**J2EE-18**

**Q-1[A]ONE MARK:**

1]full form EIS- enterprise information system (EIS)

2]what is tomcat?

a. Tomcat is an application server that executes Java servlets and renders Java Server Page pages.

3]what is thin client and thick client?

a. a thin client is a web based application and most of the processing is done on the server side. A thick client is installed into the client side. It is still connected to the server, but most of the processing is done on client side

4]what is container?

a. A container is a component which can contain other components inside itself. It is also an instance of a subclass of java.

**Q-1[B]TWO MARK:**

1]explain Resultset.

a. The java.sql.ResultSet interface represents the result set of a database query.

A ResultSet object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a ResultSet object.

The methods of the ResultSet interface can be broken down into three categories −

Navigational methods: Used to move the cursor around.

Get methods: Used to view the data in the columns of the current row being pointed by the cursor.

Update methods: Used to update the data in the columns of the current row. The updates can then be updated in the underlying database as well.

2]explain callable and prepared statements.

a. Once a connection is obtained we can interact with the database. The JDBC *Statement, CallableStatement,* and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.

They also define methods that help bridge data type differences between Java and SQL data types used in a database.

The following table provides a summary of each interface's purpose to decide on the interface to use.

|  |  |
| --- | --- |
| **Interfaces** | **Recommended Use** |
| Statement | Use this for general-purpose access to your database. Useful when you are using static SQL statements at runtime. The Statement interface cannot accept parameters. |
| PreparedStatement | Use this when you plan to use the SQL statements many times. The PreparedStatement interface accepts input parameters at runtime. |
| CallableStatement | Use this when you want to access the database stored procedures. The CallableStatement interface can also accept runtime input parameters. |

**Q-1[C]THREE MARK:**

1]explain resultset meta data and database set meta data

a. Java ResultSetMetaData Interface

The metadata means data about data i.e. we can get further information from the data.

If you have to get metadata of a table like total number of column, column name, column type etc. , ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.

Commonly used methods of ResultSetMetaData interface

|  |  |
| --- | --- |
| **Method** | **Description** |
| public int getColumnCount()throws SQLException | it returns the total number of columns in the ResultSet object. |
| public String getColumnName(int index)throws SQLException | it returns the column name of the specified column index. |
| public String getColumnTypeName(int index)throws SQLException | it returns the column type name for the specified index. |
| public String getTableName(int index)throws SQLException | it returns the table name for the specified column index. |

How to get the object of ResultSetMetaData:

|  |
| --- |
| The getMetaData() method of ResultSet interface returns the object of ResultSetMetaData.  Syntax: |

**public** ResultSetMetaData getMetaData()**throws** SQLException

Java DatabaseMetaData interface

DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views etc.

Commonly used methods of DatabaseMetaData interface

* **public String getDriverName()throws SQLException:**it returns the name of the JDBC driver.
* **public String getDriverVersion()throws SQLException:**it returns the version number of the JDBC driver.
* **public String getUserName()throws SQLException:**it returns the username of the database.
* **public String getDatabaseProductName()throws SQLException:**it returns the product name of the database.
* **public String getDatabaseProductVersion()throws SQLException:**it returns the product version of the database.
* **public ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)throws SQLException:**it returns the description of the tables of the specified catalog. The table type can be TABLE, VIEW, ALIAS, SYSTEM TABLE, SYNONYM etc.

How to get the object of DatabaseMetaData:

The getMetaData() method of Connection interface returns the object of DatabaseMetaData. Syntax:

**public** DatabaseMetaData getMetaData()**throws** SQLException

2]types of JDBC database drivers

a. JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

**Type-1 driver**

Type-1 driver or JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

* As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.
* The ODBC bridge driver is needed to be installed in individual client machines.
* Type-1 driver isn’t written in java, that’s why it isn’t a portable driver.
* This driver software is built-in with JDK so no need to install separately.
* It is a database independent driver.

**Type-2 driver**

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API, that’s why data transfer is much more secure as compared to type-1 driver.

* Driver needs to be installed separately in individual client machines
* The Vendor client library needs to be installed on client machine.
* Type-2 driver isn’t written in java, that’s why it isn’t a portable driver
* It is a database dependent driver.

**Type-3 driver**

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

* Type-3 drivers are fully written in Java, hence they are portable drivers.
* No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.
* Network support is required on client machine.
* Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.
* Switch facility to switch over from one database to another database.

**Type-4 driver**

Type-4 driver is also called native protocol driver. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

* Does not require any native library and Middleware server, so no client-side or server-side installation.
* It is fully written in Java language, hence they are portable drivers.

**Q-1[D]FIVE MARK:**

1]explain N-tier architecture

a.

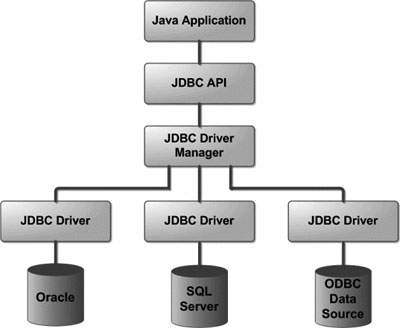
2]JDBC architecture

a. The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers −

JDBC API: This provides the application-to-JDBC Manager connection.

JDBC Driver API: This supports the JDBC Manager-to-Driver Connection.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application −



Common JDBC Components

The JDBC API provides the following interfaces and classes −

* **DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
* **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
* **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
* **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
* **SQLException:** This class handles any errors that occur in a database application.

**Q-2[A]ONE MARK:**

1]full form RMI- Remote Method Invocation (RMI)

2]explain servlet context interface

a. Defines a set of methods that a servlet uses to communicate with its servlet container,

3]explain URL rewriting.

a. Url rewriting is a process of appending or modifying any url structure while loading a page.

4]explain deployment descriptor

a. web.xml is known as deployment descriptor. When a request comes web server uses web.xml file to map the URL of the request to the specific code that handle the request.

**Q-2[A]TWO MARK:**

1]explain Request dispatcher interface

a. The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

The RequestDispatcher interface provides two methods. They are:

public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

2]explain servlet collobroation

a. The exchange of information among servlets of a particular Java web application is known as Servlet Collaboration. This enables passing/sharing information from one servlet to the other through method invocations.

\*The servlet api provides two interfaces namely:

javax.servlet.RequestDispatcher

javax.servlet.http.HttpServletResponse

These two interfaces include the methods responsible for achieving the objective of sharing information between servlets.

\* It provides the following two methods:

(a)public void forward(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The forward() method is used to transfer the client request to another resource (HTML file, servlet, jsp etc). When this method is called, the control is transferred to the next resource called. On the other hand, the include() method is used to include the content of the calling file into the called file. After calling this method, the control remains with the calling resource, but the processed output is included into the called resource.

(b)public void include(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The include() method is used to include the contents of the calling resource into the called one. When this method is called, the control still remains with the calling resource. It simply includes the processed output of the calling resource into the called one.

**Q-2[C]THREE MARK:**

1]explain single thread model

a. The servlet programmer should implement SingleThreadModel interface to ensure that servlet can handle only one request at a time. It is a marker interface, means have no methods.

This interface does not prevent synchronization problems that result from servlets accessing shared resources such as static class variables or classes outside the scope of the servlet.

2]explain cookie and session

a. **Session**

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

A session ends when the user closes the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

**Cookies**

Cookies are text files stored on the client computer and they are kept of use tracking purpose. Server script sends a set of cookies to the browser. For example name, age, or identification number etc. The browser stores this information on a local machine for future use.

When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

**Q-2[D]FIVE MARK:**

1]explain stub and skeleton

a. stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

2]explain servlet life cycle

a. The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

public void init(ServletConfig config) throws ServletException

4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

**Q-3[A]ONE MARK:**

1]<% = is used for-

2]explain scrptlet

a. In JavaServer Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in the HTML-like JSP code. The scriptlet is everything inside the <% %> tags.

A scriptlet tag is used to execute java source code in JSP.

3]what is use of <jsp:include>.

a. The jsp:include action tag is used to include the content of another resource it may be jsp, html or servlet.

4]give the tag syntax of page directive element

a. <%@ page attribute="value" %>

**Q-3[B]TWO MARK:**

1]explain JSP implicit objects.

a. There are 9 jsp implicit objects. These objects are created by the web container that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

|  |  |
| --- | --- |
| **Object** | **Type** |
| out | JspWriter |
| request | HttpServletRequest |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| page | Object |
| exception | Throwable |

2]explain scope of JSP variables

a.

**Q-3[C]THREE MARK:**

1]how can JSP handle the expections?

a. The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

1. By **errorPage** and **isErrorPage** attributes of page directive
2. By **<error-page>** element in web.xml file

### Example of exception handling in jsp by the elements of page directive

In this case, you must define and create a page to handle the exceptions, as in the error.jsp page. The pages where may occur exception, define the errorPage attribute of page directive, as in the process.jsp page.

There are 3 files:

* index.jsp for input values
* process.jsp for dividing the two numbers and displaying the result
* error.jsp for handling the exception

#### index.jsp

1. <form action="process.jsp">
2. No1:<input type="text" name="n1" /><br/><br/>
3. No1:<input type="text" name="n2" /><br/><br/>
4. <input type="submit" value="divide"/>
5. </form>

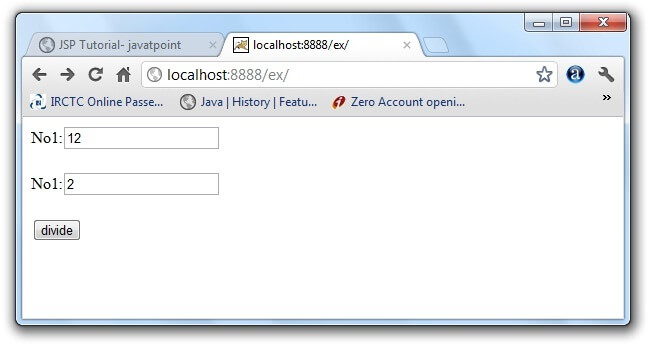
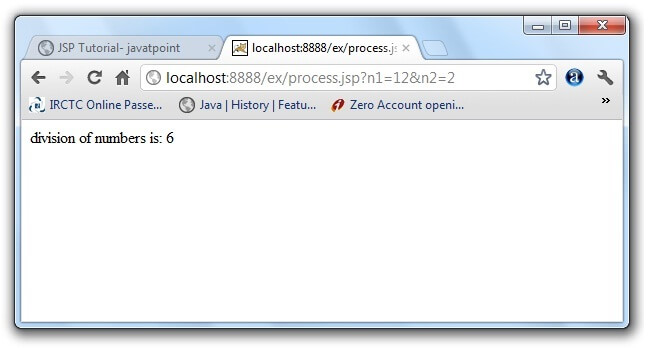
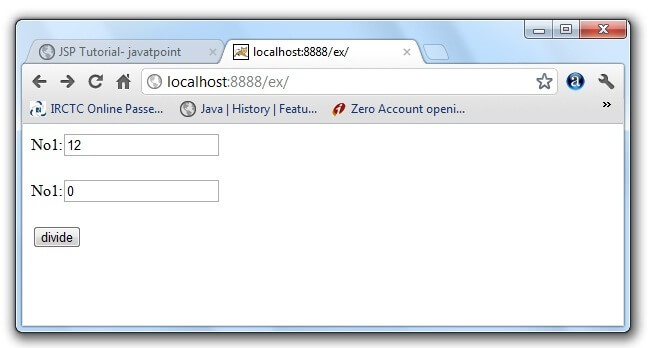
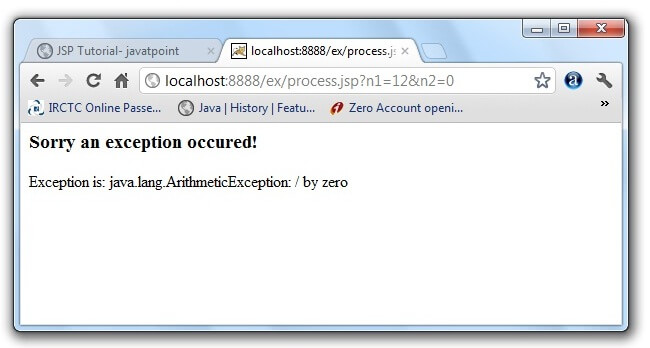
#### process.jsp

1. <%@ page errorPage="error.jsp" %>
2. <%
4. String num1=request.getParameter("n1");
5. String num2=request.getParameter("n2");
7. **int** a=Integer.parseInt(num1);
8. **int** b=Integer.parseInt(num2);
9. **int** c=a/b;
10. out.print("division of numbers is: "+c);
12. %>

#### error.jsp

1. <%@ page isErrorPage="true" %>
3. <h3>Sorry an exception occured!</h3>
5. Exception is: <%= exception %>

Output of this example:

2]explain JSP action elements[note:action tag and element are same]

a. There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks.

The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are given below.

|  |  |
| --- | --- |
| **JSP Action Tags** | **Description** |
| jsp:forward | forwards the request and response to another resource. |
| jsp:include | includes another resource. |
| jsp:useBean | creates or locates bean object. |
| jsp:setProperty | sets the value of property in bean object. |
| jsp:getProperty | prints the value of property of the bean. |
| jsp:plugin | embeds another components such as applet. |
| jsp:param | sets the parameter value. It is used in forward and include mostly. |
| jsp:fallback | can be used to print the message if plugin is working. It is used in jsp:plugin. |

**Q-3[D]FIVE MARK:**

1]what is java beans?explain its properties

a. A JavaBean is a Java class that should follow the following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

## Why use JavaBean?

According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.

\*A JavaBean property may be read, write, read-only, or write-only. JavaBean features are accessed through two methods in the JavaBean's implementation class:

1. getPropertyName ()

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

2. setPropertyName ()

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

2]JSP life cycle

a. . JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags.

>Translation of JSP page to Servlet :-

This is the first step of JSP life cycle. This translation phase deals with Syntactic correctness of JSP. Here test.jsp file is transllated to test.java.

>Compilation of JSP page :-

Here the generated java servlet file (test.java) is compiled to a class file (test.class).

>Classloading :-

Servlet class which has been loaded from JSP source is now loaded into container.

>Instantiation :-

Here instance of the class is generated. The container manages one or more instance by providing response to requests.

>Initialization :-

jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

>Request processing :-

\_jspService() method is used to serve the raised requests by JSP.It takes request and response object as parameters.This method cannot be overridden.

>JSP Cleanup :-

In order to remove the JSP from use by the container or to destroy method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

**Q-4[A]ONE MARK:**

1]full form HCQL

a. HCQL (Hibernate Criteria Query Language)

2]explain timer service

a. The timer service of the enterprise bean container enables you to schedule timed notifications for all types of enterprise beans except for stateful session beans.

3]list types of EJB

a. session beans, entity beans, and message-driven beans.

4]full form JPA

a. Java Persistence API (JPA)

**Q-4[B]TWO MARK:**

1]explain benefits of EJB

A. Following are the important benefits of EJB −

* Simplified development of large-scale enterprise level application.
* Application Server/EJB container provides most of the system level services like transaction handling, logging, load balancing, persistence mechanism, exception handling, and so on. Developer has to focus only on business logic of the application.
* EJB container manages life cycle of EJB instances, thus developer needs not to worry about when to create/delete EJB objects.

2]explain hibernate mapping

a. **Hibernate mappings** are one of the key features of **Hibernate**. They establish the relationship between two database tables as attributes in your model. That allows you to easily navigate the associations in your model and Criteria queries. ... Many to Many — It represents the many to many relationship between two tables

**Q-4[C]THREE MARK:**

1]advantages and disadvantaged of hibernate

a.  advantages:-  
Hibernate is an ORM tool

 Hibernate is an open source framework.

 Better than JBDC.

 Hibernate has an exception translator , which converts checked exceptions of JDBC in to unchecked exceptions of hibernate. So all exceptions in hibernate are unchecked exceptions and Because of this no need to handle exceptions explicitly.

 Hibernate supports inheritance and polymorphism.

 With hibernate we can manage the data stored across multiple tables, by applying relations(association)

 Hibernate has its own query language called Hibernate Query Language. With this HQL hibernate became database independent.

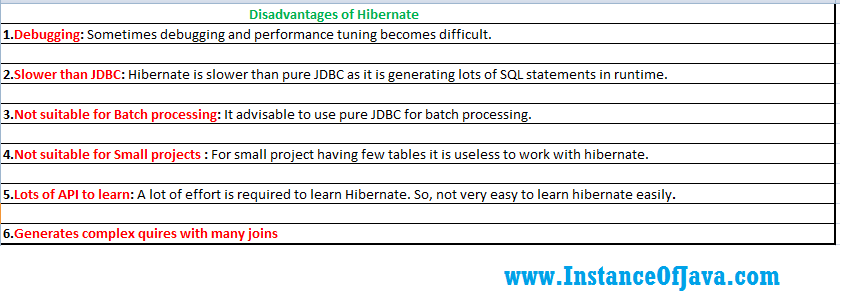
 Hibernate supports relationships like One-To-One, One-To-Many, Many-To-One ,Many-To-Many.

 Hibernate has Caching mechanism. using this number of database hits will be reduced. so performance of an application will be increases.

 Hibernate supports lot of databases.

 [Hibernate supported databases List](http://www.instanceofjava.com/2016/10/hibernate-supported-databases-list.html).

Disadvantages:-



2]hibernate inheritance

a. We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

1. Table Per Hierarchy
2. Table Per Concrete class
3. Table Per Subclass

Table Per Hierarchy

In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table .

[Table Per Hierarchy using xml file](https://www.javatpoint.com/hibernate-table-per-hierarchy-example-using-xml-file)  
[Table Per Hierarchy using Annotation](https://www.javatpoint.com/hibernate-table-per-hierarchy-using-annotation-tutorial-example)

Table Per Concrete class

In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

[Table Per Concrete class using xml file](https://www.javatpoint.com/table-per-concrete-class)  
[Table Per Concrete class using Annotation](https://www.javatpoint.com/hibernate-table-per-concrete-class-using-annotation-tutorial-example)

Table Per Subclass

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

[Table Per Subclass using xml file](https://www.javatpoint.com/table-per-subclass)  
[Table Per Subclass using Annotation](https://www.javatpoint.com/hibernate-table-per-subclass-using-annotation-tutorial-example)

**Q-4[D]FIVE MARK:**

1]MVC architecture

a. MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate

2]hibernate architecture

a. The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand api layer
* Database layer

## Elements of Hibernate Architecture

|  |
| --- |
| For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows: |

#### SessionFactory

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

#### Session

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

#### Transaction

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

#### ConnectionProvider

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

#### TransactionFactory

It is a factory of Transaction. It is optional.

**Q-5[A]ONE MARK:**

1]struts 2 framework is used for…

a. The Struts 2 framework is used to develop MVC (Model View Controller) based web applications

2]full form POJO- Plain Old Java Object

3]spring is a lightweight framework.true/false

a.true

4]full form IOC- inversion of Control (IoC)

**Q-5[B]TWO MARK:**

1]what is AOP?

A. Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity. But the key unit of modularity is aspect than class.

AOP breaks the program logic into distinct parts (called concerns). It is used to increase modularity by cross-cutting concerns.

A cross-cutting concern is a concern that can affect the whole application and should be centralized in one location in code as possible, such as transaction management, authentication, logging, security etc.

Why use AOP?

It provides the pluggable way to dynamically add the additional concern before, after or around the actual logic. Suppose there are 10 methods in a class as given below:

1. **class** A{
2. **public** **void** m1(){...}
3. **public** **void** m2(){...}
4. **public** **void** m3(){...}
5. **public** **void** m4(){...}
6. **public** **void** m5(){...}
7. **public** **void** n1(){...}
8. **public** **void** n2(){...}
9. **public** **void** p1(){...}
10. **public** **void** p2(){...}
11. **public** **void** p3(){...}
12. }

2]spring loc containers

A. The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. BeanFactory
2. ApplicationContext

**Q-5[C]THREE MARK:**

1]list out the execution flow of struts.

a.

2]explain struts 2 request life cycle

a. Request Life Cycle

* User sends a request to the server for requesting for some resource (i.e. pages).
* The Filter Dispatcher looks at the request and then determines the appropriate Action.
* Configured interceptor functionalities applies such as validation, file upload etc.
* Selected action is performed based on the requested operation.
* Again, configured interceptors are applied to do any post-processing if required.
* Finally, the result is prepared by the view and returns the result to the user.

**Q-5[D]FIVE MARK:**

1]spring architecture

a. The Spring Framework provides about 20 modules which can be used based on an application requirement.

Core Container

The Core Container consists of the Core, Beans, Context, and Expression Language modules the details of which are as follows −

* The **Core** module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
* The **Bean** module provides BeanFactory, which is a sophisticated implementation of the factory pattern.
* The **Context** module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.
* The **SpEL** module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access/Integration

The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows −

* The **JDBC** module provides a JDBC-abstraction layer that removes the need for tedious JDBC related coding.
* The **ORM** module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
* The **OXM** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
* The Java Messaging Service **JMS** module contains features for producing and consuming messages.
* The **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web

The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows −

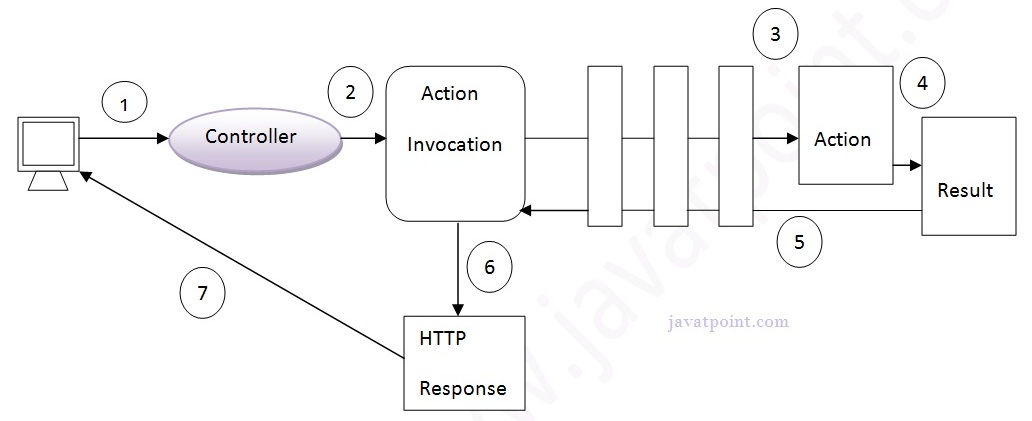
* The **Web** module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
* The **Web-MVC** module contains Spring's Model-View-Controller (MVC) implementation for web applications.
* The **Web-Socket** module provides support for WebSocket-based, two-way communication between the client and the server in web applications.
* The **Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules the details of which are as follows −

* The **AOP** module provides an aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
* The **Aspects** module provides integration with AspectJ, which is again a powerful and mature AOP framework.
* The **Instrumentation** module provides class instrumentation support and class loader implementations to be used in certain application servers.
* The **Messaging** module provides support for STOMP as the WebSocket sub-protocol to use in applications. It also supports an annotation programming model for routing and processing STOMP messages from WebSocket clients.
* The **Test** module supports the testing of Spring components with JUnit or TestNG frameworks.

2]explain struts flow diagram

a. 

1. User sends a request for the action
2. Controller invokes the ActionInvocation
3. ActionInvocation invokes each interceptors and action
4. A result is generated
5. The result is sent back to the ActionInvocation
6. A HttpServletResponse is generated
7. Response is sent to the user