**1. What are programmatic and declarative exception handling?**

Usually there are two ways in which you can catch the exceptions in struts:

**Programmatic Exception Handling :**

* In this approach the exceptions are caught using normal java language try/catch block.
* In this approach the flow of control is also maintained by the programs.
* The main drawback of the approach is the developer has to write the code for the flow of the application.

**Declarative Exception Handling :**

* Declarative Exception Handling is the way of handling Exceptions with the help of xml files so there is no need to write exception-handling code in the application.
* In this approach the exceptions are defined in the struts-config file and in case of the exception occurs the control is automatically passed to the appropriate error page.
* The biggest benefit of Declarative Exception Handling is if there is requirement to change the exception handling mechanism, changes can be made to the xml file, without recompilation of java code.

**2. Difference between HQL and Criteria Query in Hibernate**

|  |  |
| --- | --- |
| **HQL** | **Criteria Query** |
| HQL is to perform both select and non-select operations on the data | Criteria is only for selecting the data, we cannot perform non-select operations |
| Suitable for executing Static Queries | Suitable for executing Dynamic Queries |
| HQL doesn’t support pagination concept | we can achieve pagination with Criteria |
| HQL is executes faster than Criteria | Criteria used to take more time to execute then HQL |

**3. How to perform pagination using hibernate?**

Pagination can be done using Criteria query.

There are two methods of the Query interface for pagination.

* setFirstResult(int startPosition)
* setMaxResults(int maxResult)

eg: Query query = session.createQuery(hql);

query.setFirstResult(1);

query.setMaxResults(10);

**4. How to convert object from detached to persistent state?**

Detached objects can be attached to session by calling session.update() or session.saveOrUpdate() methods.

**5. Difference between Transient and Detached States:**

|  |  |
| --- | --- |
| **Transient objects** | **Detached objects** |
| Transient objects do not have association with the databases and session objects. | The detached object have corresponding entries in the database. |
| They are simple objects and not persisted to the database. | These are persistent and not connected to the Session object. |
| Once the last reference is lost, that means the object itself is lost and garbage collected. The commits and rollbacks will have no effects on these objects. | These objects have the synchronized data with the database when the session was closed and |
| They can become into persistent objects through the save method calls of Session object. | The detached object can be reattached after certain time to session in order to become persistent again. |

**6. Difference between load() and get()**

|  |  |
| --- | --- |
| **load()** | **get()** |
| When you call session.load() method, it will always return a “proxy” object. | When you call session.get() method, it will hit the database immediately and returns the original object |
| session.load(Student.class,new Integer(107)) will create a Student object in memory and will save the stdId as 107. Only while trying to fetch other properties such as stdName the actually query hit takes place. | session.get(Student.class,new Integer(107)) will immediately fetch the whole Student object from the dB corresponding to stdId 107 |
| If object [row] is not found in the database it will throw ObjectNotFoundException | If the row is not available in the database, it returns null |
| It should be used if you are sure that instance exists. | It should be used if you are not sure about the existence of instance. |

**7. Difference between save() and persist()**

* Actually the difference between hibernate save() and persist() methods is depends on generator class we are using.
* If our generator class is assigned, then there is no difference between save() and persist() methods.
* If we use other than assigned generator, and if call save() method it returns the primary key id value which is generated by hibernate whereas persist() will never give any value back

**8. How to make a immutable class in hibernate?**

* If you mark a class as mutable="false", class will be treated as an immutable class.
* By default, it is mutable="true".

**9. What is automatic dirty checking in hibernate?**

* The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction.
* It uses version number to check for dirty records (ie) it checks whether the version number hasn't changed between the time the version number was generated and before writing to dB.
* For legacy tables it checks for the time-stamp.

**10. What is lazy loading in hibernate?**

* Hibernate uses proxy object to support lazy loading.
* Lazy loading in hibernate loads the child objects on demand.
* Basically when you load data from tables, hibernate doesn’t load all the mapped objects. As soon as you reference a child or lookup object via getter methods, if the linked entity is not in the session cache, then the proxy code will go to the database and load the linked object
* Since Hibernate 3, lazy loading is enabled by default.

**11. What are the advantages of Hibernate over JDBC?**

* Hibernate removes a lot of boiler-plate code.
* Hibernate supports inheritance, associations and collections.
* Hibernate implicitly provides transaction management.
* JDBC API throws SQLException that is a checked exception. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception.
* Hibernate Query Language (HQL) is more object oriented and close to java programming language.
* Hibernate supports caching that is better for performance.

**12. What is hibernate configuration file and mapping file?**

* Hibernate configuration file contains database specific configurations and used to initialize SessionFactory.
* Hibernate mapping file is used to define the entity bean fields and database table column mappings.

**13. What is difference between openSession and getCurrentSession?**

* Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed.
* Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.

**14. What will happen if we don’t have no-args constructor in Entity bean?**

* Hibernate uses Reflection API to create instance of Entity beans, usually when you call get() or load() methods.
* The method Class.newInstance() is used for this and it requires no-args constructor.
* If you won’t have no-args constructor in entity beans, hibernate will fail to instantiate it and you will get HibernateException.

**15. What are the collection types in Hibernate?**

There are five collection types in hibernate used for one-to-many relationship mappings.

* Bag
* Set
* List
* Array
* Map

**16. Why we should not make Entity Class final?**

* Hibernate use proxy classes for lazy loading of data, only when it’s needed.
* This is done by extending the entity bean, if the entity bean will be final then lazy loading will not be possible.

**17. What is Named SQL Query?**

* Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.
* Hibernate Named Queries can be defined in Hibernate mapping files or through the use of JPA annotations @NamedQuery and @NamedNativeQuery.

**18. Which design patterns are used in Hibernate framework?**

Some of the design patterns used in Hibernate Framework are:

* **Domain Model Pattern** – An object model of the domain that incorporates both behavior and data.
* **Data Mapper** – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.
* **Proxy Pattern** for lazy loading
* **Factory pattern** in SessionFactory

**19. 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

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

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

**21. What is optimistic locking in hibernate?**

* It is a strategy where you read a record, make a note of the version numbers and before writing the record into dB it checks whether the version number has been changed.
* This is useful in finding the dirty records.

**22. How to define whether struts and spring have to do a particular job in web application?**

* This starts from registering ContextLoaderListener and StrutsPrepareAndExecuteFilter in web.xml.
* ContextLoaderListener takes init parameter 'contextConfigLocation' and is responsible for setup and start Spring WebApplicationContext.
* Now struts will take advantage of this context in spring related services specially in dependency injection.
* StrutsPrepareAndExecuteFilter look up struts.xml file in classpath and configure strut’s specific things like Action mappings, global forwards and other things defined in struts.xml file.

**23. How are the requests sent from client served?**

* When a client makes a request for a JSP or a Servlet, the request initially goes to the Web server.
* The Web server reads the special XML file the application server provides, and realizes that the request that came in should be sent to the appropriate servlet specified for processing.
* The servlet then servers the incoming request and provides the response which is then sent back to the web server for the client.

**24. How do we know that particular request comes from which client?**

The following are the keys to identify the client.

* Cookies
* IP address
* Browser fingerprinting (Use all browser data passed to the server)

**25. What is spring IoC?**

* IoC – Inversion of Control is a container that supports dependency injection.
* IoC pattern states that need not create your objects rather just describe how the objects should be created in runtime (via configuration file or annotation)

**26. What is DI?**

* DI – Dependency Injection is a pattern of injecting a class's dependencies into it at runtime.
* This is achieved by the use of interfaces.

**27. What is spring MVC?**

The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale, time zone and theme resolution as well as support for uploading files.

**28. Spring AOP**

* AOP – Aspect Oriented Programming enables cohesive development by separating application’s business logic from the system services.
* Some of the system services are transacting, error logging, security, etc.
* It is plug-able, easy to maintain and provides modularity.

**29. What is hibernate dialect?**

* Hibernate, as we know, is database independent.
* However, databases have proprietary extensions/native SQL variations, and set of SQL standard implementations.
* Hibernate uses "dialect" configuration to know which database you are using so that it can switch to the database specific SQL generator code wherever necessary.

**30. What are the scope of a bean?**

|  |  |
| --- | --- |
| **Scope** | **Description** |
| singleton | This scopes the bean definition to a single instance per Spring IoC container (default). |
| prototype | This scopes a single bean definition to have any number of object instances. |
| request | This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| session | This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| global-session | This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

**31. What is default scope of a bean?**

The default scope of a bean is singleton. This scopes the bean definition to a single instance per Spring IoC container.

**32. How to inject property file in spring container?**

Using the PropertyPlaceholderConfigurer class.

Eg: <bean

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

<property name="locations">

<list>

<value>/WEB-INF/resource/database.properties</value>

<value>/WEB-INF/resource/xmlEditor.properties</value>

</list>

</property></bean>

**33. How to do some operations before initialization of a bean in application context?**

Spring framework provide different ways through which we can provide post-initialization and pre-destroy methods in a spring bean.

* **By implementing InitializingBean and DisposableBean interfaces** - InitializingBean interface provide implementation of afterPropertiesSet() method for post-initialization and DisposableBean interface provide implementation of destroy() method for pre-destroy.
* **Providing init-method and destroy-method attribute** values for the bean in the spring bean configuration file. This is the recommended approach because of no direct dependency to spring framework and we can create our own methods.

Note that both post-init and pre-destroy methods should have no arguments but they can throw Exceptions.

Useful scenario: Sometimes we want to initialize resources in the bean classes, for example creating database connections or validating third party services at the time of initialization before any client request.

**34. How validation is done in struts2?**

There are three ways to perform validation in struts 2:

* By Custom Validation Here, we must implement the Validateable interface (or extend ActionSupport class) and provide the implementation of validate method.
* By Input Validation (built-in validators) Struts 2 provides a lot of predefined validators that can be used in struts 2 application to perform validation. (eg: requiredstring, stringlength, email, url and regex)
* By Ajax Validation (built-in validators with ajax) If we don't want to refresh the page, we can use jsonValidation interceptor to perform validation with ajax.

**35. How global exceptions are handled in struts2?**

* Can specify in the struts.xml how the framework should handle uncaught exceptions. The handling logic can apply to all actions (global exception handling) or to a specific action.
* To enable global exception handling you need to add two nodes to struts.xml: global-exception-mapping and global-results.

Eg: <global-results>

<result name="securityerror">/securityerror.jsp</result>

<result name="error">/error.jsp</result>

</global-results>

<global-exception-mappings>

<exception-mapping exception="org.apache.struts.register.exceptions.SecurityBreachException"

result="securityerror" />

<exception-mapping exception="java.lang.Exception" result="error" />

</global-exception-mappings>

* The global exception mapping node tells the Struts 2 framework what to do if an uncaught exception of the type specified (or a child of that type) is thrown by the the application.
* The global results mapping node relates the result value to a specific view page.