Version Java EE 8 (J2EE 1.8) maintained under Oracle / Jakarta EE 8 (maintained by eclipse foundation)

1. What is J2EE ?(Java Enterprise Edition)

Consists of specifications only .

Which specs ? (Rules or contract)
Specifications of primary services required for any enterprise application.

What is enterprise application ?

An enterprise application (EA) is a large software system platform designed to operate in a corporate environment .

It includes online shopping and payment processing, interactive product catalogs, computerized billing systems, security, content management, IT service management, business intelligence, human resource management, manufacturing, process automation, enterprise resource planning ....

These specifications include ---

Servlet API,JSP(Java server page) API,Security,Connection pooling ,EJB (Enterprise Java Bean), JNDI(Naming service -- Java naming & directory i/f),JPA(java persistence API),JMS(java messaging service),Java Mail, Java Server Faces , Java Transaction API, Webservices support(SOAP/REST) etc...

Vendor of J2EE specs -- Oracle / Sun / Eclipse

Implementation -- left to vendors (J2EE server vendors)
J2EE compliant web server --- Apache -- Tomcat (web server)
Services implemented --- servlet API,JSP API,Security,Connection pooling,JNDI(naming service)

J2EE complaint application server --- web container + EJB (enterprise java bean) container
+ ALL J2EE services implementation

J2EE server Vendors & Products
Apache -- tomcat(web server) / Tomee (app server)
Oracle / Sun --- reference implementation --- Glassfish
Red Hat -- JBoss (wild fly)
Oracle / BEA -- weblogic
IBM -- Websphere

2. WHY J2EE
1. Can support different types of clnts --- thin client(web clnt)
thick clnt --- application clnt(eg : TCP client)
smart clnts -- mobile clnts

2. J2EE server independence --- Create & deploy server side appln --on ANY J2ee compliant server --- guaranteed to produce SAME results w/o touching or re-deploying on ANY other J2EE server

3. Ready made implementation of primary services(eg --- security, conn,pooling,email....)--- so that J2EE developer DOESn't have to worry about primary services ---rather can concentrate on actual business logic.

4. Layers involved in HTTP request-response flow (refer to day2-data\day2_help\diags\request-response-flow.png)

Web browser sends the request (URL)
 eg : http://www.abc.com:8080/day4.2
/day2.1 --- root / context path /web app name

Host --Web server--Web Container(server side JVM)--Web application---HTML/JSP/Servlet....

5. What is a dyn web application --- server side appln --deployed on web server --- meant for servicing typically web clnts(thin) -- using application layer protocol HTTP /HTTPS
(ref : diag request-resp flow)

Read --HTTP basics including request & response structure from day1-data\day1_help\j2ee_prerequisites\HTTP Basics

6. Objective ?: Creating & deploying dyn web appln on Tomcat -- For HTML content

7. IDE automatically creates J2EE compliant web application folder structure .
Its details -- Refer to diag (J2EE compliant web app folder structure)

8. What is Web container --- (WC) & its jobs
1. Server side JVM residing within web server.
Its run-time environment for dynamic web components(Servlet & JSP,Filter) .
Jobs ---
1. Creating Http Request & Http response objects
2. Controlling life-cycle of dyn web comps (manages life cycle of servlet,JSP,Filters)
3. Giving ready-made support for services --- Naming,security,Conn pooling .
4. Handling concurrent request from multiple clients .
5. Managing session tracking...

8. What is web.xml --- Deployment descriptor one per web appln
created by -- developer
who reads it -- WC
when --- @ deployment
what --- deployment instructions --- welcome page, servlet deployment tags, sess config, sec config......

----
9. Why servlets? --- To add dynamic nature to the web application

What is a servlet ?
-- Java class (with NO main method) -- represents dynamic web component - whose life cycle will be managed by WC(web container : server side JVM)
no main method
life cycle methods --- init,service,destroy

Job list
1. Request processing
2. B.L
3. Dynamic response generation
4. Data access logic(DAO class --managing DAO layer)
5. Page navigation

Servlet API details --refer to diag servlet-api.png

Objective 1: Test basic servlet life cycle -- init , service ,destroy
10. Creating & deploying Hello Servlet.

Deployment of the servlet
1. Via annotation
eg : @WebServlet(value="/validate")
public class LoginServlet extends H.S {....}
Map :
key -- /validate
value -- F.Q servlet cls name
URL : http://host:port/day1.1/validate?....

2. Using XML tags
How to deploy a servlet w/o annotations --- via XML tags
web.xml

```
<servlet>
 <servlet-name>abc</servlet-name>
<servlet-class>pages.SecondServlet</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>abc</servlet-name>
 <url-pattern>/test2</url-pattern>
</servlet-mapping>
```

WC : map
key : /test2

value : pages.SecondServlet

eg URL --http://host:port/day1_web/hello

At the time of web app deployment ---WC tries to populate map of url patterns , from XML tags (from web.xml). Later ---it will check for @WebServlet annotation

Objective 2: Test basic servlet life cycle -- init , service ,destroy (deployed via xml)

How to read request params sent from the clnt ?

javax.servlet.ServletRequest i/f methods
1. public String getParameter(String paramName)

2. public String[] getParameterValues(String paramName)

Objective 3 : Accept different type of i/ps from user , in HTML form.Write a servlet to display request parameters.

WHY JDBC ? (Java Database Connectivity)
JDBC ensures (PARTIAL) DB vendor independence + platform independence to Java Applications .

HOW it grants DB independence?

1. JAR supplied by DB vendor or Driver vendor --- consists of JDBC driver -- i.e. a converter for Java Data <-----> Native DB types & implementation classes , vendor specific.
2. JDBC API (java.sql) consists of largely --- Interfaces.

Java supplies specifications or WHAT (i/fs) & leaves implementation to DB vendors or 3rd party JDBC drvr vendors.

eg : java.sql.Connection --i/f sun supplied(Java SE)
Imple class provided by MySQL -- com.mysql.cj.jdbc.ConnectionImpl.class
Imple class provided by Oracle
oracle.jdbc.OracleConnection

DB2Connection etc....

Generalized steps for DB connectivity (ref : jdbc overview)

1. Place the JDBC driver in the Java classpath
Typically JDBC drivers are in form of JAR(Java archival format : compressed bundle of pkged Java classes) :
Oracle supplies Type IV Thin Client type of the Driver :
ojdbc14.jar/classes12.jar/ojdbc6.jar/ojdbc8.jar
MySQL supplies Type IV JDBC Driver : mysql-connector-java-8.0.20.jar

How to add JDBC drvr's JAR to the classpath(w/o IDE)

```
set classpath=g:\oracle\jdbc\lib\ojdbc8.jar;
```

With IDE --- simply Add external Jar.

2. Load & register JDBC driver

2.1. Load the JDBC driver in JVM's memory.

Method of java.lang.Class<T>

```
public static Class forName(String F.Q clsName) throws
ClassNotFoundException
```

eg : Class.forName("com.mysql.cj.jdbc.Driver");

3. Get the fixed DB connection thro' the JDBC driver.
API : java.sql.DriverManager (class)

```
public static Connection getConnection(String dbURL,String userName,String
password) throws SQLException
```

Params : dbURL : URL to reach DB thro the drvr.
jdbc:oracle:thin:@HostDetails --- for oracle Type IV thin clnt driver

HostDetails = DBServerHost:1521:SID

eg : jdbc:oracle:thin:@localhost:1521:orcl
For mysql
jdbc:mysql://localhost:3306/test?useSSL=false&allowPublicKeyRetrieval=true
test : DB name

4. Create the JDBC statement
Connection i/f method
public Statement createStatement() throws SQLException :
creates an empty JDBC stmt to hold the query & exec.

5. Fill in the query & execute the same.
Statement i/f method
If query is : select : u must use executeQuery method.If the query is DML other than select(i.e insert,delete,update) or DDL then use the method executeUpdate

5.1. For select query :(for result set returning query)
public ResultSet executeQuery(String sql) throws SQLException
Returns the result set consisting of selected rows & cols.

5.2. For others : (queries not returning RST)
public int executeUpdate(String sql) throws SQLException
Returns the updated row count : indicating how many rows were affected.

6.1 Process the ResultSet
API : ResultSet I/f method

public boolean next() throws SQLException
Advances the RST cursor to the next row & returns true : if valid data or false if no results.(end of results)
If valid data exists : then read row data
Methods from ResultSet i/f
Type getType(int colPosition) throws SQLException(colPos : as it appears in RST)
or
Type getType(String colName) throws SQLException

Type=JDBC data type.

Mapping bet. Oracle Data types & JDBC data type

varchar/varchar2	String
number(n)	integer
number(m,n)	double/float
date	java.sql.Date
TimeStamp	java.sql.TimeStamp

7 : Insertion of a row to the table (any query returning updateCnt)
Only changes are : query & replace executeQuery by executeUpdate.

8. If Java appln is exiting : close RST,close ST & then close Cn from the finally block or finalize method.(Typically closing Cn , closes all stmts & rsts.)

What is the type of the ResultSet so far created? : Forward type only & read only
To such a RST : previous() or abs. positioning meths(absolute(n) or relative(n) positioning meths will raise SE.

How to create a scrollable RST?
Replace step 4 by the following.

6. Create the JDBC statement to support scrollable RST
Connection i/f method
public Statement createStatement(int resultSetType,int concurrencyType) throws SQLException :
resultSet type : forward type(ResultSet.TYPE_FORWARD_ONLY) or scrollable(ResultSet.TYPE_SCROLL_INSENSITIVE OR ResultSet.TYPE_SCROLL_SENSITIVE) :

Difference between these 2
A result set that is TYPE_SCROLL_INSENSITIVE does not reflect changes made while it is still open and one that is TYPE_SCROLL_SENSITIVE does. Both types of result sets will make changes visible if they are closed and then reopened:

resultset Concurrency : read only result set(ResultSet.CONCUR_READ_ONLY) or updatable result set.(i.e can make changes to RST & same changes can also be applied to the DB table. can replace thus : insert,update,delete queries)

Additional API of scrollable ResultSet :
boolean absolute(int n) throws SE : tries to place the RST cursor on the nth row.

relative,afterLast,beforeFirst,first,last,previous,getRow

Why use PreparedStatement ?

1. PST represents pre-parsed & pre-compiled Stmts. At the time of creation of the PST, 3 steps out of 4(i.e parsing ,syntax chking,compiling) take place. So when User reqs for data(eg : via a button click) the only remaining step is : fill in user data & exec query.

2. U can't pass the IN params to Statement , but can pass IN params to
   PST.

How to create PST?
1. Use Connection i/f method :
public PreparedStatement prepareStatement(String sql) throws SE
eg : PreparedStatememt pst=cn.prepareStatement("select * from my_emp where id=?");
? : IN param. to be filled prior to query exec.

The RST associated with above PST is of : TYPE_FORWARD_ONLY & CONCUR_READ_ONLY

How to make it scrollable?
API : Connection i/f

public PreparedStatement prepareStatement(String sql,int resultType,int concurrencyType)
throws SE

3. How to set IN params of PST?(to be invoked : in event listener : after user gives i/p or in server side code after clnt sends request)
API : PreparedStatement i/f

void setType(int placeHolderPos,Type value) throws SE
Type : JDBC data type
PlaceHolder pos : 1.....counted from left
eg : to set emp id .
pst.setInt(1,....);

4. exec the query
 rst=pst.executeQuery();

5. process rst in the standard manner.

RMI clnt : sends emp id & RMI srvr contact DB : if emp exists ---sends emp info , ow. raises exc empnot found -- via pst.

CallableStatement : i/f from java.sql
Statement <--- PST <--- CallableStatement
why CST ?
1. Use CST to execute stored procedures & stored funs exisitng on DB
2. To pass IN,OUT,IN OUT type of params

Steps to invoke & exec. the stored proc/fun
1. Create CST
API : Connection i/f
public CallableStatement prepareCall(String invocationSyntax) throws SqlException
invocationSyntax for stored proc : "{call procName(?,?.....?)}"

invocationSyntax for stored fun : "{?=call funcName(?,?.....?)}"
? : represents IN,OUT or IN OUT param
{} : represent the esc seq. for the JDBC drvr. JDBC drvr will translate this invocation to a native DB invocation form.

2. Set IN params : methods inherited from PST

void setType(int placeHolderPos,Type value) throws SE

3. Register OUT / IN OUT params (i.e specify the JDBC data type of the OUT/IN OUT param to JVM)
Method of CST
void registerOutParameter(int paramPosition,int type) throws SE

paramPosition : placeHolder pos 1....
type : java.sql.Types : class constant

3.5 For IN OUT PARAM : invoke step 2 & 3 (ie. set IN val & register out param data type)

4. Execute the stored proc or a fun

public boolean execute() throws SE

Ret val is ignored.

5. Extract the results from OUT/IN OUT
CallableStatement methods
Type getType(int paramPos) throws SE
type : JDBC data type

Objective : Using scanner : accept sid,did,amt for funds transfer, exec the st.proc & disp the results.

DB Transactions
Functionally grouped SQL stmts : representing a B.L.
Tx => all the stmts from a Tx either fail or succeed.
i. e If any stmt fails : entire Tx has to be discarded.
The changes made by the Tx will be made permanent : IFF all the stmts succeed.
eg : Purchase a product
Involves 1. Checking availability of the product
2. Customer credit/debit limit & updating the same
3. Updating stock .

How to do it from JDBC API?

1. Start a Tx
Connection i/f method
void setAutoCommit(boolean flag)
ie. unset the auto-commit flag.
eg : cn.setAutoCommit(false);

2. Wrap entire Tx within a separate try-catch block.
3. If the entire try block succeds (i.e at the end of try) ---> commit the Tx

API : cn.commit();

4. But if u reach inside the catch clause(due to system exc or custom exc) : rollback the Tx

API : cn.rollback();

5. To continue : set auto-commit to true again.

6. To rollback a transaction partially , there exists additional method for setting save points.

Connection i/f method
public Savepoint setSavepoint() throws SQLException

7. How to restore the DB state to a savepoint ?

Connection i/f method
void rollback(Savepoint savepoint) throws SQLException
Undoes all changes made after the given Savepoint object was set.

Updatable ResultSet :
How to create a PST which supports scrollable & updatable RST?

1. API : Connection i/f

public PreparedStatement prepareStatement(String sql,int resultType,int concurrencyType)
throws SE
resultSet type : TYPE_SCROLL_INSENSITIVE/SENSITIVE
concurrencyType : CONCUR_UPDATABLE

2. Alternative to update query

2.1. Get the updatable RST.(eg : via pst.executeQuery())
2.2. Place the RST cursor on the row to be updated.(via absolute/relative meths of RST)
2.3. Update the col. vals-- on the RST

ResultSet API
public void updateType(int colPosition,Type newVal) throws SE
type--- JDBC data type
OR
public void updateType(String colName,Type newVal) throws SE
type--- JDBC data type

2.4. Once all changes to a particular row are done invoke :

API : public void updateRow() throws SE
to apply these changes to the underlying DB table.

3. Alternative to insert query

3.1. Get the updatable RST.(eg : via pst.executeQuery())

3.2. Place the RST cursor on the new row to be inserted.
API
ResultSet : void moveToInsertRow() throws SE
This places the RST cursor on the newly created row.

3.3. Update the col. vals-- on the RST copy
Invoke update methods (mandatory for NOT NULL constraint) : as in step 2.3

ResultSet API
public void updateType(int colPosition,Type val) throws SE
type--- JDBC data type

3.4. Once all col vals are inserted :
API : public void insertRow() throws SE
to apply these changes to the underlying DB table. (i.e new row gets inserted in DB)
3.5. To place cursor back to original row
API : public void moveToCurrentRow() throws SE

4. Alternative to delete query
4.1. Get the updatable RST.(eg : via pst.executeQuery())
4.2. Place the RST cursor on the row to be deleted (via absolute/relative)
4.3. Delete row :
ResultSet API
void deleteRow() throws SE (NOTE : deletes row from RST & DB too!!!!! use it with care!)

For date/time handling from JDBC
classes to be used from java.sql are :
Date,Time & TimeStamp

```
{d 'yyyy-mm-dd'}
{t 'hh:mm:ss'}
{ts 'yyyy-mm-dd hh:mm:ss'}
```

steps : for handling date
1. Create a table with col. type=date
2. Create a PST
3. Use java.sql.Date API
method :
public static Date valueOf(String dateFormat)

dateFormat : yyyy-mm-dd

4. Use PST's method
public void setDate(int pos,Date val) throws SE.

Meta data associated with JDBC
1. Database meta data : holds the info like : DB version,DB drvr version, Tx are supported or not, scrollable/updateble rsts, names of all tables from DB.....,max conns available

```
To get D.M.D
API : Connection i/f
DatabaseMetaData getMetaData() throws SE
DatabaseMetaData : i/f
Has methods : getVersion(),getTables().....


How to get all the table names for the current user?
Use DMD : method
ResultSet getTables(String catalog,String schemaPattern,
                    String tableNamePattern,
                    String[] types)
                    throws SQLException

Usage
DatabaseMetadata dmd=cn.getMetaData();
ResultSet rst=dmd.getTables(null,null,null,new String[] {"TABLE"});
//to retrieve table name
invoke : rst.getString(3) ; //3 => table name


2.  ResultSetMetaData : metadata about the RST
How to get it?
Method in ResultSet API
ResultSetMetaData getMetaData() throws SE
eg :
ResultSetMetaData rmd=rst.getMetaData();

2.1. Methods of RMD
int getColumnCount() throws SE
String getColumnLabel(int colPos) throws SE
int getColumnType(int colPos) throws SE



Dirty Read --Enables un-committed tx data, to read from current tx.
Un-repeatable reads -- Enables to read committed data from concurrent tx,
may lead to un repeatable results.
Phantom reads-- Enables to read committed data from concurrent tx, may
lead to additional rows appearing in same tx.


Handling BLOBs with JDBC API
How to store BLOB data?
1.  Create DB table having blob type of column.
eg create table my_images(id number(2),name varchar2(30),snap blob);
2.  Accept bin file from user to store on DB.
3.  Use PreparedStatement API method -- to store BLOB on DB
API
public void setBinaryStream(int placeholderPos, InputStream in, int
length) throws SqlException
4.  Use executeUpdate to insert row data.

How to restore BLOB data from DB ?
```

1. Use API of PreparedStatement to read BLOB.
public Blob getBlob(int colPos) throws SqlException
2. Use java.sql.Blob i/f method
public byte[] getBytes(long pos,int length)
NOTE : pos begin with 1 .
3. Once u have byte[] , u can store the same on File(bin) using FOS or send it over sockets using Socket.getOutputStream()

Reference for MySQL connectivity
1. install MySQL

2. Clnt i/f
```
create database testjdbc;
use testjdbc;

create table Employee( empId int primary key, name varchar(25), deptId
int, isPermanent boolean,sal double);
insert into Employee values(1,'aa',123,true,2000);
insert into Employee values(2,'ab',101,true,3000);
```

Driver class name : com.mysql.jdbc.Driver
To load/register driver ---- Class.forName(String F.Q className) throws ClassNotFoundExc
DB URL - jdbc:mysql://hostname:3306/databaseName
root -- user name
root -- password
example code for conn to MySQL ----
```
Class.forName("com.mysql.jdbc.Driver");
String dbURL="jdbc:mysql://localhost:3306/testjdbc";
//use DM.getConnection(url,username,pass)
```

Objective ---- RMI & JDBC integration
Func requirement --1. disp emp dtls --- if present , ow. raise cust exc.
3. Insert new emp record --- ret success msg or raise cust exc in case failure.

Server side steps
1. B.I --- method decl ---
String getEmpDtls(int empId) throws RE,EmpNotFoundExc
2. String insertEmp(emp specific dtls) throws RE,EmpInsertExc
3. Create impl class --- rem obj
constr --- cn,psts
B.M ---get ----
insert

HOW TO make JDBC applns/applets completely DB independent?

1. Create text based properties file.
key & value pair.(keys --- arbitrary values---changing as per DB setting)
2. Create empty java.util.Properties<K,V> --- sub-class of HashTable
Key & values must be --- String
Can load Properties directly from any stream.
Properties API
public void load(Reader r) throws IOExc

3. Can access the Property value using API
Properties API
String getProperty(String key)
ret type=value asso with key.

eg--

```
Properties props = new Properties();

FileInputStream in = new FileInputStream("database_mysql.properties");
props.load(in);
in.close();
String drivers = props.getProperty("jdbc.drivers");
Class.forName(drivers);
String url = props.getProperty("jdbc.url");
String username = props.getProperty("jdbc.username");
String password = props.getProperty("jdbc.password");
return DriverManager.getConnection(url, username, password);
```

Regarding jar cmd line utility
0. For runnable jars --- create manifest.txt --- 1liner having Main-Class: tester.Test, new line & save file
1. cd to folder where ur classes are(eg bin)

1. From bin --- jar cvfm test.jar manifest.txt *
2. To run jar
java -jar test.jar

Why HttpServlet classs is declared as abstract class BUT with 100 %
concrete functionality ?

It is abstract because the implementations of key servicing methods have to be provided by (e.g. overridden by) servlet developer. Since it's abstract , it's instance can't be created.

A subclass of HttpServlet must override at least one method, usually one of these:

doGet, if the servlet supports HTTP GET requests
doPost, for HTTP POST requests
doPut, for HTTP PUT requests
doDelete, for HTTP DELETE requests
init and destroy, to manage resources that are held for the life of the servlet

If you extend the class without overriding any methods, you will get a useless servlet; i.e. it will give an error response for all requests.(HTTP 405 : Method not implemented) . So , if the class was not abstract, then any direct instance of HttpServlet would be useless.

So the reason for making the HttpServlet class abstract is to prevent a programming error.

As a servlet developer , you can choose to override the functionality of your requirement (eg : doPost)
& ignore other methods.

Page Navigation Techniques
Page Navigation=Taking user from 1 page to another page.

2 Ways
1. Client Pull
Taking the client to the next page in the NEXT request (coming all the way from client)
1.1. User takes some action --eg : clicking on a button or link & then client browser generates new URL to take user to the next page.

1.2. Redirect Scenario
User doesn't take any action. Client browser automatically generates new URL to take user to the next page.(next page can be from same web appln , or diff web appln on same server or any web page on any srvr)

API of HttpServletResponse i/f
public void sendRedirect(String redirectURL) throws IOException
eg : For redirecting client from Servlet1 (/s1) to Servlet2 (/s2) , use
response.sendRedirect("s2");

If the response already has been committed(pw flushed or closed) , this method throws(WC) an IllegalStateException.(since WC can't redirect the client after response is already committed)

2. Server Pull.
Taking the client to the next page in the SAME request.
Also known as resource chaining or request dispatching technique.
Client sends the request to the servlet / JSP. Same request can be chained to the next page for further servicing of the request.

Steps
1. Create Request Dispatcher object for wrapping the next page(resource --can be static or dynamic)
API of ServletRequest
javax.servlet.RequestDispatcher getRequestDispatcher(String path)

2. Forward scenario
API of RequestDispatcher
public void forward(ServletRequest rq,ServletResponse rs)

This method allows one servlet to do initial processing of a request and another resource to generate the response. (i.e division of responsibility)

Uncommitted output in the response buffer is automatically cleared before the forward.

If the response already has been committed(pw flushed or closed) , this method throws an IllegalStateException.

Limitation --only last page in the chain can generate dynamic response.

3. Include scenario
API of RequestDispatcher
```
public void include(ServletRequest rq,ServletResponse rs)
```

Includes the content of a resource @run time (servlet, JSP page, HTML file) in the response. -- server-side includes.

Limitation -- The included servlet/JSP cannot change the response status code or set headers; any attempt to make a change is ignored.

What is a Session?

Session is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session is passed between server and client in every request and response.

HTTP protocol and Web Servers are stateless, what it means is that for web server every request is a new request to process and they cant identify if its coming from client that has been sending request previously.

But sometimes in web applications, we should know who the client is and process the request accordingly. For example, a shopping cart application should know who is sending the request to add an item and in which cart the item has to be added or who is sending checkout request so that it can charge the amount to correct client.

What is the need of session tracking?

1. To identify the clnt among multiple clnts
2. To remember the conversational state of the clnt(eg : list of the purchased books/ shopping cart/bank acct details/stocks) throughout current session

session = Represents duration or time interval
default session timeout for Tomcat =30minutes

Session Consists of all requests/resps coming from/ sent to SAME clnt from login to logout or till session expiration tmout.

There are several techniques for session tracking.
J2EE specific techniques :
1. Plain Cookie based scenario
2. HttpSession interface
3. HttpSession + URL rewriting

Techniques

1. Plain Cookie based scenario

What is a cookie?
Cookie is small amount of text data.
Created by -- server (servlet or JSP prog or WC) & downloaded (sent) to clnt browser---within response header
 Cookie represents data shared across multiple dyn pages from the SAME web appln.(meant for the same client)

Steps :

1. Create cookie/s instance/s
javax.servlet.http.Cookie(String cName,String cVal)

2. Add the cookie/s to the resp hdr.

HttpServletResponse API :
void addCookie(Cookie c)

3. To retrieve the cookies :
HttpServletRequest :
Cookie[] getCookies()

4. Cookie class methods :
String getName()
String getValue()
void setMaxAge(int ageInSeconds)
def age =-1 ---> browser stores cookie in cache
=0 ---> clnt browser should delete cookie
>0 --- persistent cookie --to be stored on clnt's hard disk.

int getMaxAge()

Disadvantages of pure cookie based scenario
0. Web developer (servlet prog) has to manage cookies.
1. Cookies can handle only text data : storing Java obj or bin data difficult.
2. As no of cookies inc., it will result into increased net traffic.
3. In cookie based approach : entire state of the clnt is saved on the clnt side. If the clnt browser rejects the cookies: state will be lost : session tracking fails.

How to redirect client automatically to next page ? (in the NEXT request)
API of HttpServletResponse
public void sendRedirect(String redirectLoc)
eg : resp.sendRedirect("s2");

IMPORTANT :
WC -- throws
java.lang.IllegalStateException: Cannot call sendRedirect() after the response has been committed(eg : pw.flush(),pw.close()...)

Technique # 2 : Session tracking based on HttpSession API
In this technique :
Entire state of the client is not saved on client side , instead saved on the server side data structure (Http Sesion object) BUT the key to this Http Session object is STILL sent to client in form of a cookie.(cookie management is done by WC)

Servlet programmer can store/restore java objects directly under the session scope(API : setAttribute/getAttribute)

Above mentioned , disadvantages ---0, 1 & 2 are reomved.
BUT entire session tracking again fails , if cookies are disabled.

Steps for javax.servlet.http.HttpSession i/f based session tracking.

1. Get Http Session object from WC

API of HttpServletRequest ---
HttpSession getSession()
Meaning --- Servlet requests WC to either create n return a NEW HttpSession object(for new clnt) or ret the existing one from WC's heap for existing client.

HttpSession --- i/f from javax.servlet.http
In case of new client :
 HttpSession<String,Object> --empty map
String,Object ---- (entry)= attribute

OR
HttpSession getSession(boolean create)

2. : How to save data in HttpSession?(scope=entire session)
API of HttpSession i/f
public void setAttribute(String attrName,Object attrVal)
eg : hs.setAttribute("clnt_info",validatedCustomer);//no javac err
 attribute : server side object ---server side entry (key n value pair) --map

equivalent to map.put(k,v)
eg : hs.setAttribute("cart",l1);

3. For retrieving session data(getting attributes)
public Object getAttribute(String attrName) //key
eg : Customer cust=(Customer) hs.getAttribute("clnt_info");

4. To get session ID (value of the cookie whose name is jsessionid -- unique per client by WC)
String getId()

4.5 How to remove attribute from the session scope?
public void removeAttribute(String attrName)
eg : hs.removeAttribute("clnt_info");

5. How to invalidate session?
HttpSession API
public void invalidate()
(WC marks HS object on the server side for GC ---BUT cookie is NOT deleted from clnt browser)

6. HttpSession API
public boolean isNew()
Rets true for new client & false for existing client.

7. How to find all attr names from the session ?
public Enumeration<String> getAttributeNames()
--rets java.util.Enumeration of attr names.

8. Default session timeout value for Tomcat = 30 mins
How to change session tmout ?
HttpSession i/f method
public void setMaxInactiveInterval(int secs)
eg : hs.setMaxInactiveInterval(300); --for 5 mins .

OR via xml tags in web.xml
```
<session-config>
  <session-timeout>5</session-timeout> : unit : min
</session-config>
```

NOTE :
What is an attribute ?
attribute = server side object(entry/mapping=key value pair)
who creates server side attrs ? -- web developer (servlet or JSP prog)
Each attribute has --- attr name(String) & attr value (java.lang.Object)
Attributes can exist in one of 3 scopes --- req. scope,session scope or application scope

1. Meaning of req scoped attr = attribute is visible for current req.
2. Meaning of session scoped attr = attribute is visible for current session.(shared across multiple reqs coming from SAME clnt)
3. Meaning of application scoped attr = attribute is visible for current web appln.(shared across multiple reqs from ANY clnt BUT for the SAME web application)

Executor Framework (a part of Java SE)

Introduced in Java 5.

What's earlier support ?

Extends Thread
Implements Runnable

Why Executor Framework?

If you have thousands of task to be executed and if you create each thread for thousands of tasks, you will get performance overheads as creation and maintenance of each thread is an overhead.

Executor framework solves this problem.

In executor framework, you can create specified number of threads and reuse them to execute more tasks once it completes its current task.

It simplifies the design of creating multithreaded application and manages thread life cycles.

The programmer does not have to create or manage threads themselves, that’s the biggest advantage of executor framework.

Important classes / interfaces for executor framework.

1. java.util.concurrent.Executor
This interface is used to submit new task.
It has a method called “execute”.

```
public interface Executor {
 void execute(Runnable task);
}
```

2. ExecutorService
It is sub-interface of Executor.
Provides methods for
Submitting / executing Callable/Runnable tasks
Shutting down service
Executing multiple tasks etc.

3. ScheduledExecutorService
It is sub-interface of executor service which provides methods for scheduling tasks at fixed intervals or with initial delay.

4. Executors
This class provides factory methods for creating thread pool based executors.

Important factory methods(=public static method rets instance of ExecutorService) of Executors are:

4.1. newFixedThreadPool: This method returns thread pool executor whose maximum size is fixed.If all n threads are busy performing the task and additional tasks are submitted, then they will have to wait in the queue until thread is available.

4.2.
newCachedThreadPool: this method returns an unbounded thread pool. It doesn’t have maximum size but if it has less number of tasks, then it will tear down unused thread. If a thread has been unused for keepAliveTime , then it will tear it down.

4.3. newSingleThreadedExecutor: this method returns an executor which is guaranteed to use the single thread.

4.4. newScheduledThreadPool: this method returns a fixed size thread pool that can schedule commands to run after a given delay, or to execute periodically.

Steps for Runnable

1. Create a thread-pool executor , using suitable factory method of Executors.

eg : For fixed no of threads

```
ExecutorService executor = Executors.newFixedThreadPool(10);
```

2. Create Runnable task

3. Use inherited method

```
public void execute(Runnable command)
```

Executes this Runnable task , in a separate thread.

4. Shutdown the service

```
public void shutdown()
```

Initiates an orderly shutdown in which previously submitted tasks are executed, but no new tasks will be accepted.

5.
```
boolean awaitTermination(long timeout,TimeUnit unit)
                 throws InterruptedException
```

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs.

6.

```
List<Runnable> shutdownNow()
```

Attempts to stop all actively executing tasks, halts the processing of waiting tasks, and returns a list of the tasks that were awaiting execution.

BUT disadvantages with Runnable interface

1. Can't return result from the running task
2. Doesn't include throws Exception .

Better API

java.util.concurrent.Callable<V>

V : result type of call method

Represents a task that returns a result and may throw an exception.

Functional i/f
SAM :
public V call() throws Exception
Computes a result, or throws an exception if unable to do so.

Steps in using Callable i/f

1. Create a thread-pool executor , using suitable factory method of Executors.

eg : For fixed no of threads
ExecutorService executor = Executors.newFixedThreadPool(10);

2. Create Callable task , which returns a result.

3. To submit a task to executor service , use method of ExecutorService i/f :

public Future<T> submit(Callable<T> task)
Submits a value-returning task for execution and returns a Future representing the pending results of the task. It's a non blocking method (i.e rets immediately)

The Future's get method will return the task's result upon successful completion.

If you would like to immediately block waiting for a task, invoke get() on Future.
eg : result = exec.submit(aCallable).get();

OR
main thread can perform some other jobs in the mean time & then invoke get on Future , to actually get the results. (get : blocking call ,waits till the computation is completed n then rets result)

4. Other methods of ExecutorService i/f

public List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks) throws InterruptedException

It's a blocking call.(waits till all tasks are complete)
Executes the given tasks, returning a list of Futures holding their status and results when all complete. Future.isDone() is true for each element of the returned list.

5. Shutdown the service
public void shutdown()
Initiates an orderly shutdown in which previously submitted tasks are executed, but no new tasks will be accepted.

6. boolean awaitTermination(long timeout,TimeUnit unit)
                  throws InterruptedException
Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs.

7.

```
List<Runnable> shutdownNow()
Attempts to stop all actively executing tasks, halts the processing of
waiting tasks, and returns a list of the tasks that were awaiting
execution.
```

Regarding SERVLET CONFIG

A servlet specific configuration object created by a servlet container to pass information to a servlet during initialization.

1. Represents Servlet specific configuration.
Defined in javax.servlet.ServletConfig -- interface.

2. Who creates its instance ?
Web container(WC)
3. When ?
After WC creates servlet instance(via def constr), ServletConfig instance is created & then it invokes init() method of the servlet.
4. Usage
To store servlet specific init parameters.
(i.e the init-param is accessible to one servlet only or you can say that the init-param data is private for a particular servlet.)

5. Where to add servlet specific init parameters?
Can be added either in web.xml or @WebServlet annotation.

XML Tags
```
<servlet>
    <servlet-name>init</servlet-name>
    <servlet-class>ex.TestInitParam</servlet-class>
    <init-param>
      <param-name>name</param-name>
      <param-value>value</param-value>
    </init-param>
</servlet>
<servlet-mapping>
<servlet-name>init</servlet-name>
<url-pattern>/test_init</url-pattern>
</servlet-mapping>
```

6. How to access servlet specific init params from a servlet ?
6.1. Override init() method
6.2. Get ServletConfig
Method of Servlet i/f
`public ServletConfig getServletConfig()`
6.3. Get the init params from ServletConfig
Method of ServletConfig i/f
`String getInitparameter(String paramName)` : rets the param value.

Regarding javax.servlet.ServletContext (i/f)

1. Defined in javax.servlet package.
2. Who creates its instance -- WC
3. When -- @ Web application (=context) deployment time

NOTE : The ServletContext object is contained within the ServletConfig object, which the WC provides the servlet when the servlet is initialized.

4. How many instances ? --one per web application

5. Usages

5.1. Server side logging
API public void log(String mesg)

5.2. To create context(=application) scoped attributes
API public void setAttribute(String nm,Object val)
NOTE : Access them always in thread safe manner (using synchronized blocks)

5.3. To access global(scope=entire web application) parameters
How to add context scoped parameters ?

In web.xml

```
<context-param>
  <param-name>name</param-name>
      <param-value>value</param-value>
</context-param>
```

How to access these params in a Servlet ?
(can be accessed from init method onwards)

1. Get ServletContext
API of GenericServlet
ServletContext getServletContext() --method inherited from GenericServlet

2. ServletContext API
String getInitparameter(String paramName) : rets the param value.
eg : ctx param name : user_name value : abc
In the Servlet : getServletContext().getInitparameter("user_name") ---abc

5.4 Creating request dispatcher
H.W

What is a Servlet Listener(or web application listener)?

During the lifetime of a typical web application, a number of events take place.
eg : requests are created or destroyed.
sessions are created & destroyed
Contexts(web apps) are created & destroyed.
request or session or context attributes are added, removed, or modified etc.

The Servlet API provides a number of listener interfaces that one can implement in order to react to these events.

eg : Event Listener i/f
1. ServletRequestListener
2. HttpSessionListener
3. ServletContextListener
....
Event Handling Steps
1. Create a class , implementing from Listener i/f.
2. Register it with WC
2.1. @WebListener annotation(class level)
OR
2.2. XML tags in web.xml

```
<listener>
 <listener-class>F.Q cls name of listener</listener-class>
</listener>
```

Expression Language implicit variables(case sensitive)

1. pageContext : PageContext object (javax.servlet.jsp.PageContext) asso. with current page.
2. pageScope - a Map that contains page-scoped attribute names and their
   values.
3. requestScope - a Map that contains request-scoped attribute names and their
   values.
4. sessionScope - a Map that contains session-scoped attribute names and their
   values.
5. applicationScope - a Map that contains application-scoped attribute names
   and their values.
6. param - a Map that contains rq. parameter names to a single String parameter
   value (obtained by calling ServletRequest.getParameter(String name)).
7. paramValues - a Map that contains rq. param name to a String[] of all values
   for that parameter (similar to calling
ServletRequest.getParameterValues(name)
8. initParam - a Map that contains context initialization parameter names and their
String value (obtained by calling ServletContext.getInitParameter(String name)).
eg : ${initParam.db_drvr}

9. cookie : Map.Entry of cookies. (entrySet of cookies)
eg : ${cookie.cookiename.value}

key ---cookie name
value ---javax.servlet.http.Cookie

${cookie.JSESSIONID.value}
---cookie.get("JSESSIOIND").getValue()

10. To retrieve err details from Error handling page.
 ERR causing URI : ${pageContext.errorData.requestURI }
 ERR code : ${pageContext.errorData.statusCode}
 ERR Mesg : ${pageContext.exception.message }
 Throwable : ${pageContext.errorData.throwable}
 Throwable Root cause: ${pageContext.errorData.throwable.cause}

eg :

```
<c:set var="abc" scope="session" value="Hello User...."/>
${sessionScope.abc}
```

Session Tracking technique :
 HttpSession + URL rewriting

Why ????
To develop a web app , independent of cookies , for session tracking.

For tracking the clnt (clnt's session) : the only information, WC needs from the clnt browser is JSessionID value. If clnt browser is not sending it using cookie : Servlet/JSP prog can embed the JSessionID info in each outgoing URL .(response: location / href /form action)

What is URL Rewriting : Encoding the URL to contain the JSessionID info.

W.C always 1st chks if JsessionID is coming from cookie, if not ---> then it will chk in URL : if it finds JsessionID from the encoded URL : extracts its value & proceeds in the same manner as earlier.

How to ?

API :
For URLs generated by clicking link/buttons(clnt pull I) use HttpServletResponse method
```
public String encodeURL(String origURL)
```
Rets : origURL;JSESSIONID=12345

For URLs generated by sendRedirect : clnt pull II : use HttpServletResponse method
```
public String encodeRedirectURL(String redirectURL)
```
Rets : redirectURL;JSESSIONID=12345

What is JSP? (Java server pages)
Dynamic Web page (having typically HTML 5 markup) , can embed Java code directly.

Dynamic web component , whose life-cycle is managed by WC(JSP container/Servlet container/Servlet engine)

WHY JSP?

1. JSP allows developer to separate presentation logic(dyn resp generation) from Business logic or data manipulation logic.
Typically JSPs -- used for P.L(presentation logic)
Java Beans or Custom Tags(actions) --- will contain Business logic.

2. Ease of development --- JSP pages are auto. translated by W.C in to servlet & compiled & deployed.

3. Can use web design tools -- for faster development (RAD --rapid application development) tools.

JSP API
jsp-api.jar --- <tomcat>/lib : specs

Contains JSP API implementation classses. : jasper.jar

0. javax.servlet.Servlet -- super i/f
1. javax.servlet.jsp.JspPage -- extends Servlet i/f
1.1. public void jspInit()
1.2. public void jspDestroy()

Can be overridden by JSP page author

2. Further extended by javax.servlet.jsp.HttpJspPage
2.1. public void _jspService(HttpServletRequest rq,HttpServletResponse rs)
     throws ServletExc,IOExc.

Never override _jspService ---JSP container auto translates JSP tags (body) into _jspService.

JSP life-cycle

1. Clnt sends the 1st request to the JSP (test.jsp)
2. Web-container invokes the life cycle for JSP
3. Translation Phase : handled by the JSP container.
I/p : test.jsp O/p : test_jsp.java (name : specific to the Tomcat container)
Meaning : .jsp is translated into corresponding servlet page(.java)
Translation time errs : syntactical errs in using JSP syntax.
In case of errs : life-cycle is aborted.
4. Compilation Phase : handled by the JSP container.

I/p : Translated servlet page(.java)   O/p : Page Translation class(.class)
Meaning : servlet page auto. compiled into .class file
Compilation time errs: syntacticle errs in generated Java syntax.

5. Request processing phase / Run time phase. : typically handled by the Servlet Container.
6. S.C : will try to locate,load,instantiate the generated servlet class.
7. The 1st it calls : public void jspInit() : one time inits can be performed.(jspInit availble from javax.servlet.jsp.JspPage)
8. Then it will call follwing method using thrd created per clnt request :

public void _jspService(HttpServlet Rq,HttpServletResponse) throws ServletException,IOException(API avlble from javax.servlet.jsp.HttpJspPage)
When _jspService rets , thread's run method is over & thrd rets to the pool, where it can be used for servicing some other or same clnt's req.

9. . At the end ...(server shutting down or re-deployment of the context) : the S.C calls

public void jspDestroy()
After this : translated servlet page class inst. will be GCEd....

10. For 2nd req onwards ...... : SC will invoke step 8 onwards.

JSP 2.0/2.1/2.2/2.3 syntax

1. JSP comments

1.1. server side comment
syntax : <%-- comment text --%>
significance : JSP translator & compiler ignores the commented text.

1.2. clnt side comment
syntax : <!-- comment text -->
significance : JSP translator & compiler does not ignore the commented text BUT clnt browser will ignore it.

2. JSP's implicit objects (available only to _jspService) -- avlable to scriptlets,exprs

2.1. out - javax.servlet.jsp.JspWriter : represents the buffered writer stream connected to the clnt via HttpServletResponse(similar to your PrintWriter in servlets)

Has the same API as PW(except printf)
usage eg : out.print("some text sent to clnt");

2.2. request : HttpServletRequest (same API)

2.3. response : HttpServletResponse

2.4. config : ServletConfig (used for passing init params)

2.5. session : HttpSession (By def. all JSPs participate in session tracking i.e session obj is created)

2.6. exception : java.lang.Throwable (available only to err handling pages)

2.7. pageContext : current page environment : javax.servlet.jsp.PageContext(this class stores references to page specific objects viz -- exception,out,config,session)

2.8. application : ServletContext(used for Request dispatching, server side logging, for creating context listeners,to avail context params, to add/get context scoped attrs)

2.9. page --- current translated page class instance created for 'this' JSP

3. Scripting elements : To include the java content within JSP : to make it dynamic.

3.1. Scriptlets : can add the java code directly . AVOID scriptlets . (Use only till you learn Javabeans & custom tags or JSTL,). we will use use the scriptlets to add : Req. processing logic, B.L & P.L)

syntax : <% java code...... %> : within <body> tag.
location inside the translated page : within _jspService
usage : till Java beans / JSTL or cust. tags are introduced : scriptlets used for control flow/B.L/req. proc. logic

3.2. JSP expressions :
syntax : <%= expr to evaluate %>
--Evaluates an expression --converts it to string --send it to clnt browser.
eg : <%= new Date() %>

expr to evaluate : java method invocation which rets a value OR const expr or attributes(getAttribute) or variables(instance vars or method local)
location inside the translated page : within _jspService
significance : the expr gets evaluated---> to string -> automatically sent to clnt browser.

```
eg <%= new Date() %>
eg <%= request.getAttribute("user_dtls") %>
<%= 12*34*456 %>
<%= session.getAttribute("user_dtls") %>
<%= session.setAttribute("nm",1234) %> -- compiler error
<%= session.getId() %>
```

Better alternative to JSP Expressions : EL syntax (Expression Language : avlble from JSP 1.2 onwards)
syntax : ${expr to evaluate} (to be added directly in ,<body> tag)

EL syntax will evaluate the expr ---to String --sends it clnt browser.

JSP implicit object --- request,response,session...---accessible from scriptlets & JSP exprs. ---

EL implicit objects --- can be accessible only via EL syntax
param =Name of the map , created by WC : containing request parameters
pageScope=Name of the map , created by WC : containing page scoped attrs
requestScope=map of request scoped attrs
sessionScope=map of session scoped attrs
applicationScope=map of application(=context) scoped attrs
pageContext --- instance of PageContext's sub class
cookie -- map of cookies(cookie objects)
initParam -- map of context params.

---avlable ONLY to EL syntax ${...}
---to be added directly within <body> ...</body>

eg : ${param.user_nm} ---param.get("user_nm") --value --to string ---> clnt
request.getParameter("user_nm") --value --to string ---> clnt

${requestScope.abc} ---request.getAttribute("abc") ---to string --sent to clnt browser.

eg : suppose ctx scoped attr --- loan_scheme
${applicationScope.loan_scheme} ---
getServletContext().getAttribute("loan_scheme") ---to string --sent to clnt

${abc} ---
pageContext.getAttribute("abc") ---not null -- to string -clnt
 null
--request.getAttribute("abc") -- not null -- to string -clnt
null
session.getAttribute("abc") ---
null
getServletContext().getAttirbute("abc") --not null -- to string -clnt
null ---BLANK to clnt browser.

eg : ${sessionScope.nm} OR ${nm}

${pageContext.session.id}
--pageContext.getSession().getId() --- val of JessionId cookie w/o java code.

${pageContext.request.contextPath} ---/day5_web

${pageContext.session.maxInactiveInterval}

${param}
{user_nm=asdf, user_pass=123456}

eg : ${param.f1} ---> request.getParameter("f1").toString()---> sent to browser

param ----map of req parameters.

param : req. param map

${requestScope.abc} -----
out.print(request.getAttribute("abc").toString())

${abc} -----pageCotext.getAttribute("abc")----null ---request ---session---application ---null ---EL prints blank.

3.3. JSP declarations (private members of the translated servlet class)
syntax : <%! JSP declaration block %> (outside <body>)
Usage : 1. for creating page scoped java variables & methods (instance vars & methods/static members)
4. Also can be used for overriding life cycle methods
    (jspInit,jspDestroy)

location inside the translated page : outside of _jspService (directly within JSP's translated class)

JSP Directives --- commands/messages for JSP Engine(=JSP container=WC) -- to be used @Translation time.

Syntax ---
<%@ Directive name attrList %>
1. page directive
--- all commands applicable to current page only.
Syntax
<%@ page import="comma separated list of pkgs" contentType="text/html" %>
eg -- <%@ page import="java.util.*,java.text.SimpleDateFormat" contentType="text/html" %>
Imp page directive attributes
1. import --- comma separated list of pkgs
2. session --- boolean attribute. default=true.
To disable session tracking, spectify session="false"

3. errorPage="URI of err handling page" ---
tells WC to forward user to err handler page.
4. isErrorPage="true|false" def = false

If you enable this to true--- one can access 'exception' implicit object from this page.

This exception obj is stored under current page ---i.e under pageContext (type=javax.servlet.jsp.PageContext -- class which represents curnt JSP)
EL expresssion to display error mesg
${pageContext.exception.message}
-- evals to pageContext.getException().getMessage()

Additional EL syntax

EL syntax to be used in error handling pages

```
ERR causing URI :  ${pageContext.errorData.requestURI }<br/>
 ERR code :  ${pageContext.errorData.statusCode}<br/>
 ERR Mesg :  ${pageContext.exception.message} <br/>
 Throwable : ${pageContext.errorData.throwable}<br/>
 Throwable Root cause: ${pageContext.errorData.throwable.cause}
```

5. isThreadSafe="true|false" default=true. "true" is recommended
true=>informing WC--- JSP is already written in thrd -safe manner ---- DON'T apply thrd safety.

false=>informing WC --- apply thrd safety.

(NOT recommended) ---WC typically marks entire service(servlet scenario) or _jspService in JSP scenarion --- synchronized. --- this removes concurrent handling of multiple client request --so not recommended.
What is recommended? --- isThreadSafe=true(def.) --- identify critical section(i.e code prone to race condition among threads)--guard it in synchronized block.
eg ---Context scoped attrs are inherently thrd -un safe. So access them always from within synched block.

Equivalent step in Servlet
Servlet class can imple. tag i/f --
javax.servlet.SingleThreadModel(DEPRECATED) -- WC ensures only 1thread (representing clnt request) can invoke service method. --NOT recommended.

6. include directive
<%@ include file="URI of the page to be included" %>
Via include directive ---- contents are included @ Translation time.--- indicates page scope(continuation of the same page).
Typically used -- for including static content (can be used to include dyn conts)
eg ---one.jsp
....<%@ include file="two.jsp" %>
two.jsp.....

JSP actions ---- commands/messages meant for WC
to be interpreted @ translation time & applied @ req. processing time.(run time)

Syntax ---standard actions --specifications are present in jsp-api.jar.(implementations in jasper jar)

```
<jsp:actionName attribute list>Body of the tag/action
</jsp:actionName>
```

OR

```
<jsp:actionName attr list />
```

JSP Using Java beans(JB)
Why Java Beans
 ---1. allows prog to seperate B.L in Javabeans(Req processing logic, Page navigation & resp generation will be still part of JSP)

Javabeans can store conversational state of clnt(Javabeans 's properties will reflect clnt state) + supplies Business logic methods.

7. simple sharing of JBS across multiple web pages---gives rise to re-usability.

8. Automatic translation between req. params & JB props(string--->primitive data types automatically done by WC)

What is JB?
1. pkged public Java class
It's actually an attribute automatically created by WC.(trigger : jsp:useBean)
& WC will automatically store it under the specified scope
2. Must have def constr.(MUST in JSP using JB scenario)
3. Properties of JBs --- private, non-static , non-transient Data members --- equivalent to request params sent by clnt.(Prop names MUST match with req params for easy usage)
In proper words --- Java bean properties reflect the conversational state of the clnt.
4. per property -- if RW
naming conventions of JB
supply getter & setter.
Rules for setter (Java Bean Naming convention) : strict
public void setPropertyName(Type val)
Type -- prop type.
eg -- private double regAmount;
public void setRegAmount(double val)
{...}
Rules for getter
public Type getPropertyName()
Type -- prop type.

```
eg -- public double getRegAmount(){...}

5.  Business Logic --- methods
public methods --- no other restrictions
----------------------------
Using Java Beans from JSP Via standard actions

1.  <jsp:useBean id="BeanRef name" class="F.Q. Bean class name"
    scope="page|request|session|application/>

default = page scope.


pre-requisite --- JB class exists under <WEB-INF>/classes.
 JB = server side obj (attribute), attr name --- bean id,attr val -- bean
inst.,can be added to any scope using scope atribute.

eg :
eg --- beans.Userbean
props --- email,pass
setters/getters
B.L mehod -- for validation

Usage ---
<jsp:useBean id="user" class="beans.UserBean" scope="session"/>


W.C invokes JB life-cycle
1.  WC chks if specified Bean inst alrdy exists in specified scope
java api --- request.getAttribute("user")
---null=>JB doesn't exist
---loc/load/inst JB class
UserBean u1=new UserBean();
--add JB inst to the specified scope
java api -- request.setAttribute("user",u1);
--- not-null  -- WC continues....

2.  JSP using JB action
2.1. <jsp:setProperty name="Bean ref Name" property="propName"
     value="propVal---static/dyn" />
Usage--
<jsp:setProperty name="user" property="email"
value="a@b"/>
WC invokes --- session.getAttribute("user").setEmail("a@b");

<jsp:setProperty name="user" property="email"
value="<%= request.getParameter("f1") %>"/>

OR via EL
<jsp:setProperty name="user" property="email"
value="${param.f1}"/>

WC invokes ---
session.getAttribute("user").setEmail(request.getParameter("f1"));
```

```
2.2.
<jsp:setProperty name="Bean ref Name" property="propName" param="rq. param
name"/>


Usage eg --
<jsp:setProperty name="user" property="email" param="f1"/>


WC invokes ---
((Userbean)request.getAttribute("user")).setEmail(request.getParameter("f1
"));



2.3.
<jsp:setProperty name="Bean ref Name" property="*"/>

usage

<jsp:setProperty name="user" property="*"/>


eg -- If rq. param names are email & password(i.e matching with JB prop
names) then ---matching setters(2) will get called

3.  <jsp:getProperty name="Bean ref name" property="propName"/>
Usage --
<jsp:getProperty name="user" property="email"/>
WC ---
session.getAttribute("user").getEmail()--- toString --- sent to clnt
browser.

Better equivalent  -- EL syntax
${sessionScope.user.email} ---
session.getAttribute("user").getEmail()--- toString --- sent to clnt
browser.

${requestScope.user.validUser.email}
request.getAttribute("user").getValidUser().getEmail()

${pageContext.exception.message}


4.  JSP standard actions related to Request Dispatcher
RD's forward scenario
<jsp:forward page="dispatcher URI" />
eg : In one.jsp
<jsp:forward page="two.jsp"/>
WC invokes ---RequestDispatcher
rd=reuqest.getRequestDispatcher("two.jsp");
rd.forward(request,response);
```

```
RD's include scenario
<jsp:include page="dispatcher URI" />

eg : In one.jsp
<jsp:include page="two.jsp"/>
WC invokes ---RD rd=reuqest.getRD("two.jsp");
rd.include(request,response);




Why JSTL ? JSP standard tag library
When JSP standard actions are in-sufficient to solve requirements ,
w/o writing scriptlets --- use additional standard actions --- supplied as
JSTL actions
JSP standard Tag Library
--- has become standard part of J2EE specs from version 1.5 onwards.
---It's support exists in form of a JAR ---
1.  jstl-1.2.jar
For using JSTL steps
1.  Copy above JAR into your run-time classpath(copy jars either in
    <tomcat_home>/lib OR <web-inf>/lib
2.  Use taglib directive to import JSTL tag library into  JSP pages.
tag=action
tag library=collection of tags
supplier = JSTL vendor(specification vendor=Sun, JAR vendor=Sun/any J2EE
compliant web/app server)
jstl.jar --- consists of Tag implementation classes
Tag libr-   TLD -- Tag library descriptor -- desc of tags -- how to use
tags
<%@ taglib uri="URI of JSTL tag lib" prefix="tag prefix" %>

eg --- To import JSTL core lib
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>




3.  Invoke JSTL tag/action
3.1. eg
<c:set var="abc" value="${param.f1}"  />
WC  :
pageContext.setAttribute("abc",request.getParameter("f1"))

WC invokes --- session.setAttribute("abc",request.getparameter("f1"));

menaing of <c:set> sets the specified attr to specified scope.


<c:set var="details" value="${sessionScope.abc}" />
WC
pageContext.setAttribute("details",session.getAttribute("abc"));
```

```
4.  <c:remove var="abc" scope="request"/>
WC ---request.removeAttribute("abc") ---removes the attr from req scope.



4.1. JB --- ShopBean -- property --
private AL<Category> categories; --g & s

<c:forEach var="cat" items="${sessionScope.shop.listCategories()}">
${cat}<br/>
</c:forEach>

WC invokes ---

for(Category cat : session.getAttribute("shop").listCategories())
  out.print(cat);

eg :
<c:forEach var="acct" items="${sessionScope.my_bank.acctSummary}">
${acct.acctID} ${acct.type} ${acct.balance} <br/>
</c:forEach>


http://localhost:8080/day6_web/close_acct.jsp?acId=101

<input type="submit" name="btn" value="Withdraw"

      formaction="transactions.jsp" /></td>
                                   <td><input type="submit" name="btn"
value="Deposit"

      formaction="transactions.jsp" /></td>

<%
   request.getPrameter("btn").equals("Deposit") ---

%>
<c:if test="boolean val">
....
</c:if>

<c:if test="${param.btn eq 'Deposit'}">
  in deposit
</c:if>
<c:if test="${param.btn eq 'Withdraw'}">
  in withdraw
</c:if>

http://localhost:8080/day6_web/transactions.jsp?acId=102&amount=500&btn=De
posit
```

```
<c:redirect url="${sessionScope.my_bank.closeAccount()}"/>
WC ---
response.sendRedirect(session.getAttribute("my_bank").closeAccount());
```

4.2. JSTL action --- for URL rewriting

```
<c:url var="attr Name" value="URL to be encoded"
scope="page|request|session|application"/>

eg : <c:url var="abc" value="next.jsp" />
WC invokes --- pageContext.setAttribute("abc",resp.encodeURL("next.jsp"));

<a href="${abc}">Next</a>
```

How to set session tm out ?

1. programmatically --- using Java API
From HttpSession --- setMaxInactiveInterval(int secs)
2. declarativally -- either using Java annotations OR using XML config files (web.xml)

```
JSP Syntax (2.x)

JSP implicit objs : accessible from scriptlets(<% java code %>  n
expressions (<%= dynamic expr %>)
request,response, session, config,out,application,page , pageContext,
exception
eg : one.jsp --- one_jsp.java
<%= page %>
o/p  : org.apche.jsp.one_jsp@5375675
page => instance of translated page class

pageContext : JSP implicit obj : accessible from scriptlets n exprs

javax.servlet.jsp.PageContext class --abstract --- concrete sub class ---
server supplied jar (jasper.jar)
--PageContextImpl --current page environment

It holds the references of all other implicit objects
eg : request,response, session, config,out....
Methods : getSession, getOut,getRequest.....
+ to store page scoped attr.


Which is better alternative to <%= .... %> JSP exprs ?
EL syntax
${...} : added directly in JSP body


EL implicit objs (names of maps created by WC , except pageContext)
param,pageScope,requestScope,sessionScope,applicationScope,cookie,initPara
m
eg : ${param.email}
WC : out.print(param.get("email"));
OR : request.getParameter("email") --> sent to clnt


Solve : ${requestScope.user_info}
WC : request.getAttribute("user_info") ---> sent to clnt

${cart} :
WC : pageContext.getAttribute("cart") --null
request.getAttribute("cart") --null
session.getAttribute("cart") --not null --to string --sent to clnt



eg : In one.jsp
<%
    pageContext.setAttribute("nm",val);
%>

How to acces this attribute from one.jsp ?
Options
1.  ${nm}
```

```
2.  ${pageContext.nm}
3.  ${page.nm}
4.  ${pageScope.nm}
5.  <%= pageContext.getAttribute("nm") %>

Ans : 1,4,5



Q. How to get a cookie value , with a name JSESSIONID ?

JSP expression
<%= session.getId() %>


OR
via cookie : EL syntax
${session.id} : wrong(since session IS NOT EL impl obj)
${pageContext.session.id} => pageContext.getSession().getId() --> sent to
clnt
OR
${cookie.JSESSIONID.value} => cookie.get("JSESSIONID").getValue()  -->
sent to clnt



pageContext : can be accessed from
1.  declaration
2.  expression
3.  EL
4. scriptlets
Ans : 2,3,4

Various uses of pageContext
Solve

eg : ${pageContext.session.id} --
WC : pageContext.getSession().getId() --> sent to clnt

What will be o/p for
http://host:port/day8/one.jsp
In one.jsp :
${pageContext.request.contextPath} --
WC : pageContext.getRequest().getContextPath() ---> sent to clnt
o/p : /day8


How will you get the value of session time out ?
?
1.  session = HttpSession
2.  sessionScope = name of the map containing session scoped attrs
Ans : 1
HOW ? ${pageContext.session.maxInactiveInterval} =>
pageContext.getSession().getMaxInactiveInterval()
```

```
JSP Expression : <%= session.getMaxInactiveInterval() %>


How to set session scoped attribute ?

<%
   session.setAttribute("nm",val);
%>
How to retrieve using following options ?
1.  session => HttpSession (Hint : JSP expression)
<%= session.getAttribute("nm") %>

OR
2.  sessionScope = name of the map containing session scoped attrs
Better option (EL syntax) :
${sessionScope.nm)
OR
${nm}
```

Entity Types :

1. If an object has its own database identity (primary key value) then it’s type is Entity Type.
2. An entity has its own lifecycle. It may exist independently of any other entity.
3. An object reference to an entity instance is persisted as a reference in the database (a foreign key value).

eg : College is an Entity Type. It has it’s own database identity (It has primary key).

Value Types :

1. If an object don’t have its own database identity (no primary key value) then it’s type is Value Type.
2. Value Type object belongs to an Entity Type Object.
3. It’s embedded in the owning entity and it represents the table column in the database.
4. The lifespan of a value type instance is bounded by the lifespan of the owning entity instance.

Different types of Value Types

Basic, Composite, Collection Value Types :

1. Basic Value Types :

Basic value types are : they map a single database value (column) to a single, non-aggregated Java type.
Hibernate provides a number of built-in basic types.
String, Character, Boolean, Integer, Long, Byte, … etc.

2.

Composite Value Types :
In JPA composite types also called Embedded Types. Hibernate traditionally called them Components.

2.1. Composite Value type looks like exactly an Entity, but does not own lifecycle and identifier.

Annotations Used

1. @Embeddable :

Defines a class whose instances are stored as an intrinsic part of an owning entity and share the identity of the entity. Each of the persistent properties or fields of the embedded object is mapped to the database table for the entity. It doesn't have own identifier.
eg : Address is eg of Embeddable
Student HAS-A Address(eg of Composition --i.e Address can't exist w/o its owning Entity i.e Student)
College HAS-A Address (eg of Composition --i.e Address can't exist w/o its owning Entity i.e College)
BUT Student will have its own copy of Address & so will College(i.e Value Types don't support shared reference)

2. @Embedded :

Specifies a persistent field or property of an entity whose value is an instance of an embeddable class. The embeddable class must be annotated as Embeddable.
eg : Address is embedded in College and User Objects.

3. @AttributesOverride :
Used to override the mapping of a Basic (whether explicit or default) property or field or Id property or field.

In Database tables observe the column names. Student table having STREET_ADDRESS column and College table having STREET column. These two columns should map with same Address field streetAddress.
@AttributeOverride gives solution for this.
To override multiple column names for the same field use @AtributeOverrides annotation.
eg : In Student class :

```
@Embedded
 @AttributeOverride(name="streetAddress",
column=@Column(name="STREET_ADDRESS"))
private Address address;
```

where , name --POJO property name in Address class

4.
Collection Value Types :
Hibernate allows to persist collections.
But Collection value Types can be either collection of Basic value types, Composite types and custom types.
eg :
Collection mapping means mapping group of values to the single field or property. But we can't store list of values in single table column in database. It has to be done in a separate table.

eg : Collection of embeddables

```
@ElementCollection
      @CollectionTable(name="CONTACT_ADDRESS",
joinColumns=@JoinColumn(name="USER_ID"))
      @AttributeOverride(name="streetAddress",
column=@Column(name="STREET_ADDRESS"))
      private List<ContactAddress> address;
```

eg : collection of basic type

```
      @ElementCollection
      @CollectionTable(name="Contacts",
joinColumns=@JoinColumn(name="ID"))
      @Column(name="CONTACT_NO")
      private Collection<String> contacts;
```

What is Hibernate ?
0. Complete solution to the problem of managing persistence in Java.
1. ORM tool.(Object Relational Mapping) used mainly in data access layer or DAO layer.
2. Provides automatic & transperent persistence.
3. JPA(Java Persistence API) implementor

JPA vs Hibernate
JPA ---standard part of J2EE specification --vendor --J2EE / Jakarta EE(sun/oracle/Eclipse)
Implementation classes -- JAR ---hibenrate core JARs(implementor of JPA)

Provides automatic & transparent persistence framework to store & retrieve data from database.
Open Source Java based framework founded by Gavin King in 2001, hosted on hibernate.org
Currently hosted on sourceforge.net
Java Persistence API (JPA) compliant
Current version Hibernate 5.x / 6.x
Other popular ORM Frameworks
EclipseLink,iBATIS,Kodo etc.

WHY Hibernate?

It mediates the applications interaction with a relational database, leaving the developer free to concentrate on the business problem at hand.

J2EE developer does not have to use JDBC API & manage data persistence at RDBMS level.
No need to go to Table/Query/Column level.
One has to bootstrap Hibernate framework , create transient(=not yet persistent) POJOs & then rely entirely on Hibernate frmwork to manage persistence

ref : why hibernate readme

Details

There is huge mismatch between Object & Relational world.
Formally referred as -- Object-Relational Impedance Mismatch' (sometimes called the 'paradigm mismatch)

Important Mismatch Points
1. Granularity
2. Sub Types or inheritance n polymorphism
3. Identity
4. Associations
5. Data Navigation

Cost of Mismatch
1. SQL queries in Java code
2. Iterating through ResultSet & mapping it to POJOs or entities.
3. SQL Exception handling.

4. Transaction management
5. Caching
6. Connection pooling
7. Boiler plate code

Hibernate Frmwork --- popular ORM Tool ---JPA (Java perssitence API) provider

Hibernate 4.x --- JPA compliant --- Java persistence API --- Its part of J2EE specifications. ---Is fully JPA compliant
BUT it also has additional services / annotations --- specific to Hibernate.

Dev MUST add hibernate JARs ---while deploying appln on web server. Need not add JPA provider JARs , while working on appln server.

Transparent persistence provider.(As POJOs or Entities are not bound to any Persistence API --- its written completely independent of Persistence Provider.)

--Fully supports OOP features --- association,inheritance & polymorphism

--can persist object graphs , consisting of asso. objects

--caches data which is fetched repeatedly (via L1 & L2 cache) -- thus reduces DB traffic(L1 cache - at session level -- built in. L2 cache - pluggable) (More on caching at end of document)

--supports lazy loading -- thus increases DB performance.
(Meaning --- Lazy fetchingThe associated object or collection is fetched lazily, when its first accessed. This results in a new request to the database (unless the associated object is cached). Eager fetchingThe associated object or collection is fetched together with the owning object, using an SQL outer join, and no further database request is required.

--supports Objectified version of SQL -- HQL --works on objects & properties
--Hibernate usually obtains exactly the right lock level automatically . so developer need not worry about applying Read/Write lock.

Some basics

1. Hibernate uses runtime reflection to determine the persistent properties of a class.

2.
The objects to be persisted(called as POJO or Entity) are defined in a mapping document or marked with annotations.

Either these HBM XML docs or annotations serves to describe the persistent fields and associations, as well as any subclasses or proxies of the persistent object.

3.

The mapping documents or annotations are compiled at application startup time and provide the framework with necessary information for a persistent class.
4.

What is Hibernate config.?

An instance of Hib Configuration allows the application to specify properties and mapping documents to be used at the frmwork start-up.
The Configuration : initialization-time object.

5.

SessionFactory is created from the compiled collection of mapping documents .
The SessionFactory provides the mechanism for managing persistent classes, the Session interface.

6.
A web application or Java SE apllication will create a single Configuration, build a single instance of SessionFactory and then instantiate multiple Sessions in threads servicing client requests.

SessionFactory : immutable and does not reflect any changes done later to the Configuration.

7. The Session class provides the interface between the persistent data store and the application.
The Session interface wraps a JDBC connection, which can be user-managed or controlled by Hibernate.

Hibernate Session

A Hibernate Session is a set of managed entity instances that exist in a particular data store.

Managing an Entity Instances Life Cycle

You manage entity instances(or POJOs) by invoking operations on the entity/POJO using EntityManager/Session instance.

Entity instances are in one of four states (2 imp aspects of it : its asso. with the hibernate session & sync of its state with the underlying DB)

States : new or transient , managed or persistent, detached, removed.

New entity instances have no persistent identity and are not yet associated with a hib. session (transient)

Managed entity instances have a persistent identity and are associated with a hib. session.(persistent : via save() or saveOrUpdate()) Changes to DB will be done when tx is commited.

Detached entity instances have a persistent identity and are not currently associated with a persistence context/Hib session.

Removed entity instances have a persistent identity, are associated with a persistent context and are scheduled for removal from the data store.(removed via session.delete(obj))

Introduction to Hibernate Caching

While working with Hibernate web applications we will face so many problems in its performance due to database traffic. That too when the database traffic is very heavy . Actually hibernate is well used just because of its high performance only. So some techniques are necessary to maintain its performance.

Caching is the best technique to solve this problem.

The performance of Hibernate web applications is improved using caching by optimizing the database applications.

The cache actually stores the data already loaded from the database, so that the traffic between our application and the database will be reduced when the application want to access that data again.
At maximum the application will work with the data in the cache only. Whenever some another data is needed, the database will be accessed. Because the time needed to access the database is more when compared with the time needed to access the cache. So obviously the access time and traffic will be reduced between the application and the database.
Here the cache stores only the data related to current running application. In order to do that, the cache must be cleared time to time whenever the applications are changing.

Difference in get & load
1. Both use common API (i.e load or get(Class c,Serializable id))
Ret type = T
In get --- if id doesn't exist --- rets null
In load --- if id doesn't exist & u are accessing it from within hib session --- throws ObjectNotFoundExc
2. In get --- Hibernate uses eager fetching policy ---- meaning will generate select query always & load the state from DB in persistent POJO ref. --- so even if u access the same from within the session(persistent pojo) or outside (detached) the hib session --- NO EXCEPTION(proxy + state)

3. In load --- Hib uses lazy fetching policy ---- meaning it will , by default NOT generate any select query --- so what u have is ONLY PROXY(wrapper ---with no state loaded from DB) --- on such a proxy --- if u access anything outside the hib session(detached) ----

U WILL GET ---LazyInitializationExc
Fix --- 1. Change fetch type --- to eager (NOT AT ALL reco.=> no caching , disabling L1 cache)

4. If u want to access any POJO in detached manner(i.e outside hib session scope) -

fire non-id get method from within session & then hib has to load entire state from DB ---NO LazyInitializationExc

Session API update Vs merge
Both methods transition detached object to persistent state.

Update():- if you are sure that the session does not contain an already persistent instance with the same identifier then use update to save the data in hibernate. If session has such object with same id , then it throws --- org.hibernate.NonUniqueObjectException: a different object with the same identifier value was already associated with the session:

Merge():-if you want to save your modificatiions at any time with out knowing about the state of an session then use merge() in hibernate.

Lazy fetching (becomes important in relationships or in Load Vs get)
When a client requests an entity(eg - Course POJO) and its associated graph of objects(eg -Student POJO) from the database, it isnt usually necessary to retrieve the whole graph of every (indirectly) associated object. You wouldnt want to load the whole database into memory at once; eg: loading a single Category shouldnt trigger the loading of all Items in that category(one-->many)

What is Session?

Represents a wrapper around pooled out jdbc connection.

Session object is persistance manager for the hibernate application

Session object is the abstraction of hibernate engine for the Hibernate application

Session object provides methods to perform

CRUD operations

Example

save()	-	Inserting the record
get() / load()	-	Retrieveing the record
update()	-	Updating the record
delete()	-	Deleting the record

What is SessionFactory?

It is a factory(provider) of session objects.

we use sessionfactory object to create session
object

It is a heavy weight object, therefore it has to
be created only once for an application(typically @ appln start up time) --- typically one per DB per web application.

Its immutable --- Once SF is created , changes made to hibernate.cfg.xml
will not be auto reflected in SF.

What is Configuration Object ?

Configuration object is used to create the
SessionFactory object.

Object Oriented Representation of Hibernate
configuration file and
mapping files(or annotations) is nothing but Configuration object.

When we call configure() method on configuration
object ,hibernate configuration file(hibernate.cfg.xml from run time
classpath) and mapping
files (or resources) are loaded in the memory.

Why connection pooling?

Java applications should use connection pools because :
- Acquiring a new connection is too expensive
- Maintaining many idle connections is expensive
- Creating prepared statements is expensive

Hibernate provides basic or primitive connection pool -- useful only for
classroom testing.
Replace it by 3rd party vendor supplied connection pools(eg Apache or C3P0
or hikari in spring boot) for production grade applications.

Natural Key Vs Surrogate Key

If u have User reg system -- then u have a business rule that --- user email must be distinct. So if u want to make this as a prim key --then user will have to supply this during regsitration.
This is called as natural key. Since its value will be user supplied , u cant tell hibernate to generate it for u---i.e cant use @GeneratedValue at all.

Where as -- if u say I will reserve user id only for mapping purposes(similar to serial no), it need not come from user at all & can definitely use hib. to auto generate it for u---this is ur surrogate key & can then use @GeneratedValue.

What is Session? (org.hibernate.Session : interface)

Session object is persistance manager for the
hibernate application

Session object is the abstraction of hibernate
engine for the Hibernate application

Session object provides methods to perform
CRUD operations

Session just represents a thin wrapper around pooled out DB connection.

Session is associated implicitely with L1 cache (having same scope as the
session lifetime) , referred as Persistence context.

Example of CRUD

save()	-	Inserting the record
get() / load()	-	Retrieveing the record
update()	-	Updating the record
delete()	-	Deleting the record

public void delete(Object ref) throws HibernateExc
ref -- either persistent or detached pojo ref.

What is SessionFactory? (org.hibernate.SessionFactory : interface)

It is a provider(factory) of session objects.

We use sessionfactory object to create session
object(via openSession or getCurrentSession).
It is singleton(1 instance per DB / application) ,immutable,inherently
thrd safe.
It is a heavy weight object, therefore it has to
be created only once in the beginning for an application & that too at the
very beginning.
It is associtated with L2 cache(must be explicitely enabled)

What is Configuration Object ?(org.hibernate.cfg.Configuration)

In earlier versions of hibernate , Configuration object was used to create
the
SessionFactory object.

Object Oriented Representation of Hibernate
configuration file and
mapping file is nothing but Configuration object.

When we call configure() method on configuration
object ,hibernate configuration file(hibernate.cfg.xml placed in run time
classpath) and mapping

```
files are loaded in the memory.






Persistent Object Life cycle
----------------------------------

1.  Transient State
----------------------
An object is said to be in transient state if it
 is not associated
 with the session,and has no matching record
 in the database table.


For example
-----------------
Account account=new Account();

account.setAccno(101);
account.setName("Amol");
account.setBalance(12000);


2.  Persistent State
-----------------------
An object is said to be in persistent state if
it is associated with session
object (L1 cache) and will result into a matching record in
 the databse table.(i.e upon commit)

session.save(account);tx.commit();

or
Account account=session.get(Account.class,102);
OR via HQL


Note
------
When the object is in persistent state it
 will be in synchronization with the matching
 record i.e
 if we make any changes to the state of
  persistent object it will be
  reflected in the database.(after commiting tx)  -- i.e automatic dirty
checking will be performed.


3.  Detached state
---------------------
```

```
Object is not associated with session but
has matching record in the database table.
 If we make any changes to the state of
  detached object it will NOT  be
  reflected in the database.



session.clear();
session.evict(Object);
session.close();



Note :
-------
By calling update method on session object it
will go from detached
state to persistent state.

By calling delete method on session object it will go
from persistenet state to
transient  state.


Explain the following methods of Session API

public void persist(Object ref) -- Persists specified transient POJO on
underlying DB , upon comitting the transaction.

--------------------------------------
void clear()
--------
When clear() is called on session object all  the objects associated
with the session object become detached.
 But Databse Connection is not closed.
(Completely clears the session. Evicts all loaded instances and cancel all
pending saves, updates and deletions)

void close()
--------
When close() is called on session object all
the objects associated with the session object become detached and
also closes the  Database Connection.

public void evict(Object ref)
--------
It detaches a particular persistent object
detached or disassociates from the session.
(Remove this instance from the session cache. Changes to the instance will
not be synchronized with the database. )

void flush()
--------
```

```
When the object is in persistent state ,
whatever changes we made to the object
state will be reflected in the databse only
 at the end of transaction.

If we want to reflect the changes before the end of transaction
(i.e before commiting the transaction )
 call the flush method.
(Flushing is the process of synchronizing the underlying DB state with
persistable state of session cache )

boolean contains(Object ref)
------------
The method indicates whethere the object is
associated with session or not.

void refresh(Object ref) -- ref --persistent or detached
-----------
This method is used to get the latest  data from database and make
corresponding modifications to the persistent object state.
(Re-read the state of the given instance from the underlying database)
----------------
public void update(Object ref)
Note  :-

If object is in persistent state no
need of calling the update method .
As the object is in sync with the
database whatever changes made to the object
will be reflect to database at the
end of transaction.
eg --- updateAccount(Account a,double amt)
{
    sess, tx
    sop(a);set amt
    sess.update(a);
    sop(a);
}







When the object is in detached state record
 is present in the table
but object is not in sync with database,
therefore update() method can be called
to update the record in the table

Which exceptions update method can raise?
1.  StaleStateException -- If u are trying to update a record (using
    session.update(ref)), whose id doesn't exist.
```

i. e update can't transition from transient --->persistent
It can only transition from detached --->persistent.

eg -- update_book.jsp -- supply updated details + id which doesn't exists on db.

2. NonUniqueObjectException -- If there is already persistence instance with same id in session.
eg -- UpdateContactAddress.java

----
public Object merge(Object ref)
Can Transition from transient -->persistent & detached --->persistent.
Regarding Hibernate merge
1. The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().
2. API of Session
Object merge(Object object)
3.
Copies the state of the given object(can be passed as transient or detached) onto the persistent object with the same identifier.
4. If there is no persistent instance currently associated with the session, it will be loaded.
5. Return the persistent instance. If the given instance is unsaved, save a copy of and return it as a newly persistent instance. The given instance does not become associated with the session.
6. will not throw NonUniqueObjectException --Even If there is already persistence instance with same id in session.

----

public void saveOrUpdate(Object ref)

The method persists the object (insert) if matching record is not found (& id inited to default value) or fires update query
If u supply Object , with non-existing ID -- Fires StaleStateException.

lock()
----
when lock() method is called on the
session object for a persistent object ,
untill the transaction is commited in
the hibernate application , externally the matching record in the table cannot be modified.

session.lock(object,LockMode);

eg - session.lock(account,LockMode.UPGRADE);

There are many advantages of Hibernate Framework over JDBC

1) Opensource , Lightweight : Hibernate framework is opensource & lightweight.
2) Fast performance: The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

Third type of cache is --query level cache.(not implicitely enabled)

3) Database Independent query: HQL (Hibernate Query Language) / JPQL (Java persistence query language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, If database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic table creation: Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Simplifies complex join: To fetch data form multiple tables is easy in hibernate framework.

eg : To display the course names ordered by desc no of participants (many-to-many)

```
select c.name from dac_courses c inner join course_studs cs on c.id = cs.
c_id inner join dac_students s on cs.s_id = s.stud_id group by c.id order
by count(*) desc;
 JPQL -- select c from Course c join fetch c.students group by c.id order
by count(*) desc
```

6) Provides query statistics and database status: Hibernate supports Query cache and provide statistics about query and database status.

7. Hibernate translates checked SQLException to un checked org.hibernate.HibernateException(super cls of all hibernate related errs) ---so that prog doesn't have to handle excs.

Advantages of hibernates:

1. Hibernate supports Inheritance, Associations, Collections.
2. In hibernate if we save the derived class object, then its base class object will also be stored into the database, it means hibernate supporting inheritance
3. Hibernate supports relationships like One-To-Many,One-To-One, Many-To-Many-to-Many, Many-To-One
4. This will also supports collections like List,Set,Map (Only new collections)
5. In jdbc all exceptions are checked exceptions, so we must write code in try, catch and throws, but in hibernate we only have Un-checked exceptions, so no need to write try, catch, or no need to write throws. Actually in hibernate we have the translator which converts checked to Un-checked ;)

6. Hibernate has capability to generate primary keys automatically while we are storing the records into database
7. Hibernate has its own query language, i.e hibernate query language which is database independent

So if we change the database, then also our application will works as HQL is database independent
HQL contains database independent commands

8. While we are inserting any record, if we dont have any particular table in the database, JDBC will rises an error like View not exist, and throws exception, but in case of hibernate, if it not found any table in the database this will create the table for us ;)
9. Hibernate supports caching mechanism by this, the number of round trips between an application and the database will be reduced, by using this caching technique an application performance will be increased automatically.

Hibernate supports annotations, apart from XML

10. Hibernate provided Dialect classes, so we no need to write sql queries in hibernate, instead we use the methods provided by that API.
11. Getting pagination in hibernate is quite simple.

Hibernate API

0. SessionFactory API
getCurrentSession vs openSession

public Session openSession() throws HibernateExc
opens new session from SF,which has to be explicitely closed by prog.

public Session getCurrentSession() throws HibernateExc
Opens new session , if one doesn't exist , otherwise continues with the exisitng one.
Gets automatically closed upon Tx boundary or thread over(since current session is bound to current thread --mentioned in hibernate.cfg.xml property ---current_session_context_class ---thread)

1. CRUD logic (save method)
API (method) of org.hibernate.Session
public Serializable save(Object o) throws HibernateException

I/P ---transient POJO ref.
save() method auto persists transient POJO on the DB(upon committing tx) & returns unique serializable ID generated by (currently) hib frmwork.

2. Hibernate session API -- for data retrieval
API (method) of org.hibernate.Session
public <T> T get(Class<T> c,Serializable id) throws HibernateException
T -- type of POJO
Returns --- null -- if id is not found.
returns PERSISTENT pojo ref if id is found.

Usage of Hibernate Session API's get()
```
int id=101;
BookPOJO b1=hibSession.get(BookPOJO.class,id);

BookPOJO b1=(BookPOJO)hibSession.get(Class.forName("pojos.BookPOJO"),id);
```

3. Display all books info :

using HQL -- Hibernate Query Language/JPQL --- Objectified version of SQL --- where table names will be replaced by POJO class names & table col names will replaced by POJO property names.
(JPA--- Java Persistence API compliant syntax --- JPQL)
eg --- HQL --- "from BookPOJO"
eg JPQL -- "select b from BookPOJO b"

3.1. Create Query Object --- from Session i/f
<T> org.hibernate.query.Query<T> createQuery(String queryString,Class<T> resultType)

eg : Query<Book> q=hs.createQuery(hql/jpql,Book.class);

3.2. Execute query to get List of selected PERSISTENT POJOs
API of org.hibernate.query.Query i/f
(Taken from javax.persistence.TypedQuery<T>)

```
List<T> getResultList()

Execute a SELECT query and return the query results as a generic List<T>.
T -- type of POJO / Result

eg : hs,tx
String jpql="select b from Book b";
try {
    List<Book> l1=hs.createQuery(jpql,Book.class).getResultList();
}

 Usage ---
String hql="select b from BookPOJO b";
List<BookPOJO> l1=hibSession.createQuery(hql).getResultList();


4.  Passing IN params to query. & execute it.
Objective : Display all books from specified author , with price <
specified price.
API from org.hibernate.query.Query i/f

Query<R> setParameter(String name,Object value)

Bind a named query parameter using its inferred Type.
name -- query param name
value -- param value.I

String hql="select b from BookPOJO b where b.price < :sp_price and
b.author = :sp_auth";

How to set IN params ?
org.hibernate.query.Query<T> API
public Query<T> setPrameter(String pName,Object val)

List<Book> l1 =
hibSession.createQuery(hql,Book.class).setParameter("sp_price",user_price)
.setParameter("sp_auth",user_auth).getResultList();


Objective --Offer discount on all old books
i/p -- date , disc amt

5.  Updating POJOs --- Can be done either with select followed by update
    or ONLY with update queries(following is eg of 2nd option--Bulk update
    scenario)
Objective : dec. price of all books with author=specified author.


String jpql = "update BookPOJO b set b.price = b.price - :disc where
b.author = :au and b.publishDate < :dt ";

set named In params
exec it (executeUpdate) ---
```

```
int updateCount= hs.createQuery(hql).setParameter("disc",
disc).setParameter("dt", d1).executeUpdate();
```

---This approach is typically NOT recommended often, since it bypasses L1 cache . Cascading is not supported. Doesn't support optismistic locking directly.

6. Delete operations.
API of org.hibernate.Session
--void delete(Object object) throws HibernateException
---POJO is marked for removal , corresponding row from DB will be deleted after comitting tx & closing of session.

OR
5.5
One can use directly "delete HQL" & perform deletions.(Bulk delete)
eg

```
int deletedRows = hibSession.createQuery ("delete Subscription s  WHERE
s.subscriptionDate < :today").setParameter ("today", new Date
()).executeUpdate ();
```

API of org.hibernate.query.Query<T>

1. Iterator iterate() throws HibernateException

Return the query results as an Iterator. If the query contains multiple results per row, the results are returned Object[].

Entities returned --- in lazy manner

Pagination

2. Query setMaxResults(int maxResults)

    Set the maximum number of rows to retrieve. If not set, there is no limit to the number of rows retrieved.

3. Query setFirstResult(int firstResult)

    Set the first row to retrieve. If not set, rows will be retrieved beginnning from row 0. (NOTE row num starts from 0)

eg --- List<CustomerPOJO> l1=sess.createQuery("select c from CustomerPOJO c").setFirstResult(30).setMaxResults(10).list();

4. How to count rows & use it in pagination techniques?

```
       int pageSize = 10;
    String countQ = "Select count (f.id) from Foo f";
    Query countQuery = session.createQuery(countQ);
    Long countResults = (Long) countQuery.uniqueResult();
    int lastPageNumber = (int) ((countResults / pageSize) + 1);

    Query selectQuery = session.createQuery("From Foo");
    selectQuery.setFirstResult((lastPageNumber - 1) * pageSize);
    selectQuery.setMaxResults(pageSize);
    List<Foo> lastPage = selectQuery.list();
```

5. org.hibernate.query.Query API

`<T> T getSingleResult()`

Executes a SELECT query that returns a single typed result.

Returns: Returns a single instance(persistent) that matches the query.

Throws:
    NoResultException - if there is no result
    NonUniqueResultException - if more than one result
    IllegalStateException - if called for a Java Persistence query language UPDATE or DELETE statement

6. How to get Scrollable Result from Query?

`ScrollableResults scroll(ScrollMode scrollMode) throws HibernateException`

Return the query results as ScrollableResults. The scrollability of the returned results depends upon JDBC driver support for scrollable ResultSets.

Then can use methods of ScrollableResults ---first,next,last,scroll(n) .

7. How to create Named query from Session i/f?
What is a named query ?
Its a technique to group the HQL statements in single location(typically in POJOS) and lately refer them by some name whenever need to use them. It helps largely in code cleanup because these HQL statements are no longer scattered in whole code.

Fail fast: Their syntax is checked when the session factory is created, making the application fail fast in case of an error.
Reusable: They can be accessed and used from several places which increase re-usability.

eg : In POJO class, at class level , one can declare Named Queries
```
@Entity
@NamedQueries
({@NamedQuery(name=DepartmentEntity.GET_DEPARTMENT_BY_ID, query="select d
from DepartmentEntity d where d.id = :id")}
public class Department{....}
```

Usgae
Department d1 = (Department)
session.getNamedQuery(DepartmentEntity.GET_DEPARTMENT_BY_ID).setInteger("i
d", 1);

8. How to invoke native sql from hibernate?
Query q=hs.createSQLQuery("select * from
books").addEntity(BookPOJO.class);
 l1 = q.list();

9. Hibernate Criteria API
A powerful and elegent alternative to HQL
Well adapted for dynamic search functionalities where complex Hibernate
queries have to be generated 'on-the-fly'.

Typical steps are -- Create a criteria for POJO, add restrictions ,
projections ,add order & then fire query(via list() or uniqueResult())

10. For composite primary key
Rules on prim key class
Annotation -- @Embeddable (& NOT @Entity)
Must be Serializable.
Must implement hashCode & equals as per general contract.

In Owning Entity class
Add usual annotation -- @Id.

1.1 Testing core api
persist ---
public void persist(Object transientRef)
---persists trasient POJO .

if u give some non-null id (existing or non-existing) while calling
persist(ref) --gives exc
org.hibernate.PersistentObjectException: detached entity passed to
persist:
why its taken as detached ? ---non null id.

11.
public Serializable save(Object ref)
save --- if u give some non-null id(existing or non-existing) while
calling save(ref) --doesn't give any exc.
Ignores ur passed id & creates its own id & inserts a row.

12. saveOrUpdate
public void saveOrUpdate(Object ref)
--either inserts/updates or throws exc.
null id -- fires insert (works as save)
non-null BUT existing id -- fires update (works as update)
non-null BUT non existing id -- throws StaleStateException --to indicate
that we are trying to delete or update a row that does not exist.

3.5
merge
public Object merge(Object ref)
I/P -- either transient or detached POJO ref.
O/P --Rets PERSISTENT POJO ref.

null id -- fires insert (works as save)
non-null BUT existing id -- fires update (select , update)
non-null BUT non existing id -- no exc thrown --Ignores ur passed id & creates its own id & inserts a row.(select,insert)

13. get vs load
& LazyInitilalizationException.

14. update
Session API
public void update(Object object)
Update the persistent instance with the identifier of the given detached instance.
I/P --detached POJO containing updated state.
Same POJO becomes persistent.

Exception associated :

1. org.hibernate.TransientObjectException: The given object has a null identifier:

i. e while calling update if u give null id. (transient ----X --- persistent via update)

2. org.hibernate.StaleStateException --to indicate that we are trying to delete or update a row that does not exist.
3.
org.hibernate.NonUniqueObjectException: a different object with the same identifier value was already associated with the session

6. public Object merge(Object ref)
Can Transition from transient -->persistent & detached --->persistent.
Regarding Hibernate merge

1. The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().
2. API of Session

Object merge(Object object)
3.
Copies the state of the given object(can be passed as transient or detached) onto the persistent object with the same identifier.

4. If there is no persistent instance currently associated with the session, it will be loaded.
5. Return the persistent instance. If the given instance is unsaved, save a copy of and return it as a newly persistent instance. The given instance does not become associated with the session.

6. will not throw NonUniqueObjectException --Even If there is already
   persistence instance with same id in session.

7.public void evict(Object persistentPojoRef)

It detaches a particular persistent object
detaches or disassociates from the session level cache(L1 cache)
(Remove this instance from the session cache. Changes to the instance will
not be synchronized with the database.)

8.
void clear()

When clear() is called on session object all the objects associated with
the session object(L1 cache) become detached.
But Databse Connection is not returned to connection pool.
(Completely clears the session. Evicts all loaded instances and cancel all
pending saves, updates and deletions)

9. void close()

When close() is called on session object all
the persistent objects associated with the session object become
detached(l1 cache is cleared) and also closes the Database Connection.

10. void flush()

When the object is in persistent state , whatever changes we made to the
object
state will be reflected in the databse only at the end of transaction.

BUT If we want to reflect the changes before the end of transaction
(i.e before commiting the transaction)
call the flush method.
(Flushing is the process of synchronizing the underlying DB state with
persistable state of session cache)

11. boolean contains(Object ref)

The method indicates whether the object is
associated with session or not.(i.e is it a part of l1 cache ?)

12.
void refresh(Object ref) -- ref --persistent or detached

This method is used to get the latest data from database and make
corresponding modifications to the persistent object state.
(Re-reads the state of the given instance from the underlying database

What is Maven ?

Build automation tool for overall project management.

It helps in
1. checking a build status
2. generating reports (basically javadocs)
3. setting up the automated build process and monitors the same.

Why Maven ?
It eases out source code compilation, distribution, documentation, collaboration with different teams .

Maven tries 2 describe

1. How a software is built.
2. The dependencies, plug-ins & profiles that the project is associated in a standalone or a distributed environment.

Vendor -- Apache

Earlier build tool -- Ant
Vendor -- Apache.

Ant disadvantages
1. While using ant , project structure had to be defined in build.xml. Maven has a convention to place source code, compiled code etc. So no need to provide information about the project structure in pom.xml file.

2.
Maven is declarative, everything you define in the pom.xml file.
No such support in ant.

3.
There is no life cycle in Ant, where as life cycle exists in Maven.

Maven advantages

4. Managing dependencies
5. Uses Convention over configuration - configuration is very minimal
6. Multiple/Repeated builds can be achieved.

7. Plugin management.
8. Testing - ability to run JUnit and other integration test suites.

What is POM? (Project Object Model)

It is the core element of any maven project.
Any maven project consists of one configuration file called pom.xml.
Location --In the root directory of any maven project.

It contains the details of the build life cycle of a project.

Contents
Dependencies used in the projects (Jar files)
Plugins used
Project version
Developers involved in the project
Build profiles etc.

Maven reads the pom.xml file, then executes the goal.

Elements of maven pom.xml file

1. project It is the root element of pom.xml file.
2. modelVersion It is the sub element of project. It specifies the modelVersion.
3. groupId It is the sub element of project. It specifies the id for the project group.(typically organization name)
4. artifactId It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, WARs.
5. version It is the sub element of project. It specifies the version of the artifact under given group.
6.

packaging -- defines packaging type such as jar, war etc.

7. name -- defines name of the maven project.
8. plugins ---compiler plugins , eclipse plugins
9. dependencies -- collection of dependencies for this project.

Within that --
dependency -- defines a specific dependency.(eg : hibernate dependency,spring web)

10. scope -- defines scope for this maven project. It can be compile, provided, runtime, test and system.

Goals in Maven
Goal in maven is nothing but a particular task which leads to the compiling, building and managing of a project. A goal in maven can be associated to zero or more build phases. Only thing that matters is the order of the goals defined for a given project in pom.xml. Because, the order of execution is completely dependent on the order of the goals defined.
eg : clean , build ,install ,test

What is a Maven Repository

A maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies(JARs) in the repositories. There are 3 types of maven repository:

Local Repository
Central Repository

Remote Repository

Maven searches for the dependencies in the following order:

Local repository then Central repository then Remote repository.
maven repositories

If dependency is not found in these repositories, maven stops processing and throws an error.

1. Maven Local Repository

Maven local repository is located in the file local system. It is created by the maven when you run any maven command.

By default, maven local repository is HOME / .m2 directory.
(Can be updated by changing the MAVEN_HOME/conf/settings.xml)

2) Maven Central Repository

Maven central repository is located on the web(Created by the apache maven community)

The path of central repository is: https://mvnrepository.com/repos/central

3) Maven Remote Repository

Maven remote repository is also located on the web. Some of libraries that are missing from the central repository eg JBoss library , Oracle driver etc, can be located from remote repository.

Maven Build Life Cycle
What is it ?
The sequence of steps which is defined in order to execute the tasks and goals of any maven project is known as build life cycle in maven.

Maven comes with 3 built-in build life cycles

Clean - this phase involves cleaning of the project (for a fresh build & deployment)
Default - this phase handles the complete deployment of the project
Site - this phase handles the generating the java documentation of the project.

Build Profiles in Maven

It is a subset of elements which allows to customize builds for particular environment. Profiles are also portable for different build environments.

Build environment basically means a specific environment set for production and development instances. When developers work on development phase, they use test database from the production instance and for the production phase, the live database will be used.

So, in order to configure these instances maven provides the feature of build profiles. Any no. of build profiles can be configured and also can override any other settings in the pom.xml

eg : profiles can be set for dev, test and production phases.

Installation (w/o IDE)
1. Download Maven from Apache (version 3.x)
2. Add MAVEN_HOME as environment variable
3. Add maven/bin under path (for easy accessibility)
4. Verify maven

```
  mvn -- version
```

  OR use m2e plug-in (a standard part of Eclipse for J2EE)