return a proxy instead of a real persistent instance. A proxy is a placeholder instance of a runtime-generated subclass (through cglib or Javassist) of a mapped persistent class, it can initialize itself if any method is called that is not the mapped database identifier getter-method. On the other hand, get() never returns a proxy. Choosing between get() and load() is easy: If youâ€™re certain the persistent object exists, and nonexistence would be considered exceptional, load() is a good option. If you arenâ€™t certain there is a persistent instance with the given identifier, use get() and test the return value to see if itâ€™s null. Using load() has a further implication: The application may retrieve a valid reference (a proxy) to a persistent instance without hitting the database to retrieve its persistent state. So load() might not throw an exception when it doesnâ€™t find the persistent object in the cache or database; the exception would be thrown later, when the proxy is accessed.

OR

Hibernate Interview Questions and Answers will guide us now that Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high\_level object handling functions. Learn more about Hibernate bu this Hibernate Interview Questions with Answers guide Select More Questions to explore you knowledge regarding Hibernate Job Interview Questions and answers. Explore more thousands of professional Hibernate job interview questions and answers with us.

**1.            what is caching?**

Anything you can do to minimize traffic between a database and an application server is probably a good thing. In theory, an application ought to be able to maintain a cache containing data already loaded from the database, and only hit the database when information has to be updated. When the database is hit, the changes may invalidate the cache

**2.            Types of caching ?**

* First-Level Cache
* Second-Level Cache

**3.            What are the different options for hibernate second level of caching**

Second-level cache always associates with the Session Factory object. While running the transactions, in between it loads the objects at the Session Factory level, so that those objects will available to the entireapplication, donâ€™t bounds to single user. Since the objects are already loaded in the cache, whenever an object is returned by the query, at that time no need to go for a database transaction. In this way the second level cache works. Here we can use query level cache also. Later we will discuss about it.

To activate second-level caching, you need to define the hibernate.cache.provider\_class property in the hibernate.cfg.xml file as follows:

< hibernate-configuration >

< session-factory >

< property name=”hibernate.cache.provider\_class” >org.hibernate.cache.EHCacheProvider< / property>

< / session-factory >

< / hibernate-configuration >

By default, the second-level cache is activated and uses the EHCache provider.

To use the query cache you must first enable it by setting the property hibernate.cache.use\_query\_cache to true in hibernate.properties.

**4.            Explain the process nof achieving second level caching**

Second-level cache always associates with the Session Factory object. While running the transactions, in between it loads the objects at the Session Factory level, so that those objects will available to the entire application, donâ€™t bounds to single user. Since the objects are already loaded in the cache, whenever an object is returned by the query, at that time no need to go for a database transaction. In this way the second level cache works. Here we can use query level cache also. Later we will discuss about it.

**5.            What is evict and close?**

Hibernate does a neat trick where it tracks objects that have been changes since being loaded from the database. Once Hibernate provides an object, make a modification and flush the Session â€“ Hibernate knows the object was changed and will propagate to the database accordingly.

**6.            what are managed associations and hibernate associations**

Associations that are related to container management persistence are called managed associations. These are bi-directional associations. Coming to hibernate associations, these are unidirectional

**7.            What are the different approaches to represent an inheritance hierarchy?**

* Table per concrete class.
* Table per class hierarchy.
* Table per subclass.

**8.            What are the different methods of identifying an object?**

There are three methods by which an object can be identified.

* Object identity – Objects are identical if they reside in the same memory location in the JVM. This can be checked by using the = = operator.
* Object equality – Objects are equal if they have the same value, as defined by the equals( ) method. Classes that donâ€™t explicitly override this method inherit the implementation defined by java.lang.Object, which compares object identity.
* Database identity – Objects stored in a relational database are identical if they represent the same row or, equivalently, share the same table and primary key value.

**9.            What is Attribute Oriented Programming?**

XDoclet has brought the concept of attribute-oriented programming to Java. Until JDK 1.5, the Java language had no support for annotations; now XDoclet uses the Javadoc tag format (@attribute) to specify class-, field-, or method-level metadata attributes. These attributes are used to generate hibernate mapping file automatically when the application is built. This kind of programming that works on attributes is called as Attribute Oriented Programming.

**10.          What are the different types of property and class mappings?**

* Typical and most common property mapping

<property name=”description” column=”DESCRIPTION” type=”string”/>

Or

<property name=”description” type=”string”>

<column name=”DESCRIPTION”/>

</property>

* Derived properties

<property name=”averageBidAmount” formula=”( select AVG(b.AMOUNT) from BID b where b.ITEM\_ID = ITEM\_ID )” type=”big\_decimal”/>

* Typical and most common property mapping

<property name=”description” column=”DESCRIPTION” type=”string”/>

* Controlling inserts and updates

<property name=”name” column=”NAME” type=”string” insert=”false” update=”false”/>

**1.            Types of locks**

* Read lock
* Write lock
* Execute lock
* Fetch lock
* Save lock

**2.            What is 2-phase lock ?**

* Read only transaction
* Update transaction

**3.            What are the design patterens used in hibernate?**

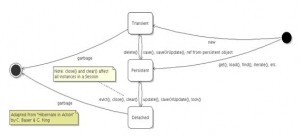
* Composite design
* DAO Design Pattern
* Abstract FactoryMVC
* Data Transfer
* Proxy
* facade

**4.            Whether POJO is java bean or not ?**

S . POJO is java beanâ€¦

POJO stands for plain old java objects. These are just basic JavaBeans that have defined setter and getter methods for all the properties that are there in that bean. Besides they can also have some business logic related to that property. Hibernate applications works efficiently with POJOs rather then simple java classes.

**5.            Please draw object life cycle in hibernate?**

[](http://candidjava.com/wp-content/uploads/2011/12/ObjectLife.jpg)Transient objects do not (yet) have any association with the database. they act like any normal Java object and are not saved to the database. When the last reference to a transient object is lost, the object itself is lost and is (eventually) garbage collected. There is no connection between transactions and such objects: commits and rollbacks have no effects on them. They can be turned into persistent objects via one of thesave method calls if the Session object or by adding a reference from a persistent object to this object.

Persistent objects do have an association with the database. They are always associated with a persistence manager, i.e., a Session object and they always participate in a transaction. Actual updates of a database from the persistent object may occur at any time between when the object is updated to the end of the transaction: it does not necessarily happen immediately. However, this feature, which allows important optimizations in database interactions, is essentially invisible to the programmer. For example, one place where one might expect to notice the difference between the in-memory persistent object and the database version is at the point of executing a query. In such a case, Hibernate will, if necessary, synchronise any dirty objects with the database (i.e., save them) in order to ensure that the query returns the correct results.

A persistent object has a primary key value set, whether or not it has been actually saved to the database yet.

Calling the delete method of the Session object on a persistent object will cause its removal from the database and will make it transient.

Detached objects are objects that were persistent but no longer have a connection to a Session object (usually because you have closed the session). Such an object contains data that was synchronised with the database at the time that the session was closed, but, since then, the database may have changed; with the result that this object is now stale.

**6.            How transaction management is done in hibernate?**

Transaction simply means a unit of work, which is atomic. If only one step fails, then the whole unit of work fails.  When we consider database, Transaction groups a set of statements/commands which gets committed together. If a single statement fails, whole work will be rolled back.   Transactions can be described using ACID criteria.

ACID means:

A: Atomicity:  In a Transaction If only one step fails, the whole unit of work must fail.This is known as atomicity

C : Consistency : The Transactions operate on Consistent data. I.e. This data is hidden from other concurrently running transactions. The data is clean and consistent.

I: Isolation: It means when a transaction is being executed, it will not be visible to other concurrently running transactions. This means, any changes done to the existing data during a transaction, before transaction finishes, is not visible to other active transactions. They work on consistent  data.

D: Durable: Also the changes done in a transaction are durable. Even if server / system fails after a transaction, the changes done by a successful transaction are permanent / persistent.

So a database transactions involve a set of SQL statements,  which either succeeds or fails. If any one of the statement fails in between,  the execution of subsequent statements are aborted and all changes done by the previous SQL statements will be rolled back.  In complex applications, which involve different database actions, one should set boundries for a transaction. This means beginning and end of the transaction should be decided. This is called Transaction demarcation. If an error occurs (either while executing operations or when committing the transaction), you have to roll back the transaction to leave the data in a consistent state.

This can be done in 2 ways.

In a programmatic manner by explicitly setting boundaries in code or using JTA  API.

Setting Transaction boundaries by declarative manner, specifying transaction boundaries to managed containers like EJB container

**7.            difference between association and component mapping?**

Association mapping refers to a many-to-one or one-to-one relationship which will be mapped by using another class which you have mapped in Hibernate (also called an “entity”). The associated object has its own lifecycle and is simply related to the first object.

Component mapping refers to mapping a class (or collection of classes) whose lifecycle is bound tightly to the parent. This is also called “composition” in the strict definition of the word in object-oriented programming. Basically if you delete the parent object the child object should also be deleted; it also cannot exist on its own without a parent.

**8.            Why hibernate is called as lazy invocation ?**

Lazy setting decides whether to load child objects while loading the Parent Object. You need to specify parent class.Lazy = true in hibernate mapping file. By default the lazy loading of the child objects is true. This make sure that the child objects are not loaded unless they are explicitly invoked in the application by calling getChild() method on parent. In this case hibernate issues a fresh database call to load the child when getChild() is actually called on the Parent object. But in some cases you do need to load the child objects when parent is loaded. Just make the lazy=false and hibernate will load the child when parent is loaded from the database. Examples: Address child of User class can be made lazy if it is not required frequently. But you may need to load the Author object for Book parent whenever you deal with the book for online bookshop.

Hibernate does not support lazy initialization for detached objects. Access to a lazy association outside of the context of an open Hibernate session will result in an exception.

**9.            What  is hibernate tuning?**

The key to obtain better performance in any hibernate application is to employ SQL Optimization, session management, and Data caching

**10.          Define hibernate statistics ?**

We’ve been doing a lot of Hibernate work at Terracotta recently and that naturally includes a fair amount of performance testing. In Hibernate, you can grab those stats using the Statistics object for aSessionFactory via getStatistics(). There are all sorts of tasty morsels inside this class to get factory-wide counts and per-entity/query/cache/etc stats. Cool stuff.

We noticed however while doing some perf testing that the code inside the StatisticsImpl is a bit problematic from a concurrency point of view. The basic gist of the class is just a set of stat counters. I can simplify the pattern to this without much loss of detail:

|  |  |  |
| --- | --- | --- |
| 01 | public class StatisticsImpl implements Statistics, StatisticsImplementor { | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | private long queryExecutionCount; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | public synchronized void queryExecuted(String hql, int rows, long time) { | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | queryExecutionCount++; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | // ... bunch of other stat collection | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | } | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | public long getQueryExecutionCount() { | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | return queryExecutionCount; | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | } | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | public synchronized void clear() { | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | queryExecutionCount = 0; | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | // ... clear all other stats | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | } | |
| 26 |  |

|  |  |
| --- | --- |
| 27 | } |

That’s basically what’s in this class but it’s repeated a couple dozen times for other counters and there are Maps used for per-item counts of a few kinds of items. There are several problems here from a concurrency point of view:

* Coarse-grained lock – the entire class shares a single lock via synchronized methods. That means that every counter in every thread is contending for that same lock. The impact of this is that when you turn on statistics collection in Hibernate, you immediately introduce a big old stop-the-world contention point across all threads. This will have an impact (possibly a very significant one) on the very stats you are trying to collect. It€™s entirely possible that the scale of your application or other bottlenecks mean that your application is not actually seeing this as a bottleneck, but it should scare you at least a little.

At the very least here, we could be used fine-grained locking to avoid creating contention between different kinds of statistics. There are also collections in here that collect stats on a per-query, per-entity, per-collection, etc basis. Those could actually have fine-grained locks per-entity as well (but they don€™t).

* Dirty reads -you’ll notice that while writes to the counters are synchronized, the reads are not. Presumably this was done for performance. Unfortunately, it’s also quite broken from a memory model point of view. These reads are not guaranteed to see writes from any other thread, so the values you’re seeing are possibly stale. In fact, it’s possible that you are never seeing any of the counter updates. In practice, the synchronization on puts is probably causing the local caches to get flushed and on the hardware we’re running, you do seem to see values that are in the ballpark at least. But the Java memory model makes no guarantee that this will work on all architectures or at any point in the future.
* Race condition on clear() -the common way that the stats are used is with some gui or other monitor sitting in a loop and periodically reading some (or all) of the stats, then calling clear(). Because time passes between the read of the first stat and the clear, you will lose all updates to the stats during the course of the reads.You may be willing to neglect a few lost updates, but consider that in many cases the monitor thread may iterate through every entity, collection, and query updated since the last loop (potentially hundreds of reads). In the cases where per-item stats are looked up, the gets() are actually synchronized as well when finding the stat in a Map. Those gets are synchronized against all other puts happening in the system. So the scope of that “read all stats” part of the monitor code may actually be quite large and you will lose all updates made between the beginning of that and the clear(), which distorts the next set of stats to an unknown degree (more activity == more distortion).
* [UPDATE] Dirty long read -As Peter mentioned in the comments, the values being read here are longs and since longs and doubles are 64-bit values, dirty reads of them are not guaranteed to see atomic writes from other threads. So you could see different 32-bit chunks that were not written together. Reads/writes of shared doubles and longs should always be done with synchronized or volatile to address this issue.

I certainly understand that stats are seen as best-effort and that the existing Hibernate code supports pre-1.5 Java and does not have access to Java 5 concurrency utilities like Atomic classes and concurrent collections, but even so I think there are things that could be done here to make stats collection a lot more accurate and less invasive. Things like fine-grained or more concurrent locking (with volatile, AtomicInteger, or ReentrantReadWriteLock) would go a long way towards fixing visibility while increasing concurrency.

In our Terracotta integration, I suspect we will be using some byte-code instrumentation to clean up these classes (we already assume Java 1.5+ so that is a constraint that we are happy to break). As it is, we don’t currently trust the stats in the first place and second we are actually seeing the single lock showing up as a hot spot in performance tests.

I hope that the next version of Hibernate (which I think is dropping pre-1.5 support?) can make some improvements as well.

**1.            What is HQL?**

HQL stands for Hibernate Query Language. Hibernate allows the user to express queries in its own portable SQL extension and this is called as HQL. It also allows the user to express in native SQL.

**2.            What is object/relational mapping metadata?**

ORM tools require a metadata format for the application to specify the mapping between classes and tables, properties and columns, associations and foreign keys, Java types and SQL types. This information is called the object/relational mapping metadata. It defines the transformation between the different data type systems and relationship representations.

**3.            What are POJOs?**

POJO stands for plain old java objects. These are just basic JavaBeans that have defined setter and getter methods for all the properties that are there in that bean. Besides they can also have some business logic related to that

**4.            What should SessionFactory be placed so that it can be easily accessed?**

As far as it is compared to J2EE environment, if the SessionFactory is placed in JNDI then it can be easily accessed and shared between different threads and various components that are hibernate aware. You can set the SessionFactory to a JNDI by configuring a property hibernate.session\_factory\_name in the hibernate.properties file.

**5.            What does hibernate.properties file consist of?**

This is a property file that should be placed in application class path. So when the Configuration object is created, hibernate is first initialized. At this moment the application will automatically detect and read this hibernate.properties file.

* hibernate.connection.datasource = java:/comp/env/jdbc/AuctionDB
* hibernate.transaction.factory\_class =net.sf.hibernate.transaction.JTATransactionFactory
* hibernate.transaction.manager\_lookup\_class = net.sf.hibernate.transaction.JBossTransactionManagerLookup
* hibernate.dialect = net.sf.hibernate.dialect.PostgreSQLDialect

**6.            What is meant by Method chaining?**

Method chaining is a programming technique that is supported by many hibernate interfaces. This is less readable when compared to actual java code. And it is not mandatory to use this format. Look how a SessionFactory is created when we use method chaining.

* SessionFactory sessions = new Configuration()
* .addResource(“myinstance/MyConfig.hbm.xml”)
* .setProperties( System.getProperties() )
* .buildSessionFactory();

**7.            What do you create a SessionFactory?**

* Configuration cfg = new Configuration();
* cfg.addResource(“myinstance/MyConfig.hbm.xml”);
* cfg.setProperties( System.getProperties() );
* SessionFactory sessions = cfg.buildSessionFactory();

**8.            What is the file extension you use for hibernate mapping file?**

The name of the file should be like this : filenam.hbm.xml

The filename varies here. The extension of these files should be”.hbm.xml”.

This is just a convention and it’s not mandatory. But this is the best practice to follow this extension.

**9.            What are different environments to configure hibernate?**

There are mainly two types of environments in which the configuration of hibernate application differs.

* Managed environment -In this kind of environment everything from database connections, transaction boundaries, security levels and all are defined. An example of this kind of environment is environment provided by application servers such as JBoss, Weblogic and WebSphere.
* Non-managed environment -This kind of environment provides a basic configuration template. Tomcat is one of the best examples that provide this kind of environment.

**10.          What are the Extension interfaces that are there in hibernate?**

There are many extension interfaces provided by hibernate.

* ProxyFactory interface – used to create proxies
* ConnectionProvider interface -used for JDBC connection management
* TransactionFactory interface – Used for transaction management
* Transaction interface -Used for transaction management
* TransactionManagementLookup interface- Used in transaction management.
* Cahce interface -provides caching techniques and strategies
* CacheProvider interface -same as Cache interface
* ClassPersister interface – provides ORM strategies
* IdentifierGenerator interface – used for primary key generation
* Dialect abstract class – provides SQL support

**1.            What are Extension interfaces?**

When the built-in functionalities provided by hibernate is not sufficient enough, it provides a way so that user can include other interfaces and implement those interfaces for user desire functionality. These interfaces are called as Extension interfaces.

**2.            What are Callback interfaces?**

These interfaces are used in the application to receive a notification when some object events occur. Like when an object is loaded, saved or deleted. There is no need to implement callbacks in hibernate applications, but they’re useful for implementing certain kinds of generic functionality.

**3.            What the Core interfaces are of hibernate framework?**

There are many benefits from these. Out of which the following are the most important one.

* Session Interface -This is the primary interface used by hibernate applications. The instances of this interface are lightweight and are inexpensive to create and destroy. Hibernate sessions are not thread safe.
* SessionFactory Interface -This is a factory that delivers the session objects to hibernate application. Generally there will be a single SessionFactory for the whole application and it will be shared among all the application threads.
* Configuration Interface -This interface is used to configure and bootstrap hibernate. The instance of this interface is used by the application in order to specify the location of hibernate specific mapping documents.
* Transaction Interface -This is an optional interface but the above three interfaces are mandatory in each and every application. This interface abstracts the code from any kind of transaction implementations such as JDBC transaction, JTA transaction.
* Query and Criteria Interface -This interface allows the user to perform queries and also control the flow of the query execution.

**4.            What is a hibernate xml mapping document and how does it look like?**

In order to make most of the things work in hibernate, usually the information is provided in an xml document. This document is called as xml mapping document. The document defines, among other things, how properties of the user defined persistence classes’map to the columns of the relative tables in database.

<?xml version=”1.0″?>

<!DOCTYPE hibernate-mapping PUBLIC ”

http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd”>

<hibernate-mapping>

<class name=”sample.MyPersistanceClass” table=”MyPersitaceTable”>

<id name=”id” column=”MyPerId”>

<generator/>

</id>

<property name=”text” column=”Persistance\_message”/>

<many-to-one name=”nxtPer” cascade=”all” column=”NxtPerId”/>

</class>

</hibernate-mapping>

**5.            How does hibernate code looks like?**

Session session = getSessionFactory().openSession();

Transaction tx = session.beginTransaction();

MyPersistanceClass mpc = new MyPersistanceClass (“Sample App”);

session.save(mpc);

tx.commit();

session.close();

**6.            What are the benefits of ORM and Hibernate?**

There are many benefits from these. Out of which the following are the most important one.

1.            Productivity -Hibernate reduces the burden of developer by providing much of the functionality and let the developer to concentrate on business logic.

2.            Maintainability -As hibernate provides most of the functionality, the LOC for the application will be reduced and it is easy to maintain. By automated object/relational persistence it even reduces the LOC.

3.            Performance -Hand-coded persistence provided greater performance than automated one. But this is not true all the times. But in hibernate, it provides more optimization that works all the time there by increasing the performance. If it is automated persistence then it still increases the performance.

Vendor independence -Irrespective of the different types of databases

**7.            What is meant by full object mapping?**

Full object mapping supports sophisticated object modeling: composition, inheritance, polymorphism and persistence. The persistence layer implements transparent persistence; persistent classes do not inherit any special base class or have to implement a special interface. Efficient fetching strategies and caching strategies are implemented transparently to the application.

**8.            What is a meant by medium object mapping?**

The application is designed around an object model. The SQL code is generated at build time. And the associations between objects are supported by the persistence mechanism, and queries are specified using an object-oriented expression language. This is best suited for medium-sized applications with some complex transactions. Used when the mapping exceeds 25 different database products at a time.

**9.            What is a meant by light object mapping?**

The entities are represented as classes that are mapped manually to the relational tables. The code is hidden from the business logic using specific design patterns. This approach is successful for applications with a less number of entities, or applications with common, metadata-driven data models. This approach is most known to all.

**10.          What is a pure relational ORM?**

The entire application, including the user interface, is designed around the relational model and SQL-based relational operations.

**1.            What are the different levels of ORM quality?**

There are four levels defined for ORM quality.

* Pure relational
* Light object mapping
* Medium object mapping
* Full object mapping

**2.            What is Hibernate?**

Hibernate is a powerful, high performance object/relational persistence and query service. This lets the users to develop persistent classes following object-oriented principles such as association, inheritance, polymorphism, composition, and collections.

**3.            What is the method we have to use for enabling second level caching in code?**

Hibernate offers naturally a first level cache for entities called a persistence context via the notion ofSession. This cache is contextual to the use case at hand. Some entities however are shared by many different use cases and are barely changed. You can cache these in what is called the second level cache.

By default, entities are not part of the second level cache. While we do not recommend that, you can override this by setting the shared-cache-mode element in your persistence.xml file or by using thejavax.persistence.sharedCache.mode property. The following values are possible:

* ENABLE\_SELECTIVE (Default and recommended value): entities are not cached unless explicitly marked as cacheable.
* DISABLE\_SELECTIVE: entities are cached unless explicitly marked as not cacheable.
* ALL: all entities are always cached even if marked as non cacheable.
* NONE: no entity are cached even if marked as cacheable. This option can make sense to disable second-level cache altogether.

The cache concurrency strategy used by default can be set with the hibernate.cache.default\_cache\_concurrency\_strategyproperty:

* read-only
* read-write
* nonstrict-read-write
* transactional

**4.            What is the hibernate filter?**

Hibernate3 provides an innovative new approach to handling data with “visibility” rules. AHibernate filteris a global, named, parameterized filter that can be enabled or disabled for a particular Hibernate session.

Hibernate3 has the ability to pre-define filter criteria and attach those filters at both a class level and a collection level. A filter criteria allows you to define a restriction clause similar to the existing “where” attribute available on the class and various collection elements. These filter conditions, however, can be parameterized. The application can then decide at runtime whether certain filters should be enabled and what their parameter values should be. Filters can be used like database views, but they are parameterized inside the application.

Example

Filter in Hibernate ——

USER ( ID INT, USERNAME VARCHAR, ACTIVATED BOOLEAN) – TABLE

|  |  |  |
| --- | --- | --- |
| 01 | &nbsp; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | public class User | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | { | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | private int id; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | private String username; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | private boolean activated; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | &nbsp; | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | &nbsp; | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | public boolean isActivated() | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | { | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | return activated; | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | } | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | public void setActivated(boolean activated) | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | { | |
| 28 |  |

|  |  |  |
| --- | --- | --- |
| 29 | this.activated = activated; | |
| 30 |  |

|  |  |  |
| --- | --- | --- |
| 31 | } | |
| 32 |  |

|  |  |  |
| --- | --- | --- |
| 33 | public int getId() | |
| 34 |  |

|  |  |  |
| --- | --- | --- |
| 35 | { | |
| 36 |  |

|  |  |  |
| --- | --- | --- |
| 37 | return id; | |
| 38 |  |

|  |  |  |
| --- | --- | --- |
| 39 | } | |
| 40 |  |

|  |  |  |
| --- | --- | --- |
| 41 | public void setId(int id) | |
| 42 |  |

|  |  |  |
| --- | --- | --- |
| 43 | { | |
| 44 |  |

|  |  |  |
| --- | --- | --- |
| 45 | this.id = id; | |
| 46 |  |

|  |  |  |
| --- | --- | --- |
| 47 | } | |
| 48 |  |

|  |  |  |
| --- | --- | --- |
| 49 | public String getUsername() | |
| 50 |  |

|  |  |  |
| --- | --- | --- |
| 51 | { | |
| 52 |  |

|  |  |  |
| --- | --- | --- |
| 53 | return username; | |
| 54 |  |

|  |  |  |
| --- | --- | --- |
| 55 | } | |
| 56 |  |

|  |  |  |
| --- | --- | --- |
| 57 | public void setUsername(String username) | |
| 58 |  |

|  |  |  |
| --- | --- | --- |
| 59 | { | |
| 60 |  |

|  |  |  |
| --- | --- | --- |
| 61 | this.username = username; | |
| 62 |  |

|  |  |  |
| --- | --- | --- |
| 63 | } | |
| 64 |  |

|  |  |
| --- | --- |
| 65 | } |

—————————————————————–

|  |  |  |
| --- | --- | --- |
| 01 | &lt;?xml version='1.0' encoding='utf-8'?&gt; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | &lt;!DOCTYPE hibernate-mapping | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN" | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | "<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>"&gt; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &nbsp; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | &lt;hibernate-mapping&gt; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | &lt;class name="User"&gt; | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | &lt;id name="id" type="int"&gt; | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | &lt;generator/&gt; | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | &lt;/id&gt; | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | &nbsp; | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | &lt;property name="username" type="string" length="32"/&gt; | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | &lt;property name="activated" type="boolean"/&gt; | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | &lt;filter name="activatedFilter" condition=":activatedParam = activated"/&gt; | |
| 28 |  |

|  |  |  |
| --- | --- | --- |
| 29 | &lt;/class&gt; | |
| 30 |  |

|  |  |  |
| --- | --- | --- |
| 31 | &lt;filter-def name="activatedFilter"&gt; | |
| 32 |  |

|  |  |  |
| --- | --- | --- |
| 33 | &lt;filter-param name="activatedParam" type="boolean"/&gt; | |
| 34 |  |

|  |  |  |
| --- | --- | --- |
| 35 | &lt;/filter-def&gt; | |
| 36 |  |

|  |  |
| --- | --- |
| 37 | &lt;/hibernate-mapping&gt; |

——————————————————————–

Save and Fetch using filter example

|  |  |  |
| --- | --- | --- |
| 01 | User user1 = new User(); | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | user1.setUsername("name1"); | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | user1.setActivated(false); | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | &nbsp; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | session.save(user1); | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | &nbsp; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | User user2 = new User(); | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | user2.setUsername("name2"); | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | user2.setActivated(true); | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | &nbsp; | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | session.save(user2); | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | &nbsp; | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | User user3 = new User(); | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | user3.setUsername("name3"); | |
| 28 |  |

|  |  |  |
| --- | --- | --- |
| 29 | user3.setActivated(true); | |
| 30 |  |

|  |  |  |
| --- | --- | --- |
| 31 | &nbsp; | |
| 32 |  |

|  |  |  |
| --- | --- | --- |
| 33 | session.save(user3); | |
| 34 |  |

|  |  |  |
| --- | --- | --- |
| 35 | &nbsp; | |
| 36 |  |

|  |  |  |
| --- | --- | --- |
| 37 | User user4 = new User(); | |
| 38 |  |

|  |  |  |
| --- | --- | --- |
| 39 | user4.setUsername("name4"); | |
| 40 |  |

|  |  |  |
| --- | --- | --- |
| 41 | user4.setActivated(false); | |
| 42 |  |

|  |  |  |
| --- | --- | --- |
| 43 | &nbsp; | |
| 44 |  |

|  |  |
| --- | --- |
| 45 | session.save(user4); |

All the four user saved to Data Base User Table.

Now Fetch the User using Filter..

|  |  |  |
| --- | --- | --- |
| 01 | Filter filter = session.enableFilter("activatedFilter"); | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | filter.setParameter("activatedParam",new Boolean(true)); | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | Query query = session.createQuery("from User"); | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | Iterator results = query.iterate(); | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | while (results.hasNext()) | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | { | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | User user = (User) results.next(); | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | System.out.print(user.getUsername() + " is "); | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | &nbsp; | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | } | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | &nbsp; |

Guess the Result :

name2

name3

Because Filer is filtering ( only true value) data before query execute.

**5.            Difference between save and save or update?**

save – save method stores an object into the database. That means it insert an entry if the identifier doesn’t exist, else it will throw error. If the primary key already present in the table, it cannot be inserted.

update – update method in the hibernate is used for updating the object using identifier. If the identifier is missing or doesn’t exist, it will throw exception.

saveOrUpdate – This method calls save() or update() based on the operation. If the identifier exists, it will call update method else the save method will be called.

**6.            Explain about session.flush()?**

Flushing the Session simply gets the data that is currently in the session synchronized with what is in the database. However, just because you have flushed, doesn’t mean the data can’t be rolled back.

Syntax:

session.flush()

**7.            Difference between hibernate and EJB**

EJB 3.0 used for developing Business component witch required container support like transaction,Security,logging,MDB’s and many more. for persistence EJB3.0 used JPA which can be plug gable to any persistence mechanism like Hibernate (Open source) or TopLink (Oracle Persistence mechanism). Hibernate is just a Persistence Mechanism.

**Struts Interview Question**

**1.            What is Front Controller.**

The Front Controller Pattern is a software design pattern listed in several pattern catalogs. The pattern relates to the design of web applications. It “provides a centralized entry point for handling requests.

**2.            Explain struts Architecture.**

Struts represents an implementation of MVC type 2 architecture. The most important components of Struts are the ActionServlet, Action, and ActionForm subclasses. ActionServlet represents the controller that intercepts the requests and forwards them for further processing based on the configuration file for the ActionForm and Action subclasses. ActionForm transfers data users entered to Action, which performs necessary operations using the Business Tier components of the application and finally forwards to a view.  The controller (ActionServlet) uses a configuration file (typically struts-config.xml) to load definitions of the Action subclasses that will be used to handle the requests.

**3.            Explain MVC in Configuration File.**

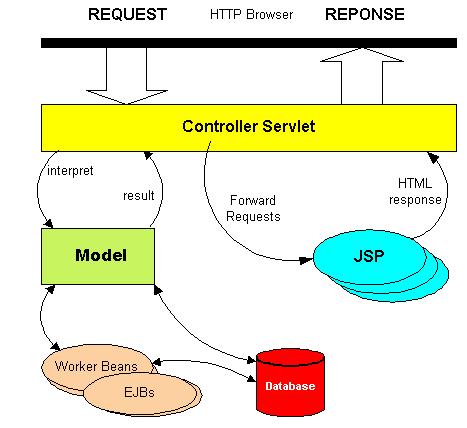
Model-View-Controller is the concept introduced by Smalltalk’s inventors to encapsulating some data together with its processing (the model) and isolate it from the manipulation (the controller) and presentation (the view) part that has to be done on a UserInterface.

A model :- is an object representing data e.g. a database table.

A view :- is some form of visualization of the state of the model.

A controller :- offers facilities to change the state of the model.

**4.            Explain MVC in Java Files.**

[](http://candidjava.com/wp-content/uploads/2011/12/mvc.jpg)

A model :- is an object representing data e.g. a database table.

A view :- is some form of visualization of the state of the model.

A controller :- offers facilities to change the state of the model.

**5.            What is Action Form.**

ActionForm will maintains the session state for web application and the its  object is automatically populated on the server side with data entered from a form on the client side.

**6.            What is Action.**

Action is part of the controller. The use  of this class is to translate the

HTTP Servlet Request to the business logic.

A Struts action is an instance of a subclass of an Action class, which implements a portion of a Web application and whose perform or execute method returns a forward.

**7.            What is ActionServlet.**

In struts technology this class plays the role of controller.controller is responsible for handling all the requests.

The class org.apache.struts.action.ActionServlet is the called the ActionServlet. In the the Jakarta Struts Framework this class plays the role of controller. All the requests to the server goes through the controller. Controller is responsible for handling all the requests.

Struts Flow start with ActionServlet then call to process() method of RequestProcessor.

Step 1. Load ActionServlet using load-on-startup and do the following tasks.

Any struts web application contain the ActionServlet configuration in web.xml file.

On load-on-startup the servlet container Instantiate the ActionServlet .

First Task by ActionServlet : The ActionServlet takes the Struts Config file name as an init-param.

At startup, in the init() method, the ActionServlet reads the Struts Config file and load into memory.

Second Task by ActionServlet : If the user types http://localhost:8080/app/submitForm.do in the browser URL bar, the URL will be intercepted and processed by the ActionServlet since the URL has a pattern \*.do, with a suffix of “do”. Because servlet-mapping is

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

Third Task by ActionServlet : Then ActionServlet delegates the request handling to another class called RequestProcessor by invoking its process() method.

Step 2. ActionServlet calls process() method of RequestProcessor

**8.            What is Global Forward.**

In Struts, Global forwards are the action forwards which can be forwarded from any action. Consider, log out link, that will be present in all the page, so, u can’t write a forward in all the action mapping and check. so, struts is using global forwards, whatever forward declared in the global forwards can be visible to all action mapping. We will use global forwards in situtaions like logoff, error page.

**9.            What is message resource.**

Message Resources Definitions file are simple .properties files and these files contains the messages that can be used in the struts project. Message Resources Definitions files can be added to the struts-config.xml file through < message-resources / > tag.

Example: < message-resources parameter= MessageResources / >

Messsage resource defination files can available to the struts enviornment in two ways

1.By using web.xml as

|  |  |  |
| --- | --- | --- |
| 01 | &lt;servlet&gt; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | &lt;servlet-name&gt;action&lt;servlet-name&gt; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | servlet-class&gt;org.apache.struts.action.ActionServlet&lt;servlet-class&gt; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | &lt;init-param&gt; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &lt;param-name&gt;application&lt;param-name&gt; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | &lt;param-value&gt;resource.Application&lt;param-value&gt; | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | &lt;/servlet&gt; |

**or**

|  |  |  |
| --- | --- | --- |
| 1 | &lt;message-resource key="myResorce" | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | parameter="resource.Application" null="false"&gt; |

**10.          How to call EJB from struts.**

Subclass your Action class and Override execute() method.

Then in body of execute() method do this:

1. Get Initial Context

2. Get home object using JNDI look up.

3 Create the bean and call teh business methods.

We can call EJB from struts by using the service locator design pattern or by Using initial context with create home object and getting return remote referenc object.

**1.            How to handle Errors and Exceptions in struts.**

Exceptions in Struts are handled in two ways:

Programmatic exception handling :

Explicit try/catch blocks in any code that can throw exception. It works well when custom value (i.e., of variable) needed when error occurs.

Declarative exception handling :You can either define <global-exceptions> handling tags in your struts-config.xml or define the exception handling tags within <action></action> tag. It works well when custom page needed when error occurs. This approach applies only to exceptions thrown by Actions.

<global-exceptions>

<exception key=”some.key”

type=”java.lang.NullPointerException”

path=”/WEB-INF/errors/null.jsp”/>

</global-exceptions>

or

<exception key=”some.key”

type=”package.SomeException”

path=”/WEB-INF/somepage.jsp”/>

2.            What are the core classes of Struts.

The Core classes of Struts Framework are

* org.apache.struts.action.ActionForm
* org.apache.struts.action.Action
* org.apache.struts.action.ActionMapping
* org.apache.struts.action.ActionForward
* org.apache.struts.action.ActionServlet
* org.apache.struts.action.ActionError
* org.apache.struts.action.ActionErrors

The core classes of Struts are:

* Action,
* ActionForm,
* ActionServlet,
* ActionMapping,
* ActionForward, etc.

**3.     What are the components of Struts.**

1. Action Servlet

2. Action Classes

3. Action Form

4. PlugIn Classes

4.1.Validator Framework

4.2.Tiles

5. Message Resources

6.Action Mapping Implemention

7. Struts Configuration XML Files

8.Exception Handler

**4.            What are config files in struts.**

* Action mappings
* Global forwards
* Form beans
* Data sources
* Global exceptions
* Controller
* Message resources
* Plugins

**5.            What is Dform.**

A specialized subclass of ActionForm that allows the creation of form beans with dynamic sets of properties (configured in configuration file), without requiring the developer to create a Java class for each type of form bean.

**6.            Difference between Actionform and dform.**

•  An ActionForm represents an HTML form that the user interacts with over one or more pages. You will provide properties to hold the state of the form with getters and setters to access them. Whereas, using DynaActionForm there is no need of providing properties to hold the state. Instead these properties and their type are declared in the struts-config.xml

•  The DynaActionForm bloats up the Struts config file with the xml based definition. This gets annoying as the Struts Config file grow larger.

•  The DynaActionForm is not strongly typed as the ActionForm. This means there is no compile time checking for the form fields. Detecting them at runtime is painful and makes you go through redeployment.

•  ActionForm can be cleanly organized in packages as against the flat organization in the Struts Config file.

•  ActionForm were designed to act as a Firewall between HTTP and the Action classes, i.e. isolate and encapsulate the HTTP request parameters from direct use in Actions. With DynaActionForm, the property access is no different than using request.getParameter( .. ).

•  DynaActionForm construction at runtime requires a lot of Java Reflection (Introspection) machinery that can be avoided.

**7.            What is the use of Dispatch Action.**

The DispatchAction class is used to group related actions into one class. Using this class, you can have a method for each logical action compared than a single execute method. The DispatchAction dispatches to one of the logical actions represented by the methods. It picks a method to invoke based on an incoming request parameter. The value of the incoming parameter is the name of the method that the DispatchAction will invoke.

**8.            What is Mapping Dispatch.**

An abstract Action that dispatches to a public method that is named by the request parameter whose name is specified by the parameter property of the corresponding ActionMapping. This Action is useful for developers who prefer to combine many similar actions into a single Action class, in order to simplify their application design.

Struts MappingDispatch Action (org.apache.struts.actions.MappingDispatchAction) is one of the Built-in Actions provided along with the struts framework.

The org.apache.struts.actions.MappingDispatchAction class is a subclass oforg.apache.struts.actions.DispatchAction class. This class enables a user to collect related functions into a single action class. It  needs to create multiple independent actions for each function.

**9.            What is Lookup Dispatch.**

LookupDispatch is similar to ActionDispatch. It distinguishes the ActionDispatch that uses the request parameter€™s values which performs a reverse lookup from resource bundle which uses the value of parameters and match it to a method in the class.

**10.          How to upload files?**

Org.apache.struts.upload.Formfile is known to be foremost interface for the upload function. This represents the files which have been uploaded by the client. Struts application directly references this interface. This class helps you in uploading files to the database. Here we uploaded a Formfile.

**Step 1.**

Create a form bean

|  |  |  |
| --- | --- | --- |
| 01 | public class FileUploadForm extends ActionForm | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | { | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | private FormFile file; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | public FormFile getFile() { | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | return file; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | } | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | public void setFile(FormFile file) { | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | this.file = file; | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | } | |
| 18 |  |

|  |  |
| --- | --- |
| 19 | } |

Step 2.

**In the struts-config.xml file add**

<form-bean

name=”FileUploadForm”

type=”com.techfaq.form.FileUploadForm”/>

**Step 3.**

**add action mapping entry in the struts-config.xml file:**

<action

path=”/FileUploadAndSave”

type=”com.techfaq.action.FileUploadAndSaveAction”

name=”FileUploadForm”

scope=”request”

validate=”true”

input=”/pages/fileupload.jsp”>

<forward name=”success” path=”/jsp/success.jsp”/>

</action>

**Step 4.**

In the JSP

|  |  |  |
| --- | --- | --- |
| 1 | &lt;html:form action="/FileUploadAndSave" method="post" enctype="multipart/form-data"&gt; | |
| 2 |  |

|  |  |  |
| --- | --- | --- |
| 3 | File Name | |
| 4 |  |

|  |  |  |
| --- | --- | --- |
| 5 | &lt;html:file property="file"/&gt; | |
| 6 |  |

|  |  |  |
| --- | --- | --- |
| 7 | &lt;html:submit&gt;Upload File&lt;/html:submit&gt; | |
| 8 |  |

|  |  |
| --- | --- |
| 9 | &lt;/html:form&gt; |

**Step 5.**

In the Action class write the code

|  |  |  |
| --- | --- | --- |
| 01 | public ActionForward execute( | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | ActionMapping mapping, | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | ActionForm form, | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | HttpServletRequest request, | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | HttpServletResponse response) throws Exception{ | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | FileUploadForm myForm = (FileUploadForm)form; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | // Process the FormFile | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | FormFile file = myForm.getFile(); | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | String contentType = file.getContentType(); | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | //Get the file name | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | String fileName = file.getFileName(); | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | int fileSize = file.getFileSize(); | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | byte[] fileData = file.getFileData(); | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | //Get the servers upload directory real path name | |
| 28 |  |

|  |  |  |
| --- | --- | --- |
| 29 | String filePath = getServlet().getServletContext().getRealPath("/") +"uploadfile"; | |
| 30 |  |

|  |  |  |
| --- | --- | --- |
| 31 | /\* Save file on the server \*/ | |
| 32 |  |

|  |  |  |
| --- | --- | --- |
| 33 | if(!fileName.equals("")){ | |
| 34 |  |

|  |  |  |
| --- | --- | --- |
| 35 | System.out.println("Server path:" +filePath); | |
| 36 |  |

|  |  |  |
| --- | --- | --- |
| 37 | //Create file | |
| 38 |  |

|  |  |  |
| --- | --- | --- |
| 39 | File fileToCreate = new File(file, fileName); | |
| 40 |  |

|  |  |  |
| --- | --- | --- |
| 41 | //If file does not exists create file | |
| 42 |  |

|  |  |  |
| --- | --- | --- |
| 43 | if(!fileToCreate.exists()){ | |
| 44 |  |

|  |  |  |
| --- | --- | --- |
| 45 | FileOutputStream fileOutStream = new FileOutputStream(fileToCreate); | |
| 46 |  |

|  |  |  |
| --- | --- | --- |
| 47 | fileOutStream.write(file.getFileData()); | |
| 48 |  |

|  |  |  |
| --- | --- | --- |
| 49 | fileOutStream.flush(); | |
| 50 |  |

|  |  |  |
| --- | --- | --- |
| 51 | fileOutStream.close(); | |
| 52 |  |

|  |  |  |
| --- | --- | --- |
| 53 | } | |
| 54 |  |

|  |  |  |
| --- | --- | --- |
| 55 | } | |
| 56 |  |

|  |  |  |
| --- | --- | --- |
| 57 | return mapping.findForward("success"); | |
| 58 |  |

|  |  |
| --- | --- |
| 59 | } |

File will be oploaded to “uploadfile” directory og your server.

**1.            Explain about token features in struts**

The problem of duplicate form submission arises when a user clicks the Submit button more than once before the response is sent back. This may result in inconsistent transactions and must be avoided. In Struts this problem can be handled by using the saveToken() and isTokenValid() methods of Action class. saveToken() method creates a token (a unique string) and saves that in the user’s current session, while isTokenValid() checks if the token stored in the user’s current session is the same as that was passed as the request parameter.

**Use the Action Token methods to prevent duplicate submits**

There are methods built into the Struts action to generate one-use tokens. A token is placed in the session when a form is populated and also into the HTML form as a hidden property. When the form is returned, the token is validated. If validation fails, then the form has already been submitted, and the user can be apprised.

saveToken(request)

on the return trip,

isTokenValid(request)

resetToken(request)

**2.            What are the drawback of struts**

Disadvantages of Struts:

**Bigger Learning Curve** To use MVC with Struts, you have to be comfortable with the standard JSP and servlet APIs and a large and elaborate framework that is almost equal in size to the core system.

**Worse Documentation** Compared to the standard servlet and JSP APIs, Struts has fewer online resources, and many first-time users find the online Apache documentation confusing and poorly organized.

**Less Transparent** Struts applications are: Harder to understand and Harder to benchmark and optimize.

Rigid Approach The flip side of the benefit that Struts encourages a consistent approach to MVC is that Struts makes it difficult to use other approaches.

**Bigger Learning Curve**

To use MVC with the standard RequestDispatcher, you need to be comfortable with the standard JSP and servlet APIs. To use MVC with Struts, you have to be comfortable with the standard JSP and servlet APIs and a large and elaborate framework that is almost equal in size to the core system. This drawback is especially significant with smaller projects, near-term deadlines, and less experienced developers; you could spend as much time learning Struts as building your actual system.

**Worse Documentation.**

Compared to the standard servlet and JSP APIs, Struts has fewer online resources, and many first-time users find the online Apache documentation confusing and poorly organized. There are also fewer books on Apache Struts than on standard servlets and JSP.

**Less Transparent.**

With Struts applications, there is a lot more going on behind the scenes than with normal Java-based Web applications. As a result, Struts applications are:

* Harder to understand
* Harder to benchmark and optimize
* Rigid Approach.

The flip side of the benefit that Struts encourages a consistent approach to MVC is that Struts makes it difficult (but by no means impossible) to use other approaches.

**3.            What are the advantage of struts**

You can easily implement the MVC approach by using RequestDispatcher.forward in your servlets and jsp:getProperty or the JSP 2.0 expression language in your JSP pages. However, Struts offers a number of significant advantages over these techniques alone. Here is a summary:

**•             Centralized File-Based Configuration.**

Rather than hard-coding information into Java programs, many Struts values are represented in XML or property files. This loose coupling means that many changes can be made without modifying or recompiling Java code, and that wholesale changes can be made by editing a single file. This approach also lets Java and Web developers focus on their specific tasks (implementing business logic, presenting certain values to clients, etc.) without needing to know about the overall system layout.

**•             Form Beans.**

In JSP, you can use property=”\*” with jsp:setProperty to automatically populate a JavaBean component based on incoming request parameters. Unfortunately, however, in the standard API this capability is unavailable to servlets, even though with MVC it is really servlets, not JSP pages, that should usually be the target of form submissions. Apache Struts extends this capability to Java code and adds in several useful utilities, all of which serve to greatly simplify the processing of request parameters.

**•             Bean Tags.**

Apache Struts provides a set of custom JSP tags (bean:write, in particular) that let you easily output the properties of JavaBeans components. Basically, these are concise and powerful variations of the standardjsp:useBean and jsp:getProperty tags.

**•             HTML Tags.**

Apache Struts provides a set of custom JSP tags to create HTML forms that are associated with JavaBeans components. This bean/form association serves two useful purposes:

* It lets you get initial form-field values from Java objects.
* It lets you redisplay forms with some or all previously entered values intact.
* Form Field Validation.

Apache Struts has builtin capabilities for checking that form values are in the required format. If values are missing or in an improper format, the form can be automatically redisplayed with error messages and with the previously entered values maintained.

This validation can be performed on the server (in Java), or both on the server and on the client (in JavaScript).

* Consistent Approach.

Struts encourages consistent use of MVC throughout your application.

**4.            What are the important methods in ActionForm**

The important methods of ActionForm are : validate() & reset().

**5.            Difference between ActionError and ActionMessage**

Action Errors(org.apache.struts.action.ActionErrors): A class that encapsulates the error messages being reported by the validate() method of an ActionForm. Validation errors are either global to the entire ActionForm bean they are associated with, or they are specific to a particular bean property (and, therefore, a particular input field on the corresponding form). Each individual error is described by an ActionMessage object, which contains a message key (to be looked up in an appropriate message resources database), and up to four placeholder arguments used for parametric substitution in the resulting message.

ActionMessage(org.apache.struts.action.ActionMessage): An encapsulation of an individual message returned by the validate method of an ActionForm, consisting of a message key (to be used to look up message text in an appropriate message resources database) plus up to four placeholder objects that can be used for parametric replacement in the message text.

**6.            What are the design patterns used in struts.**

Struts is based on model 2 MVC (Model-View-Controller) architecture. Struts controller uses the command design pattern and the action classes use the adapter design pattern. The process() method of the RequestProcessor uses the template method design pattern. Struts also implement the following J2EE design patterns.

* Service to Worker
* Dispatcher View
* Composite View (Struts Tiles)
* Front Controller
* View Helper
* Synchronizer Token

**7.            Explain Struts plugins.**

Struts Plugins are modular extensions to the Struts COntroller. They are defined by the org.apache.struts.action.Plugin interface.Struts Plugins are useful are useful when you are allocating resources or preparing connections to the databases or even JNDI resources. This interface defines two lifecycle mathods: init() and desstroy().

**8.            How to have multiple config files.**

Yes, we can have more than one struts-config.xml for a single Struts application. They can be configured as follows:

|  |  |  |
| --- | --- | --- |
| 01 | &lt;servlet&gt; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | &lt;servlet-name&gt;action&lt;/servlet-name&gt; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | &lt;servlet-class&gt; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | org.apache.struts.action.ActionServlet | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &lt;/servlet-class&gt; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | &lt;init-param&gt; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | &lt;param-name&gt;config&lt;/param-name&gt; | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | &lt;param-value&gt; | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | /WEB-INF/struts-config.xml, | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | /WEB-INF/struts-admin.xml, | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | /WEB-INF/struts-config-forms.xml | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | &lt;/param-value&gt; | |
| 24 |  |

|  |  |  |
| --- | --- | --- |
| 25 | &lt;/init-param&gt; | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | ..... | |
| 28 |  |

|  |  |
| --- | --- |
| 29 | &lt;servlet&gt; |

**9.            What is switch action**

The SwitchAction cl provides a means to switch from a resource in one module to another resource in a different module. SwitchAction is useful only if you have multiple modules in your Struts application. The SwitchAction cl can be used as is, without extending.

**10.          What is include action**

The IncludeAction cl is useful when you want to integrate Struts into an application that uses Servlets. Use the IncludeAction cl to include another resource in the response to the request being processed.

**1.            What is the use of validate and reset method**

The validate method is called by the controller servlet after the bean properties have been populated, but before the corresponding action class’s execute method is invoked.

reset(): reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm?s data members prior to the new request values being set.

**2.            Difference between perform() and execute()**

Perform method is the method which was deprecated in the Struts Version 1.1.  In Struts 1.x, Action.perform() is the method called by the ActionServlet. This is typically where your business logic resides, or at least the flow control to your JavaBeans and EJBs that handle your business logic. As we already mentioned, to support declarative exception handling, the method signature changed in perform. Now execute just throws Exception. Action.perform() is now deprecated; however, the Struts v1.1 ActionServlet is smart enough to know whether or not it should call perform or execute in the Action, depending on which one is available.

**3.            Difference between html and struts based html tags?**

Struts tags bind the property value with the Formbean property defined. HTML tags are static and Struts tags are dynamic.

**4.            Difference between ForwardAction and IncludeAction**

The difference is that you need to use the IncludeAction only if the action is going to be included by another action or jsp. Use ForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another JSP page.

**5.            How container identify struts.**

This can be accomplished by adding the following servlet definition to the web.xml file :

|  |  |  |
| --- | --- | --- |
| 01 | &lt;servelt&gt; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | &lt;servlet-name&gt;action&lt;/servlet-name&gt; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | &lt;servlet-class&gt;org.apache.struts.action.ActionServlet&lt;/servlet-class&gt; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | &lt;init-param&gt; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | &lt;param-name&gt;config&lt;/param-name&gt; | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | &lt;param-value&gt;/WEB-INF/struts-config.xml&lt;/param-value&gt; | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | &lt;/init-param&gt; | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | &lt;load-on-startup&gt;1&lt;/load-on-startup&gt; | |
| 16 |  |

|  |  |
| --- | --- |
| 17 | &lt;/servlet&gt; |

Once you told the container about the ActionServlet you need to tell it when it should be executed by adding <servelt-mapping> element in web.xml file :

|  |  |  |
| --- | --- | --- |
| 1 | &lt;servlet-mapping&gt; | |
| 2 |  |

|  |  |  |
| --- | --- | --- |
| 3 | &lt;servlet-name&gt;action&lt;/servlet-name&gt; | |
| 4 |  |

|  |  |  |
| --- | --- | --- |
| 5 | &lt;url-pattern&gt;\*.do&lt;/url-pattern&gt; | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | &lt;/servlet-mapping&gt; |

This mapping tells the web application that whenever a request is received with .do appended to the URL then the servlet named action should service the request.

**6.            Super class of Action and ActionForm.**

Action is a class and ActionForm is an abstract class.

**7.            Can we have a custom name for lib folder.**

No

**8.            What are the Advantages of Struts ?**

Struts follow MVC framework. So the JSP, Java and Action classes are organized and easily maintainable.

Struts offers many advantages to the application programmer while reducing the development time and making the manageability of the application easier.

**Advantages of Struts :**

**Centralized Configuration :**

Rather than hard coding information into java programs,many Struts values are represented in XML or property files.

Struts\_config.xml file is the place from where you can get all information?s about your web application. This is organized.

Your Action class , Form bean and JSP page information is in Struts\_config.xml so don’t need to search . All info in one place.

**Form Beans :**

Don’t need to set the form vales to your value object . When you want to capture data from a form ( In the servlet you do request.getParameter()).

In the struts you don’t need to do explicitly request.getParameter(). Struts request processor will do for you. All the input data will be set to form bean.

**Bean Tags :**

Struts provides a set of custom JSP tags (bean:write,in particular) that let you easily output the properties of JavaBeans components.

Basically,these are concise and powerful variations of the standard jsp:useBean and jsp:getProperty tags.

**HTML tags :**

Struts provides a set of custom JSP tags to create HTML forms that are associated with JavaBeans components.

**Form Field Validation :**

Apache Struts has built-in capabilities for checking that form values are in the required format.

If values are missing or in an improper format,the form can be automatically redisplayed with error messages and with the previously entered values maintained.

This validation can be performed on the server (in Java),or both on the server and on the client (in JavaScript).

**9.            Describe validate() and reset() methods?**

validate() : Used to validate properties after they have been populated; Called before FormBean is handed to Action. Returns a collection of ActionError as ActionErrors. Following is the method signature for the validate() method.

reset(): reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm’s data members prior to the new request values being set.

**1.Whats the difference between constructors and normal methods?**

Constructors must have the same name as the class and can not return a value. They are only called once while regular methods could be called many times and it can return a value or can be void.

**2.What is the final keyword denotes?**

final keyword denotes that it is the final implementation for that method or variable or class. You cant override that method/variable/class any more.

**3.  What is the purpose of Void class?**

The Void class is an uninstantiable placeholder class to hold a reference to the Class object representing the primitive Java type void.

**4.What is the finalize method do?**

Before the invalid objects get garbage collected, the JVM give the user a chance to clean up some resources before it got garbage collected

**5.What is JIT and its use?**

Really, just a very fast compiler In this incarnation, pretty much a one-pass compiler no offline computations. So you can not look at the whole method, rank the expressions according to which ones are re-used the most, and then generate code. In theory terms, its an on-line problem( JIT is a compiler for Byte code, The JIT-Complier is part of the *JVM*, it complies byte code into executable code in real time, piece-by-piece on demand basis.)

**6.  What is Abstraction ?**

Showing the essential and hiding the non-Essential is known as Abstraction.

**7.What is Encapsulation?**

The Wrapping up of data and functions into a single unit is known as Encapsulation.*Encapsulation* is the term given to the process of hiding the implementation details of the object. Once an object is encapsulated, its implementation details are not immediately accessible any more. Instead they are packaged and are only indirectly accessed via the interface of the object.

**8.What is mean by Inheritance?**

Inheritance is the Process by which the *Obj* of one class acquires the properties of *Objs*another Class.  
*Inheritance* is the method of creating the new class based on already existing class , the new class derived is called Sub class which has all the features of existing class and its own, i.e sub class.  
Adv: Reusability of code , accessibility of variables and methods of the Base class by the Derived class.

**9.What is Polymorphism?**

The ability to take more that one form, it supports Method Overloading & Method Overriding.

**10.What is Method overloading ?**

When a method in a class having the *same method name* with *different arguments* (diff Parameters or Signatures) is said to be Method Overloading. This is *Compile time*Polymorphism.

When a method in a class having the same method name with different arguments (diff Parameters o

**12.What is Method Overriding?**

When a method in a sub Class having same method name with same arguments present in a super class it is said to be Method overriding. This is Run time Polymorphism.

**13.What is mean by Dynamic dispatch?**

Dynamic dispatch is a mechanism by which a call to Overridden function is resolved at runtime rather than at Compile time , and this is how Java implements Run time Polymorphism.

**14.What is mean by Dynamic Binding?**

Means the code associated with the given procedure call is not known until the time of call the call at run time. (it is associated with Inheritance & Polymorphism).

**15.What is mean by Bite code?**

Is a optimized set of instructions designed to be executed by Java-run time system, which is called the Java Virtual machine (JVM), i.e. in its standard form, the JVM is an Interpreter for byte code.

**16.What is class variable?**

A data item associated with a particular class as a whole–not with particular instances of the class. Class variables are defined in class definitions. Also called a static field.

**17.What is instance variable?**

Any item of data that is associated with a particular object. Each instance of a class has its own copy of the instance variables defined in the class. Also called a field

**18.What is local variable?**

A data item known within a block, but inaccessible to code outside the block. For example, any variable defined within a method is a local variable and can’t be used outside the method.

**19.What is class method ?**

A method that is invoked without reference to a particular object. Class methods affect the class as a whole, not a particular instance of the class. Also called a static method.

**20.What is instance method ?**

Any method that is invoked with respect to an instance of a class. Also called simply a method.

**21.Define Interface&Rules?**

Interfaces can be used to implement the Inheritance relationship between the non-related classes that do not belongs to the same hierarchy, i.e. any Class and anywhere in hierarchy.   Using Interface, you can specify what a class must do but not how it does.

**Rules for defining a interface**

* A class can implement more than one Interface.
* An Interface can extend one or more interfaces, by using the keyword extends.
* All the data members in the interface are public, static and Final by default.
* An Interface method can have only Public, default and Abstract modifiers.
* An Interface is loaded in memory only when it is needed for the first time.
* A Class, which implements an Interface, needs to provide the implementation of all the methods in that Interface.
* If the Implementation for all the methods declared in the Interface are not provided , the class itself has to declare abstract, other wise the Class will not compile.
* If a class Implements two interface and both the Intfs have identical method declaration, it is totally valid.
* If a class implements tow interfaces both have identical method name and argument list, but different return types, the code will not compile.
* An Interface cant be instantiated. Intf Are designed to support dynamic method resolution at run time.
* An interface can not be native, static, synchronize, final, protected or private.
* The Interface fields can not be Private or Protected.
* A Transient variables and Volatile variables can not be members of  Interface.
* The extends keyword should not used after the Implements keyword, the Extends must always come before the Implements keyword.
* A top level Interface can not be declared as static or final.
* If an Interface species an exception list for a method, then the  class implementing the interface need not declare the method with  the exception list.
* If an Interface can not specify an exception list for a method, the class can not throw an exception.
* If an Interface does not specify the exception list for a method, he class can not throw any exception list.

**21.What is mean Abstract Class ?**

Abstract classes can be used to implement the inheritance relationship between the classes that belongs same hierarchy.  
Classes and methods can be declared as abstract.  
If a Class is declared as abstract , no instance of that class can be created.  
If a method is declared as abstract, the sub class gives the implementation of that class.  
Even if a single method is declared as abstract in a Class ,Â the class itselfÂ can be declared as abstract.  
In abstract Class the keyword abstract must be used for method.  
Abstract class cannot be marked as final  
22.Difference Between Interfaces And Abstract class ?  
All the methods declared in the Interface are Abstract, where as abstract class have both abstract method and concrete.  
In abstract class keyword abstract must be used for method, where as in Interface we need not use the keyword for methods(interface methods are by default abstract).  
Interface can have only abstract method and constants  
Interface does not have constructor where as abstract class have constructor.

**23.What are access specifiers and access modifiers ?**  
Public  
: The Variables and methods can be access any where and any package.  
Protected  
: The Variables and methods can be access same Class, same Package & sub class. Protected keyword is used when we have different packages  
Private  
: The variable and methods can be access in same class only.  
**24. What are Identifiers ?**

Identifiers are the Variables that are declared under particular Datatype.  
**25. What are Literals?**

Literals are the values assigned to the Identifiers.  
**26. Where did the Java name come from? What does it stand for?**

The name was chosen during one of several brainstorming sessions held by the Java software team. We were aiming to come up with a name that evoked the essence of the technology — liveliness, animation, speed, interactivity, and more. “Java” was chosen from among many, many suggestions. The name is not an acronym, but rather a reminder of that hot, aromatic stuff that many programmers like to drink lots of.

**27. Why developers should not write programs that call ‘sun’ packages**

Java Software supports into the future only classes in java.\* packages, not sun.\* packages. In general, API in sun.\* is subject to change at any time without notice. For more details, see the article Why Developers Should Not Write Programs That Call ‘sun’ Packages.

**28. What releases of Java technology are currently available? What do they contain?**

The Java programming language is currently shipping from Sun Microsystems, Inc. as the Java Development Kit (JDKTM). All Sun releases of the JDK software are available from the JDK software home page (http://java.sun.com/products/jdk/). Each release of the Java Development Kit (JDK) contains:

Java Compiler  
Java Virtual Machine\*  
Java Class Libraries  
Java AppletViewer  
Java Debugger and other tools  
Documentation (in a separate download bundle)  
To run Java 1.0 applets, use Netscape Navigator 3.x or other browsers that support Java applets. To run Java 1.1.x applets, use HotJavaTM 1.x or Netscape Navigator 4.x or other browsers that support the newest version of the Java API.

**29.What is the difference between a constructor and a method?**

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator.  
A method is an ordinary member function of a class. It has its own name, a return type (which may bevoid), and is invoked using the dot operator.  
**30.How are this() and super() used with constructors?**

A: This() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

**31.What is the difference between static and non-static variables?**

A static variable is associated with the class as a whole rather than with specific instances of a class.

Non-static variables take on unique values with each object instance.

**32. What is the difference between a while statement and a do statement?**

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute the body of a loop at least once.

**33. Does Java provide any construct to find out the size of an object?**

No there is not sizeof operator in Java. So there is not direct way to determine the size of an object directly in Java.

**34. What are pass by reference and passby value?**

Pass By Reference means the passing the address itself rather than passing the value. Passby Value means passing a copy of the value to be passed.

**35. What is an Iterator?**

Some of the collection classes provide traversal of their contents via a java.util.Iterator interface. This interface allows you to walk through a collection of objects, operating on each object in turn. Remember when using Iterators that they contain a snapshot of the collection at the time the Iterator was obtained; generally it is not advisable to modify the collection itself while traversing an Iterator.

**36. Declaration of access specifier and access modifiers :**

Class – Public, Abstract, Final

Inner Class – Public, Protected, Private, Final, Static,

Anonymous – Public, Protected, Private, Static

Variable – Public, Protected, Private, Final, Static, Transient, Volatile, Native

Method – Public, Protected, Private, Final, Abstract, Static, Native, Synchronized

Constructor – Public, Protected, Private

Free-floating code block – Static, Synchronized.

**37. What is an abstract class?**

Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is

abstract may not be instantiated (ie, you may not call its constructor), abstract class may contain

static data. Any class with an abstract method is automatically abstract itself, and must be declared as

such.A class may be declared abstract even if it has no abstract methods. This prevents it from being

instantiated.

**38.What is static in java?**

Static means one per class, not one for each object no matter how many instance of a class might

exist. This means that you can use them without creating an instance of a class. Static methods are

implicitly final, because overriding is done based on the type of the object, and static methods are

attached to a class, not an object. A static method in a superclass can be shadowed by another static

method in a subclass, as long as the original method was not declared final. However, you can’t

override a static method with a nonstatic method. In other words, you can’t change a static method

into an instance method in a subclass.

**39. What is final?**

A final class can’t be extended ie., final class may not be subclassed. A final method can’t be

overridden when its class is inherited. You can’t change value of a final variable (is a constant).

**40.What if the static modifier is removed from the signature of the main method?**

Program compiles. But at runtime throws an error “NoSuchMethodError”.

**41.What if I write static public void instead of public static void?**

Program compiles and runs properly.

**42.What if I do not provide the String array as the argument to the method?**

Program compiles but throws a runtime error “NoSuchMethodError”.

**43.What is the first argument of the String array in main method?**

The String array is empty. It does not have any element. This is unlike C/C++ where the first element by default is the program name.

**44.If I do not provide any arguments on the command line, then the String array of Main method will be empty or null?**

It is empty. But not null.

**45.How can one prove that the array is not null but empty using one line of code?**

Print args.length. It will print 0. That means it is empty. But if it would have been null then it would have thrown a NullPointerException on attempting to print args.length.

**46.What environment variables do I need to set on my machine in order to be able to run Java programs?**

CLASSPATH and PATH are the two variables.

**47.Can an application have multiple classes having main method?**

Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned. Hence there is not conflict amongst the multiple classes having main method.

**48.Can I have multiple main methods in the same class?**

No the program fails to compile. The compiler says that the main method is already defined in the class.

**49. What is the difference between declaring a variable and defining a variable?**

In declaration we just mention the type of the variable and it’s name. We do not initialize it. But defining means declaration + initialization.e.g String s; is just a declaration while String s = new String (“abcd”); Or String s = “abcd”; are both definitions.

**50. What is the default value of an object reference declared as an instance variable?**

Null unless we define it explicitly.

**51.  Does garbage collection guarantee that a program will not run out of memory?**

Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to garbage collection.

**52. Can an unreachable object become reachable again?**

An unreachable object may become reachable again. This can happen when the object’s finalize() method is invoked and the object performs an operation which causes it to become accessible to reachable objects.

**53.  What modifiers are allowed for methods in an Interface?**

Only public and abstract modifiers are allowed for methods in interfaces.

**54.  What are some alternatives to inheritance?**

Delegation is an alternative to inheritance. Delegation means that you include an instance of another class as an instance variable, and forward messages to the instance. It is often safer than inheritance because it forces you to think about each message you forward, because the instance is of a known class, rather than a new class, and because it doesn’t force you to accept all the methods of the super class: you can provide only the methods that really make sense. On the other hand, it makes you write more code, and it is harder to re-use (because it is not a subclass).

**55.  What does it mean that a method or field is “static”?**

Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That’s how library methods like System.out.println() work out is a static field in the java.lang.System class.

**56.  Is Empty .java file a valid source file?**

Yes, an empty .java file is a perfectly valid source file.

**57.  Can a .java file contain more than one java classes?**

Yes, a .java file contain more than one java classes, provided at the most one of them is a public class.

**58.  Is String a primitive data type in Java?**

No String is not a primitive data type in Java, even though it is one of the most extensively used object. Strings in Java are instances of String class defined in java.lang package.

**59.  Is main a keyword in Java?**  
No, main is not a keyword in Java.  
**60.          Is next a keyword in Java?**  
No, next is not a keyword.

**61.          Is delete a keyword in Java?**  
No, delete is not a keyword in Java. Java does not make use of explicit destructors the way C++ does.

**62.          Is exit a keyword in Java?**  
No. To exit a program explicitly you use exit method in System object.

**63.          What happens if you dont initialize an instance variable of any of the primitive types in Java?**  
Java by default initializes it to the default value for that primitive type. Thus an int will be initialized to 0, a boolean will be initialized to false.

**64.          What will be the initial value of an object reference which is defined as an instance variable?**  
The object references are all initialized to null in Java. However in order to do anything useful with these references, you must set them to a valid object, else you will get NullPointerExceptions everywhere you try to use such default initialized references.

**65.          What are the different scopes for Java variables?**  
The scope of a Java variable is determined by the context in which the variable is declared. Thus a java variable can have one of the three scopes at any given point in time.  
Instance : -  
These are typical object level variables, they are initialized to default values at the time of creation of object, and remain accessible as long as the object accessible.  
Local : -  
These are the variables that are defined within a method. They remain accessbile only during the course of method excecution. When the method finishes execution, these variables fall out of scope.  
Static: -  
These are the class level variables. They are initialized when the class is loaded in JVM for the first time and remain there as long as the class remains loaded. They are not tied to any particular object instance.

**66.          What is the default value of the local variables?**  
The local variables are not initialized to any default value, neither primitives nor object references. If you try to use these variables without initializing them explicitly, the java compiler will not compile the code. It will complain abt the local varaible not being initilized..

**67.          Can main method be declared final?**  
Yes, the main method can be declared final, in addition to being public static.

**68.          What will be the output of the following statement?**  
System.out.println (“1″ + 3);  
It will print 13.

**Q1. What are the components of Struts Framework?**

Ans: Struts framework is comprised of following components:

1. Java Servlets
2. JSP (Java Server Pages)
3. Custom Tags
4. Message Resources

**Q2. What’s the role of a handler in MVC based applications?**

Ans:. It’s the job of handlers to transfer the requests to appropriate models as they are bound to the model layer of MVC architecture. Handlers use mapping information from configuration files for request transfer.

**Q3. What’s the flow of requests in Struts based applications?**

Ans: Struts based applications use MVC design pattern. The flow of requests is as follows:

* User interacts with View by clicking any link or by submitting any form.
* Upon user’s interaction, the request is passed towards the controller.
* Controller is responsible for passing the request to appropriate action.
* Action is responsible for calling a function in Model which has all business logic implemented.
* Response from the model layer is received back by the action which then passes it towards the view where user is able to see the response.

**Q4.  Which file is used by controller to get mapping information for request routing?**

Ans: Controller uses a configuration file “struts-config.xml file to get all mapping information to decide which action to use for routing of user’s request.

**Q5.  What’s the role of Action Class in Struts?**

[](http://cdn.gointerviews.com/wp-content/uploads/2012/03/Struts-Interview-Questions.jpg)

Ans: In Struts, Action Class acts as a controller and performs following key tasks:

* After receiving user request, it processes the user’s request.
* Uses appropriate model and pulls data from model (if required).
* Selects proper view to show the response to the user.

**Q6. How an actionForm bean is created?**

**Surrogate**

Ans: actionForm bean is created by extending the classorg.apache.struts.action.ActionForm

In the following example we have created an actionForm bean with the name 'testForm':

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | import javax.servlet.http.HttpServletRequest;    import org.apache.struts.action.\*;    public class testForm extends ActionForm   {   private String Id=null;   private String State=null;   public void setId(String id){   this.Id=id;   }    public String getId(){   return this.Id;   }    public void setState(String state){   this.State=state;   }    public String getState(){   return this.State;   } |

**Q7. What are the two types of validations supported by Validator FrameWork?**

Ans: Validator Framework is used for form data validation. This framework provides two types of validations:

1. Client Side validation on user’s browser
2. Server side validation

**Q8. What are the steps of Struts Installation?**

Ans: In order to use Struts framework, we only need to add Struts.Jar file in our development environment. Once jar file is available in the CLASSPATH, we can use the framework and develop Strut based applications.

**Q9. How client side validation is enabled on a JSP form?**

Ans: In order to enable client side validation in Struts, first we need to enable validator plug-in in struts-config.xml file. This is done by adding following configuration entries in this file:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!--  Validator plugin -->   <plug-in className="org.apache.struts.validator.ValidatorPlugIn">   <set-property   property="pathnames"   value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>   </plug-in> |

Then Validation rules are defined in validation.xml file. If a form contains email field and we want to enable client side validation for this field, following code is added in validation.xml file:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <form name="testForm">   <field  property="email"   depends="required">   <arg key="testForm.email"/>   </field>   </form> |

**Q10. How action-mapping tag is used for request forwarding in Struts configuration file?**

Ans: In Struts configuration file (struts-config.xml), forwarding options are defined under action-mapping tag.

In the following example, when a user will click on the hyperlink **test.do**, request will be forwarded to **/pages/testing.jsp**using following configurations from struts-config.xml file:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1 | <action  path="/test" forward="/pages/testing.jsp"> |

This forwarding will take place when user will click on following hyperlink on the jsp page:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1 | <html:link</strong> page="/test.do</strong>">Controller Example</html:link> |

**Q11. How duplicate form submission can be controlled in Struts?**

Ans: In Struts, action class provides two important methods which can be used to avoid duplicate form submissions.

saveToken() method of action class generates a unique token and saves it in the user’s session. isTokenValid() method is used then used to check uniqueness of tokens.

**Q12. In Struts, how can we access Java beans and their properties?**

Ans: Bean Tag Library is a Struts library which can be used for accessing Java beans.

**Q13. Which configuration file is used for storing JSP configuration information in Struts?**

Ans: For JSP configuration details, Web.xml file is used.

**Q14. What’s the purpose of Execute method of action class?**

Ans: Execute method of action class is responsible for execution of business logic. If any processing is required on the user’s request, it’s performed in this method. This method returns actionForward object which routes the application to appropriate page.

In the following example, execute method will return an object of actionForward defined in struts-config.xml with the name “exampleAction”:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | import javax.servlet.http.HttpServletRequest;   import javax.servlet.http.HttpServletResponse;    import org.apache.struts.action.Action;   import org.apache.struts.action.ActionForm;   import org.apache.struts.action.ActionForward;   import org.apache.struts.action.ActionMapping;    public class actionExample extends Action   {     public ActionForward execute(     ActionMapping mapping,     ActionForm form,     HttpServletRequest request,     HttpServletResponse response) throws Exception{     return mapping.findForward("exampleAction");     }   } |

**Q15. What’s the difference between validation.xml and validator-rules.xml files in Struts Validation framework?**

Ans: In Validation.xml, we define validation rules for any specific Java bean while in validator-rules.xml file, standard and generic validation rules are defined.

**Q16. How can we display all validation errors to user on JSP page?**

Ans: To display all validation errors based on the validation rules defined in validation.xml file, we use <html:errors /> tag in our JSP file.

**Q17. What’s declarative exception handling in Struts?**

Ans: When logic for exception handling is defined in struts-config.xml or within the action tag, it’s known as declarative exception handling in Struts.

In the following example, we have defined exception in struts-config.xml file for NullPointerException:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <global-exceptions>    <exception key="test.key"    Type="java.lang.NullPointerException"    Path="/WEB-INF/errors/error\_page.jsp"    </global-exceptions> |

**Q18. What’s DynaActionForm?**

Ans: DynaActionForm is a special type of actionForm class (sub-class of ActionForm Class) that’s used for dynamically creating form beans. It uses configuration files for form bean creation.

**Q19. What configuration changes are required to use Tiles in Struts?**

Ans: To create reusable components with Tiles framework, we need to add following plugin definition code in struts-config.xml file:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <plug-in className="org.apache.struts.tiles.TilesPlugin" >    <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />    <set-property property="moduleAware" value="true" />    </plug-in> |

**Q20. What’s the difference between Jakarta Struts and Apache Struts? Which one is better to use?**

Ans: Both are same and there is no difference between them.

**Q21. What’s the use of Struts.xml configuration file?**

Ans: Struts.xml file is one the key configuration files of Struts framework which is used to define mapping between URL and action. When a user’s request is received by the controller, controller uses mapping information from this file to select appropriate action class.

**Q22. How tag libraries are defined in Struts?**

Ans: Tag libraries are defined in the configuration file (web.xml) inside <taglib> tag as follows:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <taglib>    <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>    <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>    </taglib> |

**Q23. What’s the significance of logic tags in Struts?**

Ans: Use of logic tags in Struts helps in writing a clean and efficient code at presentation layer without use of scriptlets.

**Q24. What are the two scope types for formbeans?**

Ans: 1. Request Scope: Formbean values are available in the current request only

2. Session Scope: Formbean values are available for all requests in the current session.

**Q25. How can we group related actions in one group in Struts?**

Ans: To group multiple related actions in one group, we can use DispatcherAction class.

**Q26. When should we use SwtichAction?**

Ans: The best scenario to use SwitchAction class is when we have a modular application with multiple modules working separately. Using SwitchAction class we can switch from a resource in one module to another resource in some different module of the application.  
**Q27. What are the benefits of Struts framework?**

Ans: Struts is based on MVC and hence there is a good separation of different layers in Struts which makes Struts applications development and customization easy. Use of different configuration files makes Struts applications easily configurable. Also, Struts is open source and hence, cost effective.

**Q28. What steps are required to for an application migration from Struts1 to Struts2?**

Ans: Following Steps are required for Struts1 to Struts2 migration:

1. Move Struts1 actionForm to Struts2 POJO.
2. Convert Struts1 configuration file (struts-config.xml) to Struts2 configuration file (struts.xml)

**Q29. How properties of a form are validated in Struts?**

Ans: For validation of populated properties, validate() method of ActionForm class is used before handling the control of formbean to Action class.

**Q30. What’s the use of reset method of ActionForm class?**

Ans: reset method of actionForm class is used to clear the values of a form before initiation of a new request.

**Q31. What are disadvantages of Struts?**

Ans: Although Struts have large number of advantages associated, it also requires bigger learning curve and also reduces transparency in the development process.

Struts also lack proper documentation and for many of its components, users are unable to get proper online resources for help.

**Q32. What’s the use of resourcebundle.properties file in Struts Validation framework?**

Ans: resourcebundle.properties file is used to define specific error messages in key value pairs for any possible errors that may occur in the code.

This approach helps to keep the code clean as developer doesn’t need to embed all error messages inside code.

**Q33. Can I have html form property without associated getter and setter formbean methods?**

Ans: For each html form property, getter and setter methods in the formbean must be defined otherwise application results in an error.

**Q34. How many servlet controllers are used in a Struts Application?**

Ans: Struts framework works on the concept of centralized control approach and the whole application is controlled by a single servlet controller. Hence, we require only one servlet controller in a servlet application.

**Q35. For a single Struts application, can we have multiple struts-config.xml files?**

Ans: We can have any number of Struts-config.xml files for a single application.

We need following configurations for this:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | <servlet>    <servlet-name>action</servlet-name>    <servlet-class>    org.apache.struts.action.ActionServlet    </servlet-class>    <init-param>    <param-name>config</param-name>    <param-value>    /WEB-INF/struts-config.xml    /WEB-INF/struts-config\_user.xml    /WEB-INF/struts-config\_admin.xml    </param-value>    </init-param>    .............    .............    </servlet> |

**Q36. Which model components are supported by Struts?**

Ans: Struts support all types of models including Java beans, EJB, CORBA. However, Struts doesn’t have any in-built support for any specific model and it’s the developer’s choice to opt for any model.

**Q37. When it’s useful to use IncludeAction?**

Ans: IncludeAction is action class provided by Struts which is useful when an integration is required between Struts and Servlet based application.

**Q38. Is Struts thread safe?**

Ans: Yes Struts are thread safe. In Struts, a new servlet object is not required to handle each request; rather a new thread of action class object is used for each new request.

**Q39. What configuration changes are required to use resource files in Struts?**

Ans:  Resource files (.properties files) can be used in Struts by adding following configuration entry in struts-config.xml file:

<message-resources parameter=”com.login.struts.ApplicationResources”/>

**Q40. How nested beans can be used in Struts applications?**

Ans: Struts provide a separate tag library (Nested Tag Library) for this purpose. Using this library, we can nest the beans in any Struts based application.

**Q41. What are the Core classes of Struts Framework?**

Ans: Following are the core classes provided by Struts Framework:

* Action Class
* ActionForm Class
* ActionMapping Class
* ActionForward Class
* ActionServlet Class

**Q42. Can we handle exceptions in Struts programmatically?**

Ans: Yes we can handle exceptions in Struts programmatically by using try, catch blocks in the code.

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | try {    // Struts code    }    Catch (Exception e) {    // exception handling code    } |

**Q43. Is Struts Framework part of J2EE?**

Ans: Although Struts framework is based on J2EE technologies like JSP, Java Beans, Servlets etc but it’s not a part of J2EE standards.

**Q44.  How action mapping is configured in Struts?**

Ans: Action mappings are configured in the configuration file struts-config.xml under the tag <action-mapping> as follows:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <pre><action-mappings>  <action path="/login"  type="login.loginAction"  name="loginForm"  input="/login.jsp"  scope="request"  validate="true">  <forward name="success" path="/index.jsp"/>  <forward name="failure" path="/login\_error.jsp"/>  </action>  </action-mappings> |

**Q45. When should be opt for Struts Framework?**

Ans: Struts should be used when any or some of the following conditions are true:

* A highly robust enterprise level application development is required.
* A reusable, highly configurable application is required.
* A loosely coupled, MVC based application is required with clear segregation of different layers.

**Q46. Why ActionServlet is singleton in Struts?**

Ans: In Struts framework, actionServlet acts as a controller and all the requests made by users are controlled by this controller. ActionServlet is based on singleton design patter as only one object needs to be created for this controller class. Multiple threads are created later for each user request.

**Q47. What are the steps required for setting up validator framework in Struts?**

Ans: Following Steps are required to setup validator framework in Struts: – ***Wrong Spelling***

1. In WEB-INF directory place valdator-rules.xml and validation.xml files.
2. Enable validation plugin in struts-config.xml files by adding following:

[?](http://www.gointerviews.com/top-50-struts-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4 | <plug-in className="org.apache.struts.validator.ValidatorPlugIn">   <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,   /WEB-INF/validation.xml"/>   </plug-in> |

**Q48. Which technologies can be used at View Layer in Struts?**

Ans: In Struts, we can use any of the following technologies in view layer:

* JSP
* HTML
* XML/XSLT
* WML Files
* Velocity Templates
* Servlets

**Q49. What are the conditions for actionForm to work correctly?**

Ans: ActionForm must fulfill following conditions to work correctly:

* It must have a no argument constructor.
* It should have public getter and setter methods for all its properties.

**Q50.  Which library is provided by Struts for form elements like check boxes, text boxes etc?**

Ans: Struts provide HTML Tags library which can be used for adding form elements like text fields, text boxes, radio buttons etc.

**Q1. What is the difference between an Inner Class and a Sub-Class?**

Ans: An Inner class is a class which is nested within another class. An Inner class has access rights for the class which is nesting it and it can access all variables and methods defined in the outer class.

A sub-class is a class which inherits from another class called super class. Sub-class can access all public and protected methods and fields of its super class.

**Q2. What are the various access specifiers for Java classes?**

Ans: In Java, access specifiers are the keywords used before a class name which defines the access scope. The types of access specifiers for classes are:

1. Public : Class,Method,Field is accessible from anywhere.

2. Protected:Method,Field can be accessed from the same class to which they belong or from the sub-classes,and from the class of same package,but not from outside.

3. Default: Method,Field,class can be accessed only from the same package and not from outside of it’s native package.

4. Private: Method,Field can be accessed from the same class to which they belong.

**Q3. What’s the purpose of Static methods and static variables?**

Ans: When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.

**Q4. What is data encapsulation and what’s its significance?**

Ans: Encapsulation is a concept in Object Oriented Programming for combining properties and methods in a single unit.

Encapsulation helps programmers to follow a modular approach for software development as each object has its own set of methods and variables and serves its functions independent of other objects. Encapsulation also serves data hiding purpose.

**Q5. What is a singleton class? Give a practical example of its usage.**

A singleton class in java can have only one instance and hence all its methods and variables belong to just one instance. Singleton class concept is useful for the situations when there is a need to limit the number of objects for a class.

The best example of singleton usage scenario is when there is a limit of having only one connection to a database due to some driver limitations or because of any licensing issues.

**Q6. What are Loops in Java? What are three types of loops?**

[](http://cdn.gointerviews.com/wp-content/uploads/2012/03/Java-Interview-Questions.png)

Ans: Looping is used in programming to execute a statement or a block of statement repeatedly. There are three types of loops in Java:

1. For Loops

For loops are used in java to execute statements repeatedly for a given number of times. For loops are used when number of times to execute the statements is known to programmer.

1. While Loops

While loop is used when certain statements need to be executed repeatedly until a condition is fulfilled. In while loops, condition is checked first before execution of statements.

1. Do While Loops

Do While Loop is same as While loop with only difference that condition is checked after execution of block of statements. Hence in case of do while loop, statements are executed at least once.

**Q7: What is an infinite Loop? How infinite loop is declared?**

Ans: An infinite loop runs without any condition and runs infinitely. An infinite loop can be broken by defining any breaking logic in the body of the statement blocks.

Infinite loop is declared as follows:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | For (;;)    {    // Statements to execute    // Add any loop breaking logic    } |

**Q8.  What is the difference between continue and break statement?**

Ans: break and continue are two important keywords used in Loops. When a break keyword is used in a loop, loop is broken instantly while when continue keyword is used, current iteration is broken and loop continues with next iteration.

In below example, Loop is broken when counter reaches 4.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | For (counter=0;counter  System.out.println(counter);    If (counter==4) {    Break;}    } |

In the below example when counter reaches 4, loop jumps to next iteration and any statements after the continue keyword are skipped for current iteration.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | For (counter=0;counter  System.out.println(counter);    If (counter==4) {    continue;    }    System.outprintln("This will not get printed when counter is 4");    } |

**Q9. What is the difference between double and float variables in Java?**

Ans: In java, float takes 4 bytes in memory while Double takes 8 bytes in memory. Float is single precision floating point decimal number while Double is double precision decimal number.

**Q10.  What is Final Keyword in Java? Give an example.**

Ans: In java, a constant is declared using the keyword Final. Value can be assigned only once and after assignment, value of a constant can’t be changed.

In below example, a constant with the name const\_val is declared and assigned avalue:

Private Final int const\_val=100

When a method is declared as final,it can be overridden by the subclasses.This method are faster than any other method,because they are resolved at complied time.

When a class is declares as final,it cannot be subclassed. Example String,Integer and other wrapper classes.

**Q11. What is ternary operator? Give an example.**

Ans: Ternary operator , also called conditional operator is used to decide which value to assign to a variable based on a Boolean value evaluation. It’s denoted as ?

In the below example, if rank is 1, status is assigned a value of “Done” else “Pending”.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class conditionTest {   public static void main(string args[]) {       String status;       int rank;       status= (rank == 1) ? "Done": "Pending";    }  } |

**Q12: What are 6 different types of operators in Java?**

Ans: In java, operators can be classified in following six types:

* Arithmetic Operators

Used for arithmetic calculations. Example are **+,-,\*,/,%,++,–**

* Relational Operators

Used for relational comparison. E.g. **==,!=, >,<,<=,>=**

* Bitwise operators

Used for bit by bit operations. E.g. **&,|,^,~**

* Logical Operators

Used for logical comparisons. E.g. **&&,||,!**

* Assignment Operators

Used for assigning values to variables. E.g. **=,+=,-=,\*=,/=**

**Q13. What is default switch case? Give example.**

Ans: In a switch statement, default case is executed when no other switch condition matches. Default case is an optional case .  
It can be declared only once all other switch cases have been coded.

In the below example, when score is not 1 or 2, default case is used.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class switchExample {      int score=4;  public static void main(String args[]) {     switch (score) {  case 1:  System.out.println("Score is 1");  break;  case 2:  system.out.println("Score is 2");  break;  default:  System.out.println("Default Case");  }    }    } |

**Q14.  What’s the base class in Java from which all classes are derived?**

Ans: java.lang.object

**Q15. Can main() method in Java can return any data?**

Ans: In java, main() method can’t return any data and hence, it’s always declared with a void return type.

**Q16. What are Java Packages? What’s the significance of packages?**

Ans: In Java, package is a collection of classes and interfaces which are bundled together as they are related to each other. Use of packages helps developers to modularize the code and group the code for proper re-use. Once code has been packaged in Packages, it can be imported in other classes and used.

**Q17.  Can we declare a class as Abstract without having any abstract method?**

Ans: Yes we can create an abstract class by using abstract keyword before class name even if it doesn’t have any abstract method. However, if a class has even one abstract method, it must be declared as abstract otherwise it will give an error.

**Q18. What’s the difference between an Abstract Class and Interface in Java?**

Ans: The primary difference between an abstract class and interface is that an interface can only possess declaration of public static methods with no concrete implementation while an abstract class can have members with any access specifiers (public, private etc) with or without concrete implementation.

Another key difference in the use of abstract classes and interfaces is that a class which implements an interface must implement all the methods of the interface while a class which inherits from an abstract class doesn’t require implementation of all the methods of its super class.

A class can implement multiple interfaces but it can extend only one abstract class.

**Q19. What are the performance implications of Interfaces over abstract classes?**

Ans:  Interfaces are slower in performance as compared to abstract classes as extra indirections are required for interfaces. Another key factor for developers to take into consideration is that any class can extend only one abstract class while a class can implement many interfaces.

Use of interfaces also puts an extra burden on the developers as any time an interface is implemented in a class; developer is forced to implement each and every method of interface.

**Q20. Does Importing a package imports its sub-packages as well in Java?**

Ans: In java, when a package is imported, its sub-packages aren’t imported and developer needs to import them separately if required.

For example, if a developer imports a package university.\*, all classes in the package named university are loaded but no classes from the sub-package are loaded. To load the classes from its sub-package ( say department), developer has to import it explicitly as follows:

Import university.department.\*

**Q21. Can we declare the main method of our class as private?**

Ans: In java, main method must be public static in order to run any application correctly. If main method is declared as private, developer won’t get any compilation error however, it will not get executed and will give a runtime error.

**Q22.  How can we pass argument to a function by reference instead of pass by value?**

Ans:  In java, we can pass argument to a function only by value and not by reference.

**Q23. How an object is serialized in java?**

Ans: In java, to convert an object into byte stream by serialization, an interface with the name Serializable is implemented by the class. All objects of a class implementing serializable interface get serialized and their state is saved in byte stream.

**Q24. When we should use serialization?**

Ans: Serialization is used when data needs to be transmitted over the network. Using serialization, object’s state is saved and converted into byte stream .The  byte stream is transferred over the network and the object is re-created at destination.

**Q25. Is it compulsory for a Try Block to be followed by a Catch Block in Java for Exception handling?**

Ans: Try block needs to be followed by either Catch block or Finally block or both. Any exception thrown from try block needs to be either caught in the catch block or else any specific tasks to be performed before code abortion are put in the Finally block.

**Q26. Is there any way to skip Finally block of exception even if some exception occurs in the exception block?**

Ans:  If an exception is raised in Try block, control passes to catch block if it exists otherwise to finally block. Finally block is always executed when an exception occurs and the only way to avoid execution of any statements in Finally block is by aborting the code forcibly by writing following line of code at the end of try block:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1 | System.exit(0); |

**Q27. When the constructor of a class is invoked?**

Ans: The constructor of a class is invoked every time an object is created with new keyword.

For example, in the following class two objects are created using new keyword and hence, constructor is invoked two times.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class const\_example {    const\_example() {    System.out.println("Inside constructor");    }    Public static void main(String args[]) {    const\_example c1=new const\_example();    const\_example c2=new const\_example();    }    } |

**Q28. Can a class have multiple constructors?**

Ans: Yes, a class can have multiple constructors with different parameters. Which constructor gets used for object creation depends on the arguments passed while creating the objects.

**Q29. Can we override static methods of a class?**

Ans: We cannot override static methods. Static methods belong to a class and not to individual objects and are resolved at the time of compilation (not at runtime).Even if we try to override static method,we will not get an complitaion error,nor the impact of overriding when running the code.

**Q30. In the below example, what will be the output?**

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | public class superclass {    public void displayResult() {    System.out.println("Printing from superclass");    }    }    public class subclass extends superclass {    public void displayResult() {    System.out.println("Displaying from subClass");    super.displayResult();    }    public static void main(String args[]) {    subclass obj=new subclass();    obj.displayResult();    }    } |

**Ans**: Output will be:

Displaying from subclass

Displaying from superclass

**Q31. Is String a data type in java?**

Ans: String is not a primitive data type in java. When a string is created in java, it’s actually an object of Java.Lang.String class that gets created. After creation of this string object, all built-in methods of String class can be used on the string object.

**Q32.  In the below example, how many String Objects are created?**

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | String s1="I am Java Expert";    String s2="I am C Expert";    String s3="I am Java Expert"; |

Ans: In the above example, two objects of Java.Lang.String class are created. s1 and s3 are references to same object.

**Q33. Why Strings in Java are called as Immutable?**

Ans: In java, string objects are called immutable as once value has been assigned to a string, it can’t be changed and if changed, a new object is created.

In below example, reference str refers to a string object having value “Value one”.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1 | String str="Value One"; |

When a new value is assigned to it, a new String object gets created and the reference is moved to the new object.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1 | str="New Value"; |

**Q34. What’s the difference between an array and Vector?**

Ans: An array groups data of same primitive type and is static in nature while vectors are dynamic in nature and can hold data of different data types.

**Q35. What is multi-threading?**

Ans: Multi threading is a programming concept to run multiple tasks in a concurrent manner within a single program. Threads share same process stack and running in parallel. It  helps  in performance improvement of any program.

**Q36. Why Runnable Interface is used in Java?**

Ans: Runnable interface is used in java for implementing multi threaded applications. Java.Lang.Runnable interface is implemented by a class to support multi threading.

**Q37. What are the two ways of implementing multi-threading in Java?**

Ans: Multi threaded applications can be developed in Java by using any of the following two methodologies:

1. By using Java.Lang.Runnable Interface. Classes implement this interface to enable multi threading. There is a Run() method in this interface which is implemented.

2. By writing a class that extend Java.Lang.Thread class.

**Q38. When a lot of changes are required in data, which one should be a preference to be used? String or StringBuffer?**

Ans: Since StringBuffers are dynamic in nature and we can change the values of StringBuffer objects unlike String which is immutable, it’s always a good choice to use StringBuffer when data is being changed too much. If we use String in such a case, for every data change a new String object will be created which will be an extra overhead.

**Q39. What’s the purpose of using Break in each case of Switch Statement?**

Ans: Break is used after each case (except the last one) in a switch so that code breaks after the valid case and doesn’t flow in the proceeding cases too.

If break isn’t used after each case, all cases after the valid case also get executed resulting in wrong results.

**Q40.  How garbage collection is done in Java?**

Ans: In java, when an object is not referenced any more, garbage collection takes place and the object is destroyed automatically. For automatic garbage collection java calls either System.gc() method or Runtime.gc() method.

**Q41. How we can execute any code even before main method?**

Ans: If we want to execute any statements before even creation of objects at load time of class, we can use a static block of code in the class. Any statements inside this static block of code will get executed once at the time of loading the class even before creation of objects in the main method.

**Q42. Can a class be a super class and a sub-class at the same time? Give example.**

Ans: If there is a hierarchy of inheritance used, a class can be a super class for another class and a sub-class for another one at the same time.

In the example below, continent class is sub-class of world class and it’s super class of country class.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class world {    ..........    }    public class continenet extends world {    ............    }    public class country extends continent {    ......................    } |

**Q43.  How objects of a class are created if no constructor is defined in the class?**

Ans: Even if no explicit constructor is defined in a java class, objects get created successfully as a default  constructor is implicitly used for object creation. This constructor has no parameters.

**Q44.  In multi-threading how can we ensure that a resource isn’t used by multiple threads simultaneously?**

Ans: In multi-threading, access to the resources which are shared among multiple threads can be controlled by using the concept of synchronization. Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

**Q45. Can we call the constructor of a class more than once for an object?**

Ans:  Constructor is called automatically when we create an object using new keyword. It’s called only once for an object at the time of object creation and hence, we can’t invoke the constructor again for an object after its creation.

**Q46. There are two classes named classA and classB. Both classes are in the same package. Can a private member of classA can be accessed by an object of classB?**

Ans: Private members of a class aren’t accessible outside the scope of that class and any other class even in the same package can’t access them.

**Q47. Can we have two methods in a class with the same name?**

Ans: We can define two methods in a class with the same name but with different number/type of parameters. Which method is to get invoked will depend upon the parameters passed.

For example in the class below we have two print methods with same name but different parameters. Depending upon the parameters, appropriate one will be called:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | public class methodExample {    public void print() {    System.out.println("Print method without parameters.");    }    public void print(String name) {    System.out.println("Print method with paramter");    }    public static void main(String args[]) {    methodExample obj1=new methodExample();    obj1.print();    obj1.print("xx");    }    } |

**Q48. How can we make copy of a java object?**

Ans: We can use the concept of cloning to create copy of an object. Using clone, we create copies with the actual state of an object.

Clone() is a method of Cloneable interface and hence, Cloneable interface needs to be implemented for making object copies.

**Q49. What’s the benefit of using inheritance?**

Ans: Key benefit of using inheritance is reusability of code as inheritance enables sub-classes to reuse the code of its super class. Polymorphism (Extensibility ) is another great benefit which allow new functionality to be introduced without effecting existing derived classes.

**Q50.  What’s the default access specifier for variables and methods of a class?**

Ans: Default access specifier for variables and method is package protected i.e variables and class is available to any other class but in the same package,not outside the package.

**Q51. Give an example of use of Pointers in Java class.**

Ans: There are no pointers in Java. So we can’t use concept of pointers in Java.

**Q52.  How can we restrict inheritance for a class so that no class can be inherited from it?**

Ans: If we want a class not to be extended further by any class, we can use the keyword **Final**with the class name.

In the following example, Stone class is Final and can’t be extend

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <pre><em>  </em>public Final Class Stone {    // Class methods and Variables    } |

**Q53. What’s the access scope of Protected Access specifier?**

Ans: When a method or a variable is declared with Protected access specifier, it becomes accessible in the same class,any other class of the same package as well as a sub-class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modifier** | **Class** | **Package** | **Subclass** | **World** |
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| no modifier | Y | Y | N | N |
| private | Y | N | N | N |
| Access Levels | | | | |

**Q54. What’s difference between Stack and Queue?**

Ans: Stack and Queue both are used as placeholder for a collection of data. The primary difference between a stack and a queue is that stack is based on Last in First out (LIFO) principle while a queue is based on FIFO (First In First Out) principle.

**Q55. In java, how we can disallow serialization of variables?**

Ans: If we want certain variables of a class not to be serialized, we can use the keyword**transient**while declaring them. For example, the variable trans\_var below is a transient variable and can’t be serialized:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class transientExample {    private transient trans\_var;    // rest of the code    } |

**Q56.  How can we use primitive data types as objects?**

Ans: Primitive data types like int can be handled as objects by the use of their respective wrapper classes. For example, Integer is a wrapper class for primitive data type int. We can apply different methods to a wrapper class, just like any other object.

**Q57. Which types of exceptions are caught at compile time?**

Ans: Checked exceptions can be caught at the time of program compilation. Checked exceptions must be handled by using try catch block in the code in order to successfully compile the code.

**Q58. Describe different states of a thread.**

Ans: A thread in Java can be in either of the following states:

* Ready: When a thread is created, it’s in Ready state.
* Running: A thread currently being executed is in running state.
* Waiting: A thread waiting for another thread to free certain resources is in waiting state.
* Dead: A thread which has gone dead after execution is in dead state.

**Q59. Can we use a default constructor of a class even if an explicit constructor is defined?**

Ans: Java provides a default no argument constructor if no explicit constructor is defined in a Java class. But if an explicit constructor has been defined, default constructor can’t be invoked and developer can use only those constructors which are defined in the class.

**Q60. Can we override a method by using same method name and arguments but different return types?**

Ans: The basic condition of method overriding is that method name, arguments as well as return type must he exactly same as is that of the method being overridden.  Hence using a different return type doesn’t override a method.

**Q61.What will be the output of following piece of code?**

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class operatorExample {    public static void main(String args[]) {    int x=4;    System.out.println(x++);    }    } |

Ans: In this case postfix ++ operator is used which first returns the value and then increments. Hence it’s output will be 4.

**Q61. A person says that he compiled a java class successfully without even having a main method in it? Is it possible?**

Ans: main method is an entry point of Java class and is required for execution of the program however; a class gets compiled successfully even if it doesn’t have a main method. It can’t be run though.

**Q62.  Can we call a non-static method from inside a static method?**

Ans: Non-Static methods are owned by objects of a class and have object level scope and in order to call the non-Static methods from a static block (like from a static main method), an object of the class needs to be created first. Then using object reference, these methods can be invoked.

**Q63. What are the two environment variables that must be set in order to run any Java programs?**

Ans: Java programs can be executed in a machine only once following two environment variables have been properly set:

1. PATH variable
2. CLASSPATH variable

**Q64. Can variables be used in Java without initialization?**

Ans: In Java, if a variable is used in a code without prior initialization by a valid value, program doesn’t compile and gives an error as no default value is assigned to variables in Java.

**Q65. Can a class in Java be inherited from more than one class?**

Ans: In Java, a class can be derived from only one class and not from multiple classes. Multiple inheritances is not supported by Java.

**Q66. Can a constructor have different name than a Class name in Java?**

Ans: Constructor in Java must have same name as the class name and if the name is different, it doesn’t act as a constructor and compiler thinks of it as a normal method.

**Q67. What will be the output of Round(3.7) and Ceil(3.7)?**

Ans: Round(3.7) returns 3 while Ceil(3.7) returns 4.

**Q68: Can we use goto in Java to go to a particular line?**

Ans: In Java, there is not goto keyword and java doesn’t support this feature of going to a particular labeled line.

**Q69. Can a dead thread be started again?**

Ans: In java, a thread which is in dead state can’t be started again. There is no way to restart a dead thread.

**Q70. Is the following class declaration correct?**

**Ans:**

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | public abstract final class testClass {    // Class methods and variables    } |

Ans: The above class declaration is incorrect as an abstract class can’t be declared as Final.

**Q71. Is JDK required on each machine to run a Java program?**

Ans: JDK is development Kit of Java and is required for development only and to run a Java program on a machine, JDK isn’t required. Only JRE is required.

**Q72. What’s the difference between comparison done by equals method and == operator?**

Ans: In Java, equals() method is used to compare the contents of two string objects and returns true if the two have same value while == operator compares the references of two string objects.

In the following example, equals() returns true as the two string objects have same values. However == operator returns false as both string objects are referencing to different objects:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | public class equalsTest {    public static void main(String args[]) {    String srt1="Hello World";    String str2="Hello World";    If (str1.equals(str2))    {// this condition is true    System.out.println("str1 and str2 are equal in terms of values");    }    If (str1==str2) {    //This condition is not true    System.out.println("Both strings are referencing same object");    }    Else    {    // This condition is true    System.out.println("Both strings are referencing different objects");    }    }} |

**Q73. Is it possible to define a method in Java class but provide it’s implementation in the code of another language like C?**

Ans: Yes, we can do this by use of native methods. In case of native method based development, we define public static methods in our Java class without its implementation and then implementation is done in another language like C separately.

**Q74.  How destructors are defined in Java?**

Ans: In Java, there are no destructors defined in the class as there is no need to do so. Java has its own garbage collection mechanism which does the job automatically by destroying the objects when no longer referenced.

**Q75. Can a variable be local and static at the same time?**

Ans:  No a variable can’t be static as well as local at the same time. Defining a local variable as static gives compilation error.

**Q76. Can we have static methods in an Interface?**

Ans: Static methods can’t be overridden in any class while any methods in an interface are by default abstract and are supposed to be implemented in the classes being implementing the interface. So it makes no sense to have static methods in an interface in Java.

**Q77. In a class implementing an interface, can we change the value of any variable defined in the interface?**

Ans: No, we can’t change the value of any variable of an interface in the implementing class as all variables defined in the interface are by default public, static and Final and final variables are like constants which can’t be changed later.

**Q78. Is it correct to say that due to garbage collection feature in Java, a java program never goes out of memory?**

Ans: Even though automatic garbage collection is provided by Java, it doesn’t ensure that a Java program will not go out of memory as there is a possibility that creation of Java objects is being done at a faster pace compared to garbage collection resulting in filling of all the available memory resources.

So, garbage collection helps in reducing the chances of a program going out of memory but it doesn’t ensure that.

**Q79. Can we have any other return type than void for main method?**

Ans: No, Java class main method can have only void return type for the program to get successfully executed.

Nonetheless , if you absolutely must return a value to at the completion of main method , you can use System.exit(int status)

**Q80. I want to re-reach and use an object once it has been garbage collected. How it’s possible?**

Ans: Once an object has been destroyed by garbage collector, it no longer exists on the heap and it can’t be accessed again. There is no way to reference it again.

**Q81. In Java thread programming, which method is a must implementation for all threads?**

Ans: Run() is a method of Runnable interface that must be implemented by all threads.

**Q82. I want to control database connections in my program and want that only one thread should be able to make database connection at a time. How can I implement this logic?**

Ans: This can be implemented by use of the concept of synchronization. Database related code can be placed in a method which hs **synchronized**keyword so that only one thread can access it at a time.

**Q83. How can an exception be thrown manually by a programmer?**

Ans: In order to throw an exception in a block of code manually, **throw** keyword is used. Then this exception is caught and handled in the catch block.

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public void topMethod(){   try{   excMethod();   }catch(ManualException e){ }   }    public void excMethod{   String name=null;   if(name == null){   throw (new ManualException("Exception thrown manually ");   }   } |

**Q84.  I want my class to be developed in such a way that no other class (even derived class) can create its objects. How can I do so?**

Ans: If we declare the constructor of a class as private, it will not be accessible by any other class and hence, no other class will be able to instantiate it and formation of its object will be limited to itself only.

**Q85. How objects are stored in Java?**

Ans: In java, each object when created gets a memory space from a heap. When an object is destroyed by a garbage collector, the space allocated to it from the heap is re-allocated to the heap and becomes available for any new objects.

**Q86. How can we find the actual size of an object on the heap?**

Ans: In java, there is no way to find out the exact size of an object on the heap.

**Q87. Which of the following classes will have more memory allocated?**

**Class A: Three methods, four variables, no object**

**Class B: Five methods, three variables, no object**

Ans:  Memory isn’t allocated before creation of objects. Since for both classes, there are no objects created so no memory is allocated on heap for any class.

**Q88. What happens if an exception is not handled in a program?**

Ans: If an exception is not handled in a program using try catch blocks, program gets aborted and no statement executes after the statement which caused exception throwing.

**Q89.  I have multiple constructors defined in a class. Is it possible to call a constructor from another constructor’s body?**

Ans: If a class has multiple constructors, it’s possible to call one constructor from the body of another one using **this()**.

**Q90. What’s meant by anonymous class?**

Ans: An anonymous class is a class defined without any name in a single line of code using new keyword.

For example, in below code we have defined an anonymous class in one line of code:

[?](http://www.gointerviews.com/top-100-core-java-interview-questions/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | public java.util.Enumeration testMethod()    {    return new java.util.Enumeration()    {    @Override    public boolean hasMoreElements()    {    // TODO Auto-generated method stub    return false;    }    @Override    public Object nextElement()    {    // TODO Auto-generated method stub    return null;    }    } |

**Q91. Is there a way to increase the size of an array after its declaration?**

Ans: Arrays are static and once we have specified its size, we can’t change it. If we want to use such collections where we may require a change of size ( no of items), we should prefer vector over array.

**Q92. If an application has multiple classes in it, is it okay to have a main method in more than one class?**

Ans:  If there is main method in more than one classes in a java application, it won’t cause any issue as entry point for any application will be a specific class and code will start from the main method of that particular class only.

**Q93. I want to persist data of objects for later use. What’s the best approach to do so?**

Ans: The best way to persist data for future use is to use the concept of serialization.

**Q94. What is a Local class in Java?**

Ans: In Java, if we define a new class inside a particular block, it’s called a local class. Such a class has local scope and isn’t usable outside the block where its defined.

**Q95. String and StringBuffer both represent String objects. Can we compare String and StringBuffer in Java?**

Ans: Although String and StringBuffer both represent String objects, we can’t compare them with each other and if we try to compare them, we get an error.

**Q96. Which API is provided by Java for operations on set of objects?**

Ans: Java provides a Collection API which provides many useful methods which can be applied on a set of objects. Some of the important classes provided by Collection API include ArrayList, HashMap, TreeSet and TreeMap.

**Q97. Can we cast any other type to Boolean Type with type casting?**

Ans: No, we can neither cast any other primitive type to Boolean data type nor can cast Boolean data type to any other primitive data type.

**Q98.  Can we use different return types for methods when overridden?**

Ans: The basic requirement of method overriding in Java is that the overridden method should have same name,  and parameters.But a method can be overridden with a different return type as long as the new return type extends the original.

For example , method is returning a reference type.

Class B extends A{

A method(int x){

//original method

}

B method(int x){

//overridden method

}

}

**Q99. What’s the base class of all exception classes?**

Ans: In Java, **Java.Lang.throwable** is the super class of all exception classes and all exception classes are derived from this base class.

**Q100. What’s the order of call of constructors in inheritiance?**

Ans: In case of inheritance, when a new object of a derived class is created, first the constructor of the super class is invoked and then the constructor of the derived class is invoked.

### Q1. Explain Struts.

Struts is open source software used to develop java based web page.

* Struts uses Jakarta Packages, Java Servlets, JavaBeans, ResourceBundles, and XML
* Struts takes the help of Model View Controller (MVC) architecture. Where Model is referring to business or database, View is referring to the Page Design Code, and Controller is referring to navigational code.

### Q2. What is Action Class?

An Action class in the struts application is used to handle the request.

* It acts as interface or communication medium between the HTTP request coming to it and business logic used to develop the application.
* Action class consists of RequestProcessor which act as controller. This controller will choose the best action for each incoming request, generate the instance of that action and execute that action.
* This should be in thread-safe manner, because RequestProcessor uses the same instance for no. of requests at same time.

### Q3. What is Struts Validator Framework?

Struts Validator Framework enables us to validate the data of both client side and server side.

* When some data validation is not present in the Validator framework, then programmer can generate own validation logic, this User Defined Validation logic can be bind with Validation Framework.
* Validation Framework consist of two XML configuration Files:  
  o Validator-Rules.xml file  
  o Validation.xml file

### Q4. What is the need of Struts?

We need Struts in Java because of following reasons:

* Helps in creation and maintenance of the application.
* Make use of Model View Controller (MVC) architecture. Where Model is referring to business or database, View is referring to the Page Design Code, and Controller is referring to navigational code.
* Enables developer to make use of Jakarta Packages, Java Servlets, JavaBeans, ResourceBundles, and XML

### Q5. What are the classes used in Struts?

Struts Framework consists of following classes:

* **Action Servlets:** used to control the response for each incoming request.
* **Action Class:** used to handle the request.
* **Action Form:** it is java bean, used to referred to forms and associated with action mapping
* **Action Mapping:** used for mapping between object and action.
* **Action Forward:** used to forward the result from controller to destination.

### Q6. How exceptions are handled in Struts application?

Exceptions are handled in struts by using any one of the following two ways:

* **Programmatically handling:** In this exception are handled by using try and catch block in program. Using this programmer can define how to handle the situation when exception arises.
* **Declarative handling:** In this exception handling is done by using the XML file. Programmer defines the exception handling logic in the XML file. There are two ways of defining the exception handling logic in the XML file:  
  -Global Action Specific Exception Handler Definition.  
  -Local Action Specific Exception Handler Definition.

### Q7. What is MVC?

Model View Controller (MVC) is a design pattern used to perform changes in the application.

* **Model:** Model is referring to business or database. It stores the state of the application. Model has no knowledge of the View and Controller components.
* **View:** View is referring to the Page Design Code. It is responsible for the showing the result of the user’s query. View modifies itself when any changes in the model happen.
* **Controller:** Controller is referring to navigational code. Controller will chose the best action for each incoming request, generate the instance of that action and execute that action.

### Q8. Describe Validate() and reset() methods.

Validate() Method: this method is used to validate the properties after they are explored by the application.

* Validate method is Called before FormBean is handed to Action.
* This method returns a collection of ActionError.
* Syntax of Validate() Method:  
  public ActionErrors validate(ActionMapping mapping,HttpServletRequest request)

Reset() Method: this method is called by the Struts Framework with each request that uses the defined ActionForm.

* Used to reset all the data from the ActionForm
* Syntax of Reset() Method:  
  public void reset() {}

### Q9. What design patterns are used in Struts?

There are following types of design patterns are used in Struts:

* Service to Worker
* Dispatcher View
* Composite View (Struts Tiles)
* Front Controller
* View Helper
* Synchronizer Token

### Q10.What is the difference between session scope and request scope when saving FormBean?

The difference between session scope and request scope when saving FormBean are following:

* In Request Scope, values of FormBean are available to current request but in Session Scope, values of FormBean are available throughout the session.

### Q11.What is the different actions available in Struts?

The different kinds of actions in Struts are:

* ForwardAction
* IncludeAction
* DispatchAction
* LookupDispatchAction
* SwitchAction

### Q12.What is DispatchAction?

The DispatchAction enable the programmer to combine together related function or class.

* Using Dispatch Action programmer can combine the user related action into a single UserAction. like add user, delete user and update user
* DispatchAction execute the action based on the parameter value it receives from the user.

### Q13. How to use DispatchAction?

We can use the Dispatch Action we executing following steps:

* Create a class that extends DispatchAction.
* In a new class, add a method: method has the same signature as the execute() method of an Action class.
* Do not override execute() method.
* Add an entry to struts-config.xml

### Q14. .What is the difference between ForwardAction and IncludeAction?

The difference between ForwardAction and InculdeAction are:

* IncludeAction is used when any other action is going to intake that action whereas ForwardAction is used move the request from one resource to another resource.

### Q15.What is difference between LookupDispatchAction and DispatchAction?

The difference between LookupDispatchAction and DispatchAction are given below:

* LookupDispatchAction is the subclass of the DispatchAction
* Actual method that gets called in LookupDispatchAction whereas DispatchAction dispatches the action based on the parameter value.

### Q16.What is LookupDispatchAction?

The LookupDispatchAction class is a subclass of DispatchAction

* The LookupDispatchAction is used to call the actual method.
* For using LookupDispatchAction, first we should generate a subclass with a set of methods.
* It control the forwarding of the request to the best resource in its subclass
* It does a reverse lookup on the resource bundle to get the key and then gets the method whose name is associated with the key into the Resource Bundle.

### Q18. What is the use of ForwardAction?

The ForwardAction is used when we want to combine Struts with existing application.

* Used when we want to transfer the control form JSP to local server.
* Used to integrate with struts in order to take benefit of struts functionality, without writing the Servlets again.
* Use to forward a request to another resource in your application

### Q19.What is IncludeAction?

The IncludeAction is used to integrate the one action file in another action file.

* It is same as ForwardAction but the only difference is that the resource is present in HTTP response.
* Is used to combine the Struts functionality and control into an application that uses Servlets.
* Use the IncludeAction class to include another resource in the response to the request being processed.

### Q20. What are the various Struts tag libraries?

The various Struts tag libraries are:

* **HTML Tags:** used to create the struts input forms and GUI of web page.
* **Bean Tags:** used to access bean and their properties.
* **Logic Tags:** used to perform the logical operation like comparison
* **Template Tags:** used to changes the layout and view.
* **Nested Tags:** used to perform the nested functionality
* **Tiles Tags:** used to manages the tiles of the application

### Q21. What is the life cycle of ActionForm?

The lifecycle of ActionForm is as follows:

* Retrieve or Create Form Bean associated with Action
* "Store" FormBean in appropriate scope (request or session)
* Reset the properties of the FormBean
* Populate the properties of the FormBean
* Validate the properties of the FormBean
* Pass FormBean to Action

### Q22. What are the loop holes of Struts?

The drawbacks of Struts are following:

* Absence of backward flow mechanism.
* Only one single controller Servlets is used.
* Bigger learning curve
* Worst documentation
* No exception present in this framework
* Less transparent
* Rigid approach.
* With struts 1, embedding application into JSP can’t be prevented.
* Non-XML compliance of JSP syntax

### Q23. Difference between Html tags and Struts specific HTML Tags

Difference between HTML tag and Struts specific HTLM tags are:

* HTML tags are static in nature but Struts specific HTML tags are dynamic in nature.
* HTML tags are not User Defined whereas Struts tags can be user defined.
* HTML tags provide the different templates and themes to the programmer whereas Struts specific HTML tag provides the integrating the property value with the Formbean properties.
* HTML tags are integral part of Struts whereas Struts have HTML tag libraries.

### Q24. What is the difference between session scope and request scope when saving FormBean?

The difference between session scope and request scope when saving FormBean are following:

* In Request Scope, values of FormBean are available to current request but in Session Scope, values of FormBean are available throughout the session.

### Q25. How to display validation errors on JSP page?

**Validation error:** Validation error are those error which arises when user or client enters the invalid format data into the form. For this validation of data struts enables the programmer with the Validator() method which validates both the data from client side and the server side.

We can display all error in the JSP page by using the following syntax in the code.

SYNTAX: <html:error/>

### Q26. How to use forward action to restrict a strut application to MVC?

We can use the ForwarAction to restrict the Struts application to Model View Controller by following coding:

<global-forwards>   
<statements>  
<forward name="CitizenDetailsPage"   
path="/gotoCitizenDetails.do" />  
</global-forwards>   
  
<action-mappings>   
<statements>  
<action path=”/gotoCitizenDetails”   
parameter=”/CitizenDetails.jsp”   
type=”org.apache.struts.actions.ForwardAction” />   
</action-mappings>

### Q27. What is ActionMapping?

In action mapping is the mapping of the action performed by the user or client on the application.

-We specify the action class for a specific user’s action. Like we provide the path or URL and different view based on user event.  
-We can also define where control of the page deviate in case of validation error in the form.  
-We can include ActionMapping in code like this:  
<action-mappings>  
<action path="/a" type=myclasse.A name="myForm">   
<forward name="Login" path="/login.jsp"/>  
<forward name="error" path="/error.jsp"/>  
</action-mappings>

### Q28. What is role of Action Class?

An Action class in the struts application is used to handle the request.

* It acts as interface or communication medium between the HTTP request coming to it and business logic used to develop the application.
* Action class consists of RequestProcessor which act as controller. This controller will choose the best action for each incoming request, generate the instance of that action and execute that action.
* This should be in thread-safe manner, because RequestProcessor uses the same instance for no. of requests at same time.

### Q29. How to combine the Struts with Velocity Template?

We can combine Struts and Velocity template by performing following steps:

1. Set classpath to Velocity JARs  
2. Make web.xml file to identify the Velocity servlet.  
3. Select Velocity toolbox.xml in WEB-INF directory.  
4. Modify struts-config to point its views to Velocity templates instead of JSPs.  
5. Create a Velocity template for each page you want to render.

### Q31. In how many ways duplicate form submission can occurs?

The submission form can be duplicated by the any of the following ways:

* Using refresh button.
* By clicking submit button more than once before the server sent back the response.
* By clicking back navigation button present in browser.
* The browser is restores to submit the form again.
* By clicking multiple times on a transaction that is delayed than usual.

### Q32. What is the difference between Struts 1 and struts2?

The difference between struts1 and struts2 are below:

* Struts1 uses ActionServlet as Controller where as Struts2 uses Filter as a controller.
* Struts1 uses the ActionForm for mapping the JSP forms but in struts2 there no such ActionForm.
* Struts1 uses Validation() method to validate the data in the page forms where as struts 2 validation of data is ensure by Validator Framework.
* In Struts 1 the bean and logic tag libraries are often replaced by JSTL, but Struts 2 has such tag libraries that we don’t need to use JSTL.

### Q33. What are the steps used to setup dispatch action?

To setup the dispatch action the following steps are used:

* Create a subclass for DispatchAction.
* Create method for logical action and their related actions.
* Request Parameter is created for each action.
* Define ActionMapping.
* The JSP takes on the subclass defined for dispatch action method names as their values

### Q34. What is difference between Interceptors and Filters?

The difference between Interceptors and filter are below:

* Filters are based on Servlet Specification whereas Interceptors are based on Struts2.
* Filters are executed only when patter matches whereas Interceptors executes for all request qualifies for a front controller.
* Filters are not Configurable method calls whereas Interceptors methods can be configured.

### Q35. What are the Custom tags?

Custom Tags are User Defined Tags, which means that user can create those tags depending upon their need in the application.

* When a JSP page consisting of user- defined or custom tag is translated into a Servlet, the custom is also get translated into operation
* Help in fast development of the application due to custom tag reusability.

### Q36. What is the difference between empty default namespace and root namespace?

The difference between the empty default namespace and root name space are:

* When namespace attribute is not defined then it is referred to as Empty Default Namespace whereas when name space attribute is assign with forward slash(/) then it is referred to as Root Name Space.
* The root namespace must be matched.

### Q37. What is the difference between RequestAware and ServletRequestAware interface?

The difference between RequestAware and ServletRequestAware are:

* RequestAware enables programmer with the attributes in the Servlet Request as a map whereas ServletRequestAware enables programmer with HttpServletRequest object.
* ServletRequestAware are more flexible than RequestAware.
* ServletRequestAware makes action class highly coupled with Servlet environment which is not possible in RequestAware.

### Q38. What are inner class and anonymous class?

Inner class: classes that are defined within other classes.

* The nesting is a relationship performed between two different classes.
* An inner class can access private members and data.
* Inner classes have clearly two benefits:  
  o Name control   
  o Access control.  
  Anonymous class: Anonymous class is a class defined inside a method without a name.
* It is instantiated and declared in the same method.
* It does not have explicit constructors.

### Q39. What is struts.devMode?

The struts.devMode is used to make sure that framework is running in development mode or production mode by setting true or false. struts.devMode is set to false in production phase to reduce impact of performance. By default it is "false". It is used because of the following reasons:

* **Resource Reloading:** Resource bundle reload on every request
* **Modification:** struts.xml can be modified without restarting or redeploying the application
* **Error Handling:** The error occurs in the application will be reported, as oppose to production mode.

### Q40. What are action errors?

Action error: when user or client submits the incorrect or invalid data in the application, then these errors are known as Action error.

* Action errors are generated by the clients.
* Action error should be determined as soon as possible.

The impacts of such Action Error are:

* Wastage of server time and resources.
* Negative impact on code quality.

<http://careerride.com/Interview-Questions-Java-Struts.aspx>