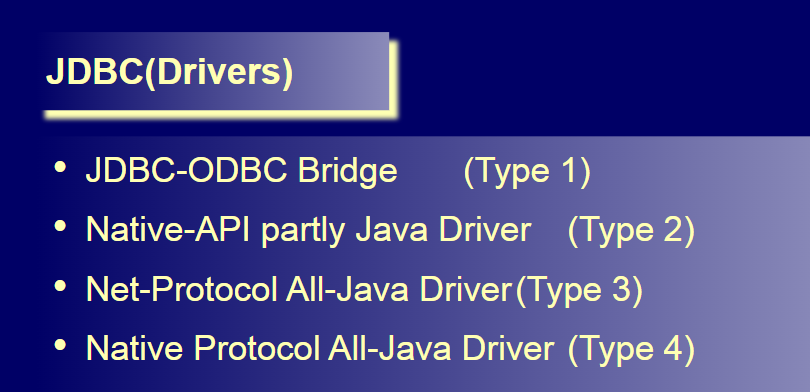
ADVANCED JAVA

Interview Questions

* public CallableStatement prepareCall(String invocationSyntax) throws SqlException
* servlet, read day2 sequence.txt
* sendRedirect vs requestDispatcher
* forward vs include

READ readme\_JDBC.txt in day1 in fast manner

JDBC Driver 🡪 A driver is a program that converts the Java method calls to the corresponding method calls understandable by the database in use.



Statement <--- PrepareStatement <--- CallableStatement

PST 🡪 no sql injection, can set in params,

CST 🡪 to execute stored proc $ func in db, can pass IN, OUT, IN Out params

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

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.

Default nature of ResultSet (i/f) object

is TYPE\_FORWARD (i.e can use only next() methods)

READ\_ONLY

**ResultSet is never NULL, it is either EMPTY or Populated**

To execute stored proc or fun 🡪 stmt.execute() public boolean execute() throws SE

To execute DML / DDL 🡪 stmt.executeUpdate() public int executeUpdate(String sql) throws SQLException

To execute Select 🡪 stmt.executeQuery() public ResultSet executeQuery(String sql) throws SQLException

How to do TX from JDBC API?

1. Start a Tx

Connection i/f method

void setAutoCommit(boolean flag:false)

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 : public void commit() throws SQLException

eg : cn.commit();

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

API : public void rollback() throws SQLException

eg : cn.rollback();

5. To continue , in a non transactional manner : 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

It will undo all changes made after the given Savepoint object was set.

**Version Java EE(Enterprise Edition) 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.

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...

J2EE server Vendors & Products

Apache -- Tomcat(web server) / Tomee (app server)

Oracle / Sun --- reference implementation --- Glassfish

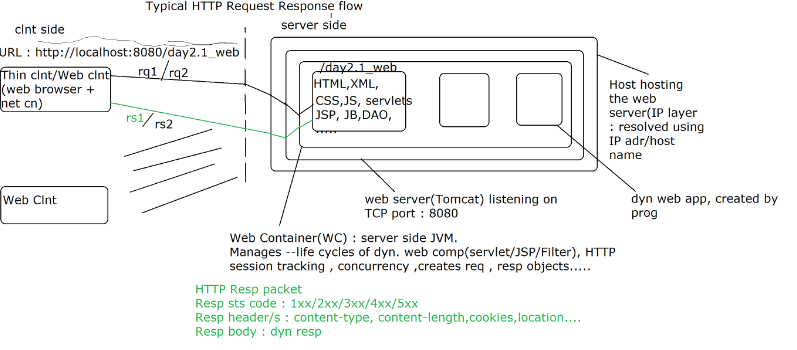
Red Hat -- JBoss (wild fly)

Oracle / BEA -- weblogic

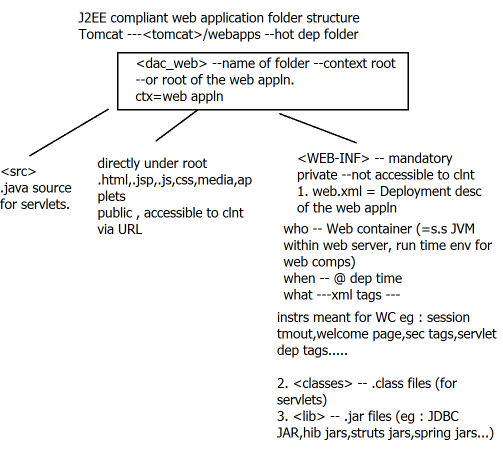
IBM -- Websphere

Interface is a way to achieve multiple inheritance in Java by allowing a class to inherit behaviors from multiple superclasses. On the other hand, a Specification is a higher-level concept that defines the design and architecture of a software system.

**J2EE**



Read notebook page for better understanding



What is **Web container** --- (WC) & its jobs

1. Server side JVM residing within web server.

Its a 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...

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......

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)

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

WC creates SingleTon instance of servlet.api.jar

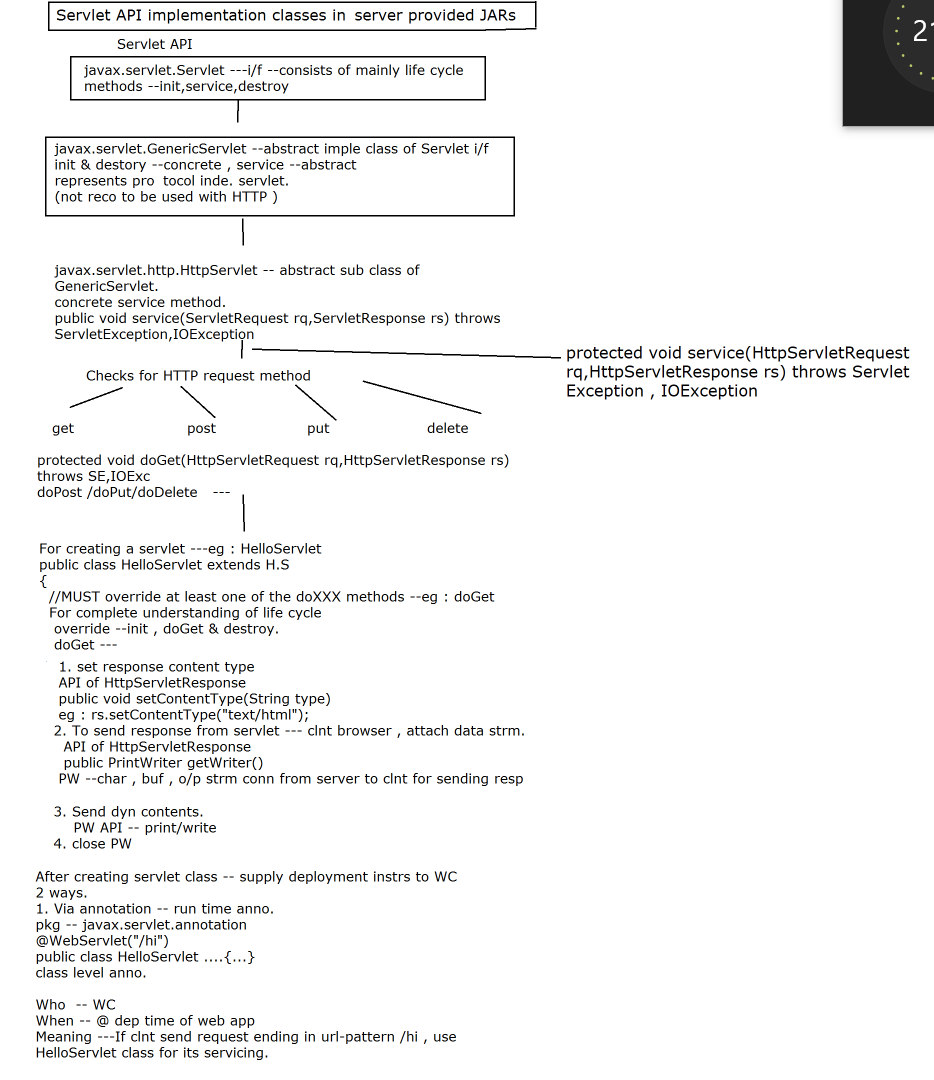
Servlet (i/f)

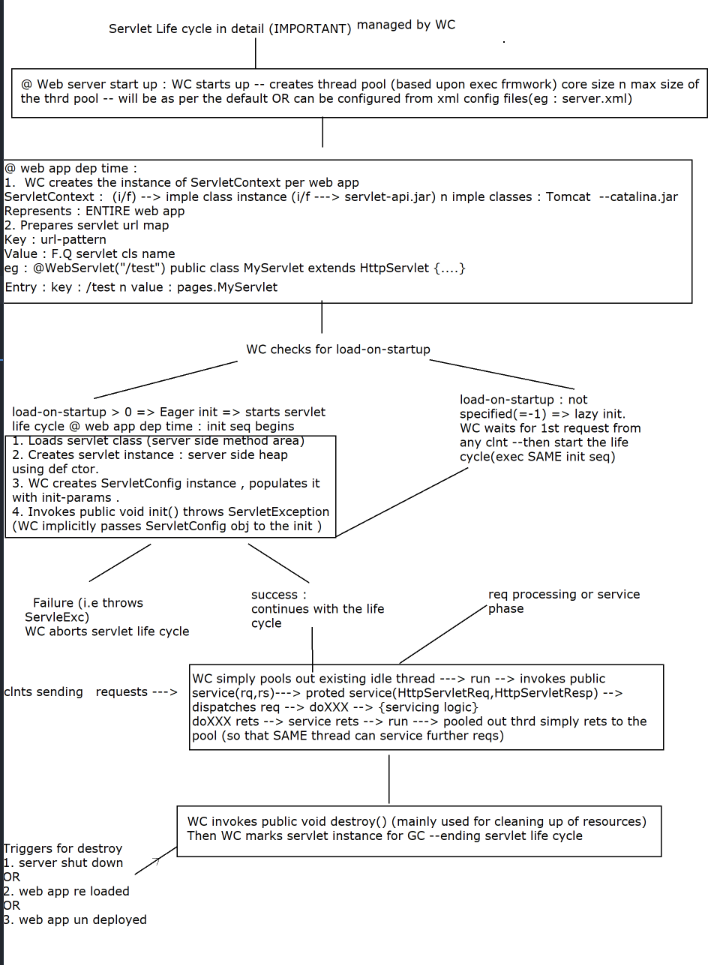
↑

Impl GenericServlet

↑

Extends HttpServlet





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.

What is default loading policy of WC for servlets ? lazy (i.e WC will start life cycle of the servlet :only after clnt sends the 1st request to the servlet)

Can you change it to eager ? Yes

Use Case : Typically for time consuming(heavy weight) inits.

eg : setting up DB conn, setting up spring frmwork....

HOW ?

@WebServlet (value="/test",loadOnStartup=1)

.....

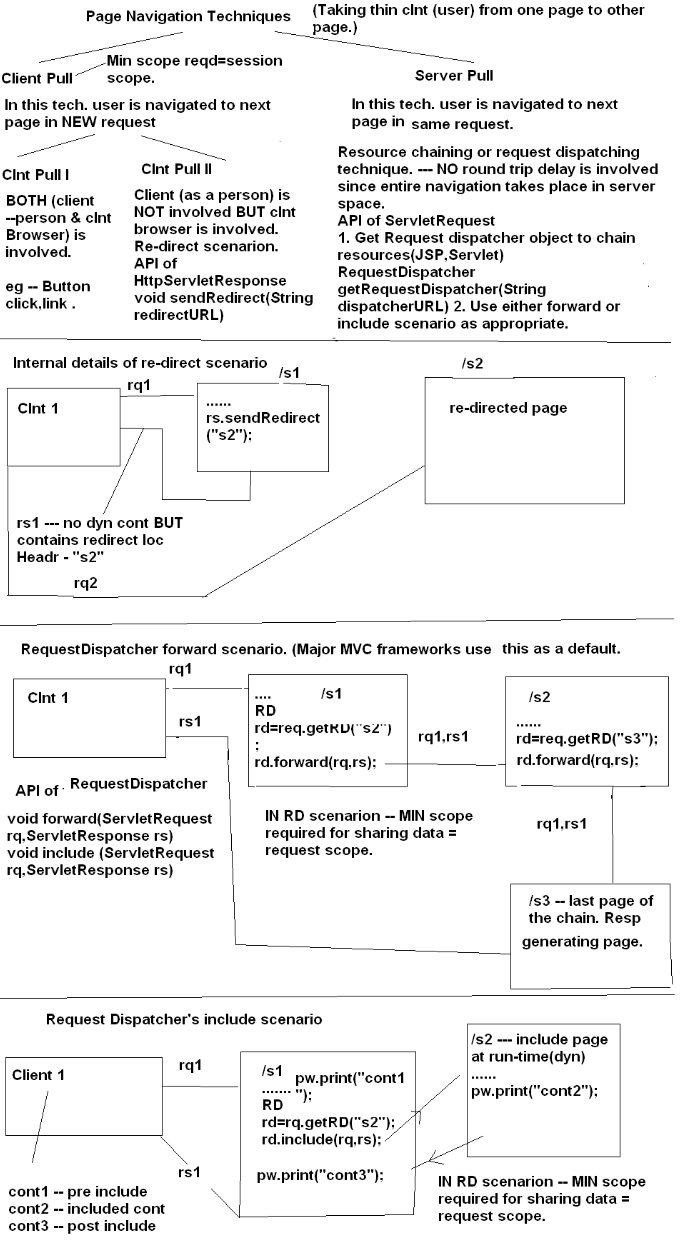
OR

xml tag :

<load-on-startup>1</load-on-startup>

**Page Navigation Techniques**

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



2 Ways

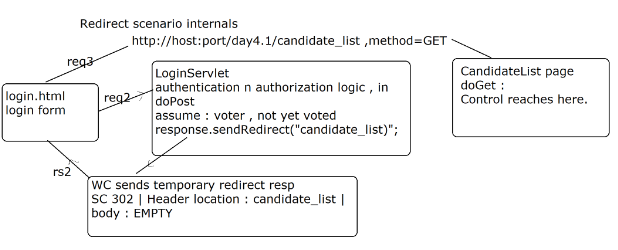
1. Client Pull

Min scope reqd for sharing data is Session scope

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 submit 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");

**What happens internally ?**

**WC -- discards resp buffer.**

**sends temp redirect response to the clnt**

**SC 302 | location : s2 .... | Body : EMPTY**

clnt browser --> sends a NEW request

URL : http://host:port/day2/s2 ,method = GET

Meaning : you are navigating the clnt to the next page , in the NEXT request.

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)

IMPORTANT :

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

2. Server Pull.

Min scope reqd for sharing data is Request scope

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.

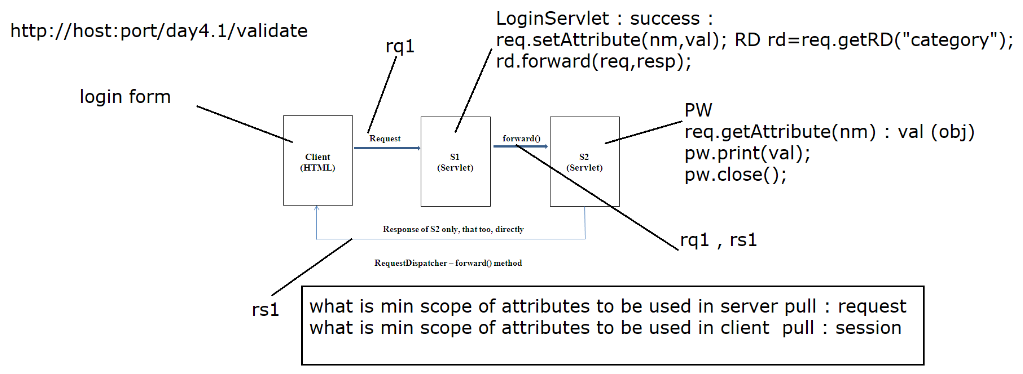
Steps

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)

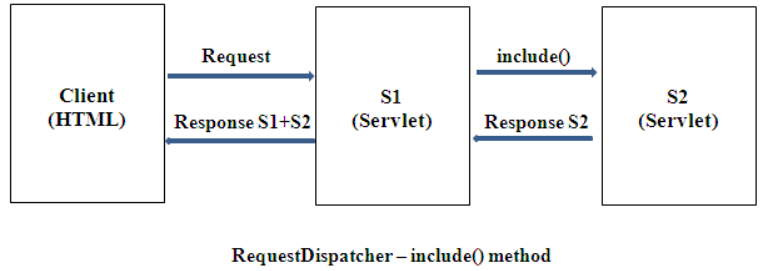
This method allows one servlet to do initial processing of a request and another resource to generate the response. (i.e division of responsibility)

**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

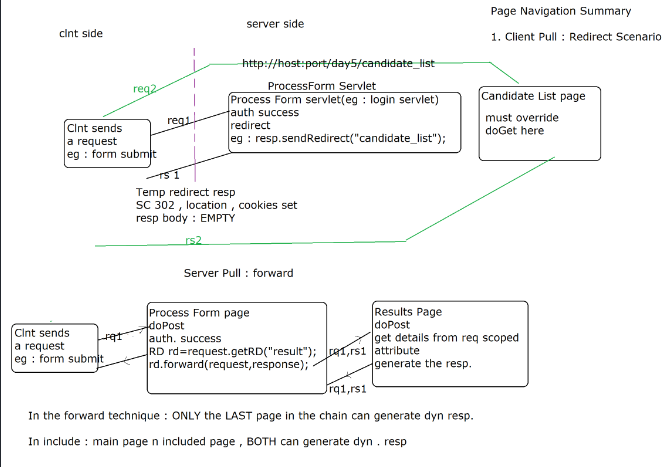


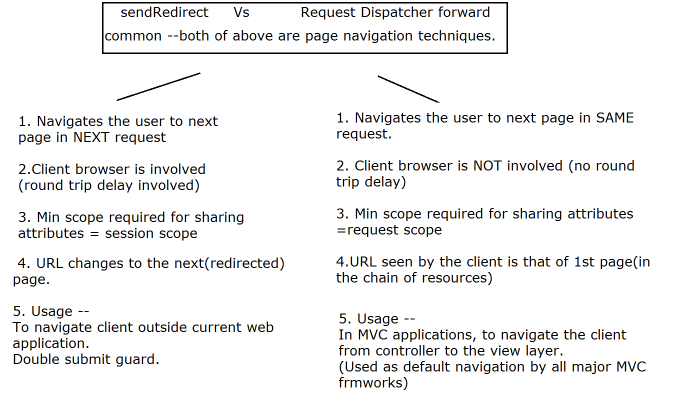
API of RequestDispatcher 🡪 public void **include**(ServletRequest rq,ServletResponse rs)

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.**

**Summary:**

****

****

**Revise server pull**

eg : In servlet1 (/s1 : doPost) ---> servlet2(/s2) : forward

RD rd=request.getRD("s2");

rd.forward(request,resp);

**WC : clrs resp buffer,suspends exec of s1 --> Servlet2's doPost --can generate dyn resp --s2' doPost rets--> control comes back --s1 (CAN'T genearte dyn resp) --s1's doPost rets ---resp content**

SC 200 | headers ..| body : non empty (resp from last page in the chain)

Use case : In MVC based web app frmworks (JSF,Struts,Spring MVC,Apache Camel)

RD's forward tech. is used to forward clnt from Controller --> View layer

RD's include

In servlet1 (/s1 : doPost) ---> servlet2(/s2) : forward

RD rd=request.getRD("s2");

rd.include(request,resp);

**WC : retains resp buffer,suspends exec of s1 --> Servlet2's doPost --can generate dyn resp --s2' doPost rets--> control comes back --s1 (can continue to genearte dyn resp) --s1's doPost rets ---resp content(s1+s2)**

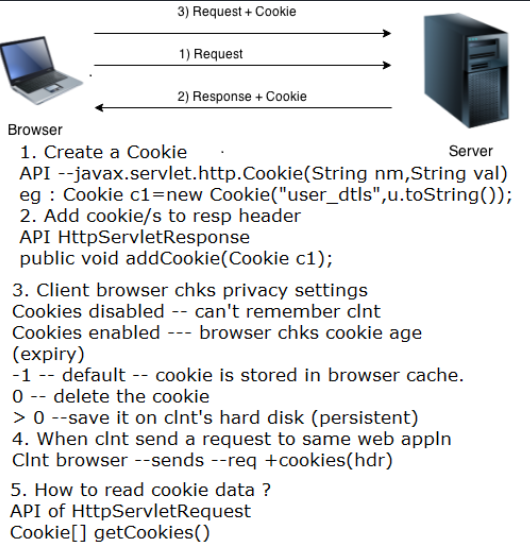
SC 200 | headers ..| body : non empty (resp from s1+s2)

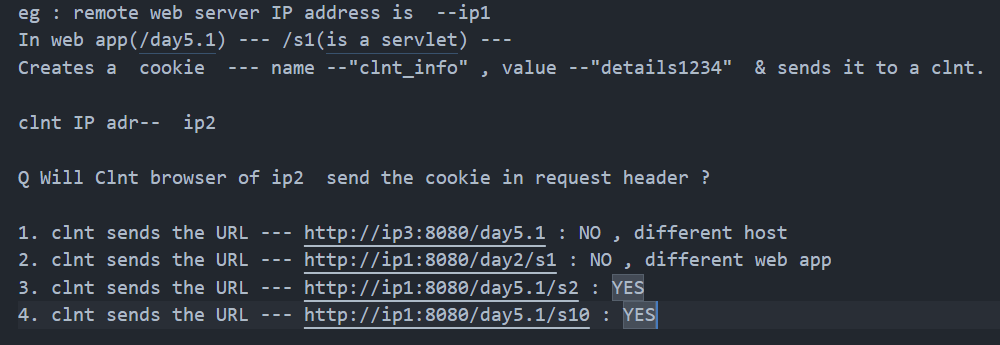
**Session Tracking**

There are several techniques for session tracking.

J2EE specific techniques :

1. Plain Cookie based scenario





**cookies are by default sent to the SAME web app of origin(i.e the web site which has generated cookies)**

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.

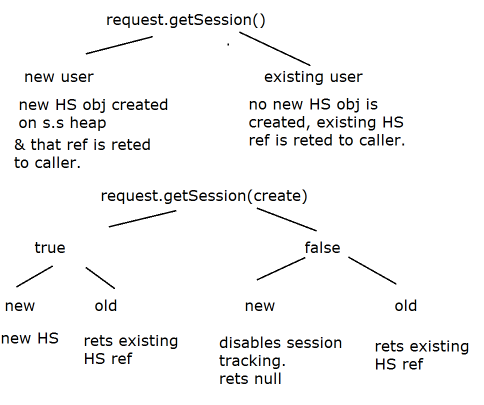
2. As no of cookies inc., it will result into increased net traffic.

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.

2. HttpSession interface

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)



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)

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

BUT entire session tracking again fails , if cookies are disabled.

3. HttpSession + URL rewriting

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 information.

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=fjghsjfkhgk12345gjgh

For URLs generated by sendRedirect : clnt pull II : use

HttpServletResponse method

public String encodeRedirectURL(String redirectURL)

Rets : redirectURL;JSESSIONID=12345

**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>

OR

@WebServlet(value="/test", initParams={@WebInitParam(name="nm1",value="val1"),@WebInitParam(name="nm2",value="val2")})

public class MyServlet extends HttpServlet {....}

6. How to access servlet specific init params from a servlet ?

6.1 Override init() method

6.2 Get ServletConfig

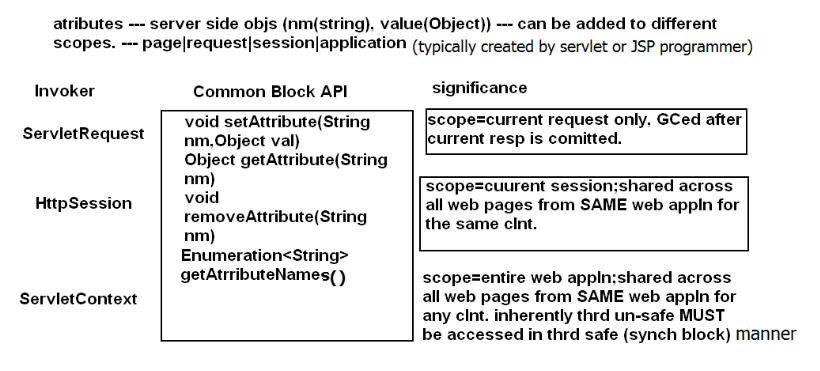
Method of Servlet i/f

public ServletConfig getServletConfig()

6.3 Get the init params from ServletConfig

Method of ServletConfig i/f

String getInitparameter(String paramName) : rets the param value.



**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

**Executor Framework (a part of Java SE)**

Introduced in Java 5.

What's earlier support ? 🡪 Extends Thread , Implements Runnable

Executor Framework (a part of Java SE)

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 public static ExecutorService **newFixedThreadPool**(int maxNoOfThreads): 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

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>

**JSP (Java server page)**

It is a Dynamic Web page (having typically HTML 5 markup) , can embed Java code directly.

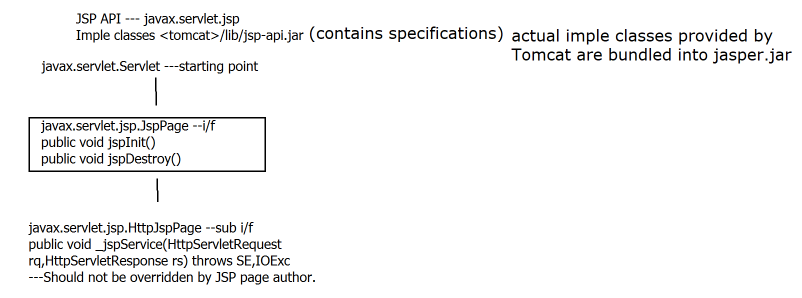
Dynamic web component , whose life-cycle is managed by **WC(JSP container/Servlet container/Servlet engine)**

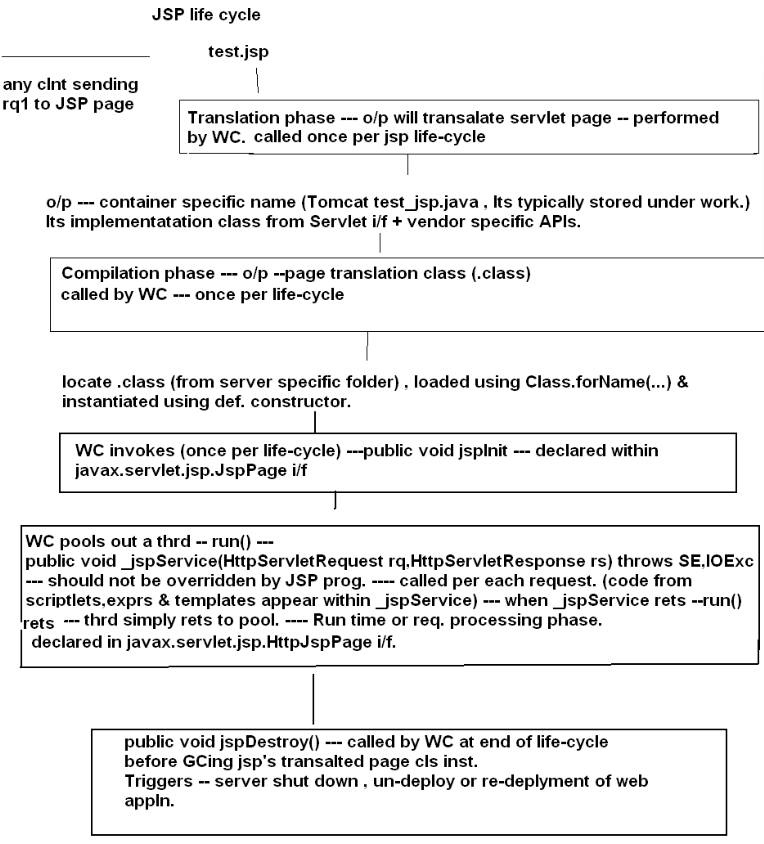
0. javax.servlet.Servlet -- super i/f

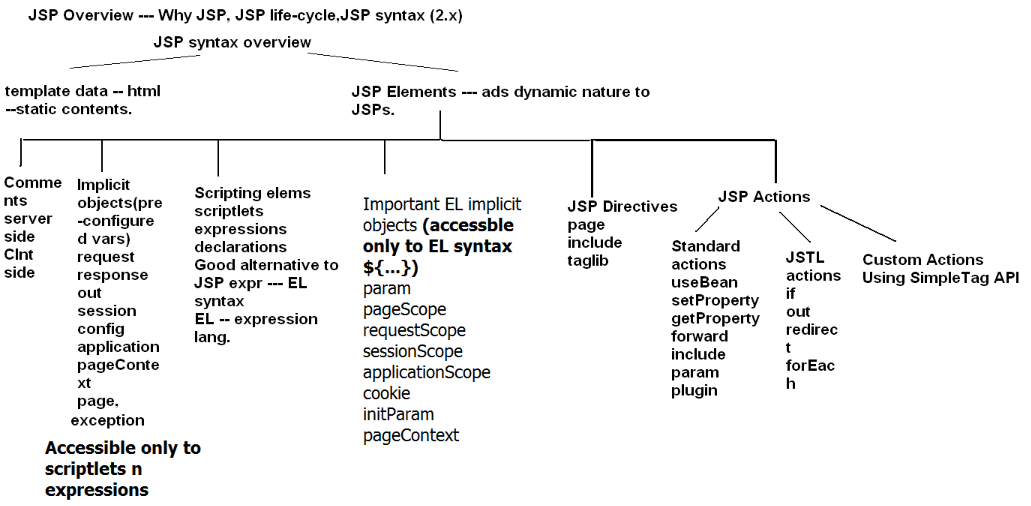
1. javax.servlet.jsp.JspPage -- extends Servlet i/f

1.1 public void jspInit()

1.2 public void jspDestroy()







Eg

1. ${sessionScope.team\_dtls}

sessionScope.get("team\_dtls") --> to string --> sent to clnt

OR

session.getAttribute("team\_dtls") --> to string --> sent to clnt

Suppose you have added a page scoped attribute , how ?

eg : one.jsp

<%

pageContext.setAttribute(nm,val);

%>

How to display ?

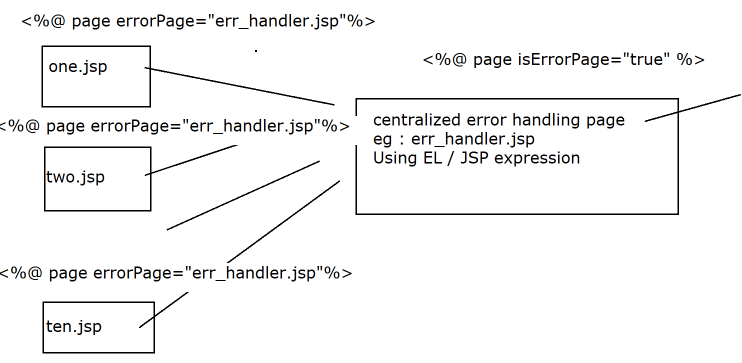
${pageContext.nm}

${pageScope.nm}

${page.nm}

${nm}

Options : 2 , 4



**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.

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

<%@ 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" %>

<c:set var="abc" value="${param.f1}" />

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

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

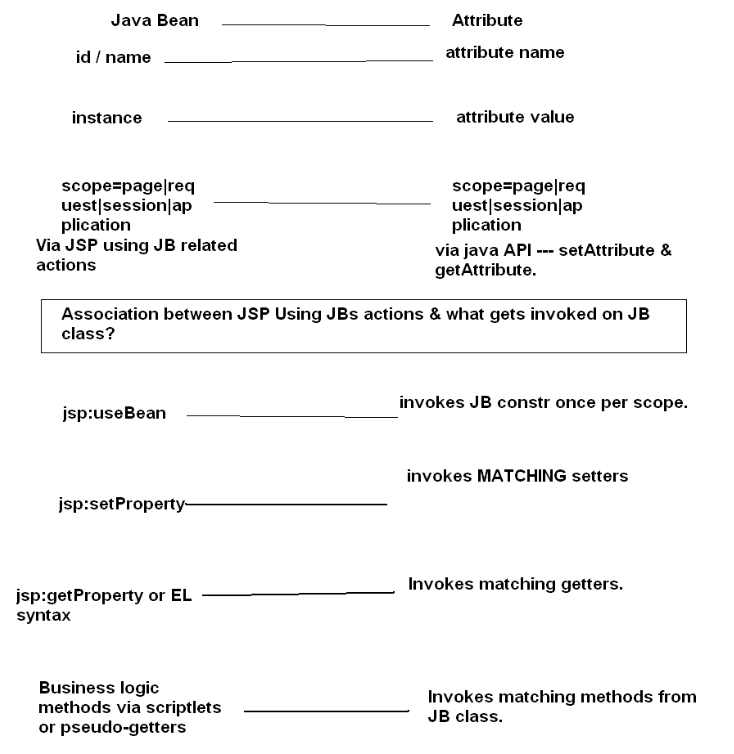
${cat}<br/>

</c:forEach>

WC invokes ---

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

out.print(cat);



**What is Hibernate ?**

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

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))

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.

**Read Hibernate\_Help.txt in day7**

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)

2. 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 modifications at any time without knowing about the state of an session then use merge() in hibernate.

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:

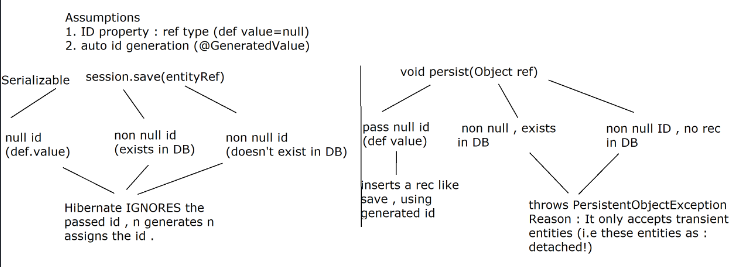
why its taken as detached ? ---non null id.

2.

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.



3. 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.

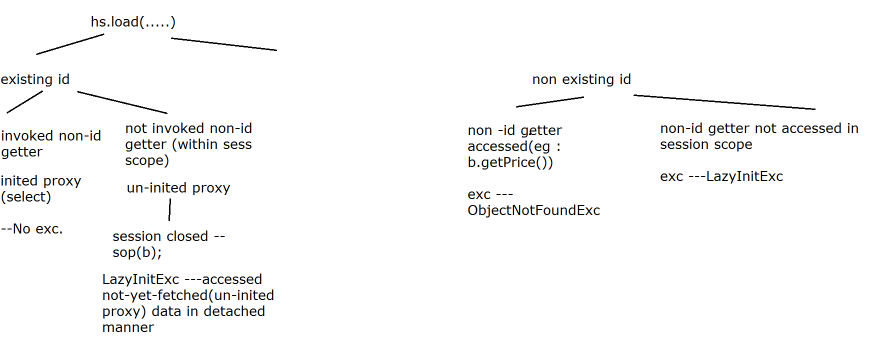
null id -- fires insert (works as save)

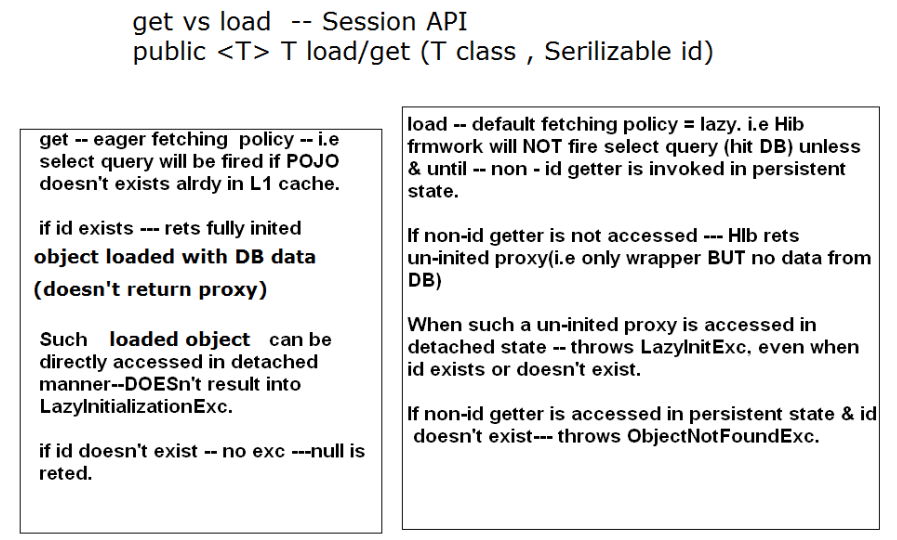
non-null BUT existing id -- fires update (select , update)

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

4. get vs load

& LazyInitilalizationException.





5. 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.

3.If there is no persistent instance currently associated with the session, it will be loaded.

4.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.

5. 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.

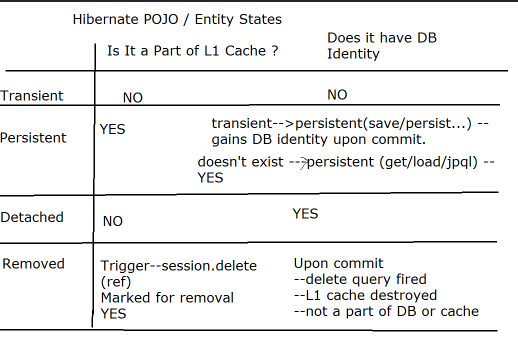
void refresh(Object ref) -- ref --persistent or detached

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

**Read day9 employee code**



ORM tool

JPA implementor

JPA : Java Persistence API --- Java EE / Jakarata EE specs (javax.persistence)

Hibernate : JPA implementor

(DB Journey in Java ---1. JDBC 2. Hibernate (native hibernate) 3. JPA 4. Spring Data JPA

POJO Annotations

Package : javax.persistence

@Entity : Mandatory : cls level

@Id : Mandatory : field level or property (getter) : PK

Optional annotation for further customization :

@Table(name="tbl\_name) : to specify table name n more

@GeneratedValue : to tell hibernate to auto generate ids

auto / identity(auto incr : Mysql) / table / sequence(oracle)

eg : @Id => PK

@GeneratedValue(strategy=GenarationType.IDENTITY) => auto increment

@Column(name,unique,nullable,insertable,updatable,length,columnDefinition="double(8,2)") : for specifying col details

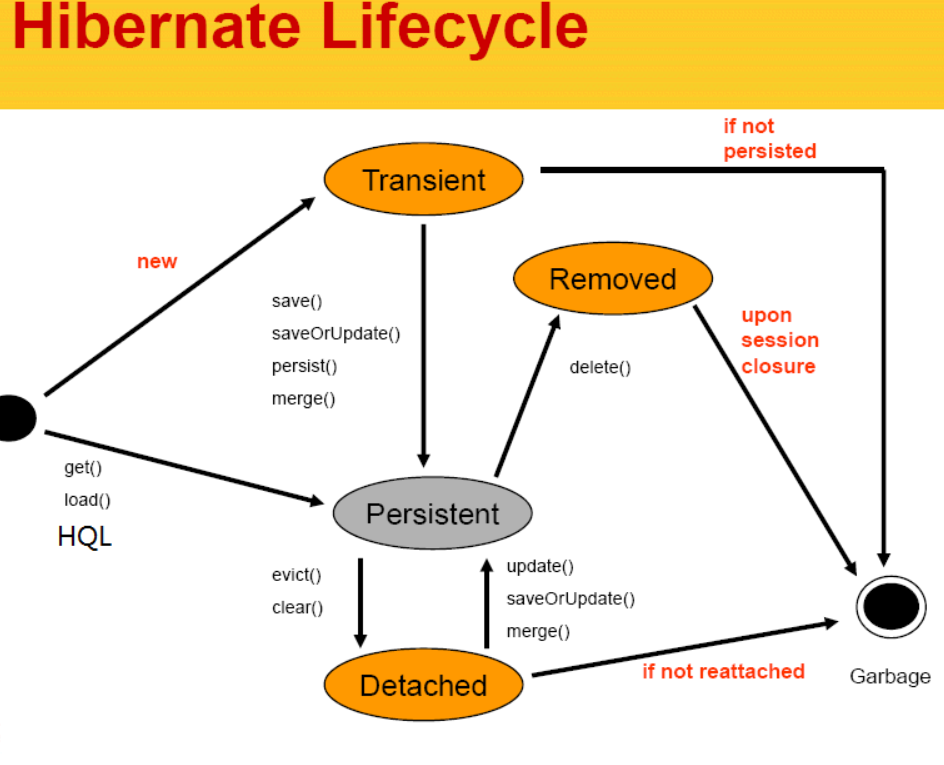
@Transient : Skipped from persistence(no col will be generated in DB table)

@Temporal : java.util.Date , Calendar , GregorianCalendar

LocalDate(date) ,LocalTime(time) , LocalDateTime (timestamp/datetime) : no temporal anno.

@Lob : BLOB(byte[]) n CLOB(char[]) : saving / restoring large bin /char data to/from DB

@Enumerated (EnumType.STRING): enum (def : ordinal : int)



Advanced Hibernate

1. The inverse side of a bidirectional relationship must refer to its owning side (i.e the Entity which contains the foreign key)

The mappedBy element refers to the property or field in the entity that is the owner of the relationship.

The many side of @ManyToOne bidirectional relationships must not define the mappedBy element. The many side is always the owning side of the relationship.

For @OneToOne bidirectional relationships, the owning side corresponds to the side that contains @JoinColumn i.e the corresponding foreign key.

For @ManyToMany bidirectional relationships, either side may be the owning side.

Bi directional association example for Deaprtment 1 <------> \* Employee

Department -- parent side , non owning (inverse) side

Employee --(students) --child side , owning side (containing FK)

NOTE : A parent is the table that stores the primary key, A child is any table that references the parent with a foreign key. These tables are named as parents and children because the child inherits values from the parent

In current case , add

@OneToMany in Department side

@ManyToOne in Employee side

2. Next Problem : Additional table is created (which is un necessary)

Why ? -- In a bi-dir association , you must supply owning side n inverse side information to hibernate.

How to supply ? --Add mappedBy attribute in @OneToMany

Where to add -- in the inverse side.(eg : Department POJO)

What should be value of mappedBy attribute?

Name of the association property , as it appears in the owning side , i.e Employee side

Cascading refers to the ability to automatically propagate the state of an entity across associations between entities.

Cascading in Hibernate refers to the automatic persistence of related entities.

java.persistence.CascadeType : enum

Values : ALL,PERSIST, MERGE, REMOVE, REFRESH, DETACH

Solution : Add a cascade type.

eg : @OneToMany (mappedBy = "dept",cascade = CascadeType.ALL)

private List<Employee> emps=new ArrayList<>();

NOTE : IN a bi-dir association , you have to establish both sides of the association.

Solution : Apply this solution in POJO level

As per Gavin King's suggestion , add helper methods.

addChild n removeChild

eg : addEmployee n removeEmployee

orphanRemoval : a property of @OneToMany set to true in Departments

JPA/Hibernate follows default fetching policies for different types of associations

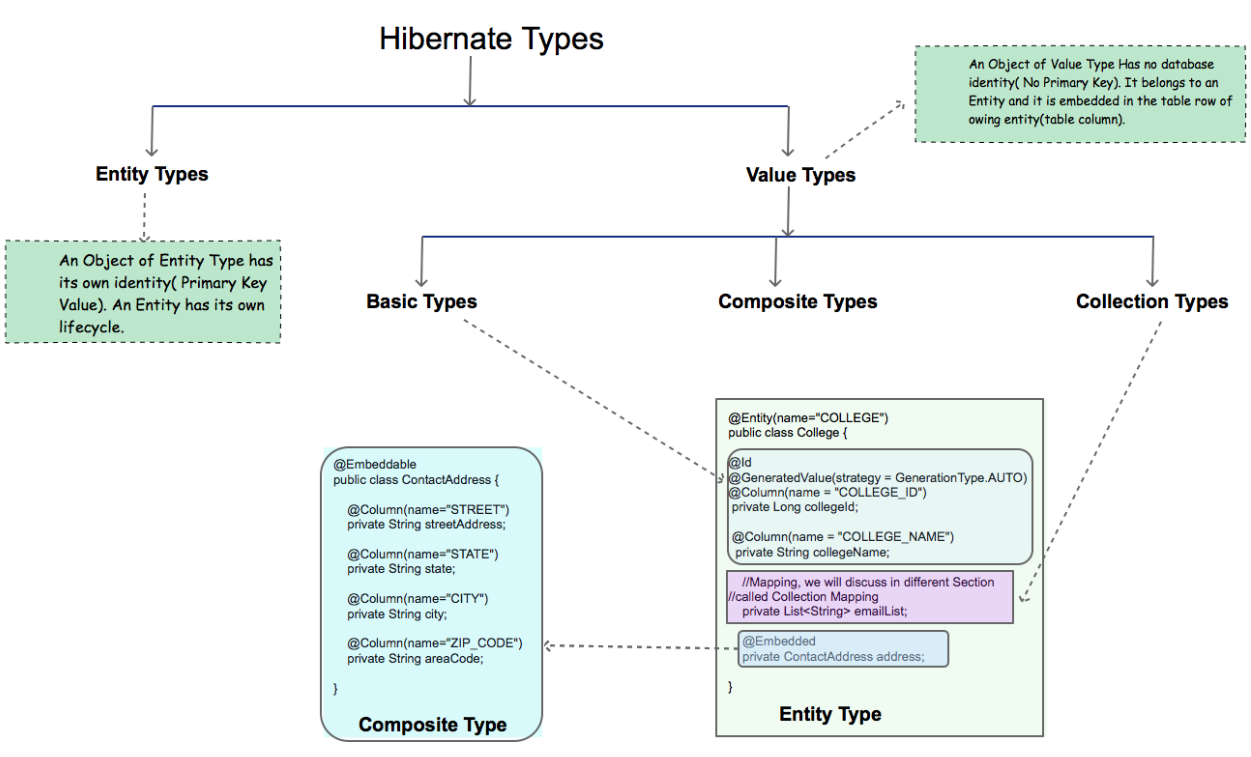
one-to-one : EAGER

one-to-many : LAZY

many-to-one : EAGER

many-to-many : LAZY

The best way to map a @OneToOne relationship is to use @MapsId.



What is Maven ?

It is a popular build automation and dependency management tool used mainly in Java projects.

By default,

java source code resides in the src/main/java directory

resources (config xml files) in the src/main/resources directory.

test cases' code in the src/test/java directory

for test cases , config files : src/test/resource

POM (Project Object Model):

The pom.xml file is the heart of the Maven project.

It defines the project's metadata, dependencies, build settings, and more.

The POM contains information such as the project's group ID, artifact ID, version, and packaging type.

It also specifies the project's dependencies, plugins, repositories, and the build process.

Spring

Why Spring : loosely coupled application

Via : D.I (Dependency Injection)

What is D.I ?(Dependency injection=wiring=collaboration between dependent & dependency)

Instead of dependent objs managing their dependencies , 3rd party containers(eg : Angular / Spring/ EJB/ WC) will auto create the dependecies & make it available to dependents, directly @ run time.

Since dependent are no longer managing dependencies --its called as IoC ---Inversion of control

Hollywood principle --You don't call us , we will call you....

SC --- > Dependent objs (i.e SC will create the dependencies for the dependent objs)

eg :

In DAO layer

@AutoWired

private SessionFactory sf;

prototype : => SC creates a separate instance of the bean : as per demand i.e getBean)

lazy-init : def value =false (applicable to ONLY singleton beans)

init-method : name of custom init method

pattern : public void anyName() throws Exc {....}

Will be invoked for singleton as well as prototype beans

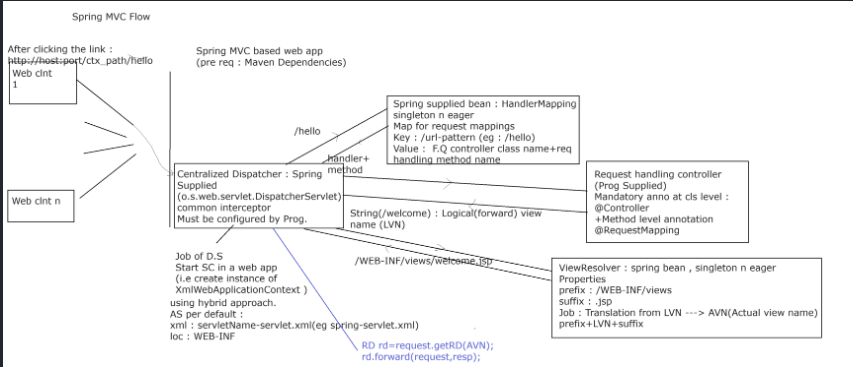
destroy-method : name of custom destroy method : SC invokes it just before GC

pattern : public void anyName() throws Exc {....}

Will be invoked only for singleton beans

factory-method : to supply the name of the factory method , in factory method based DI

MVC



<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

Meaning : To Configure spring supplied Front controller(o.s.w.s.DispatcherServlet) to intercept any request from any client, in web.xml.

Detailed explanation of tags

<url-pattern>/</url-pattern> => any request received from any client

<servlet-name>spring</servlet-name> => can be replaced by any name

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class> => spring supplied Front Controller

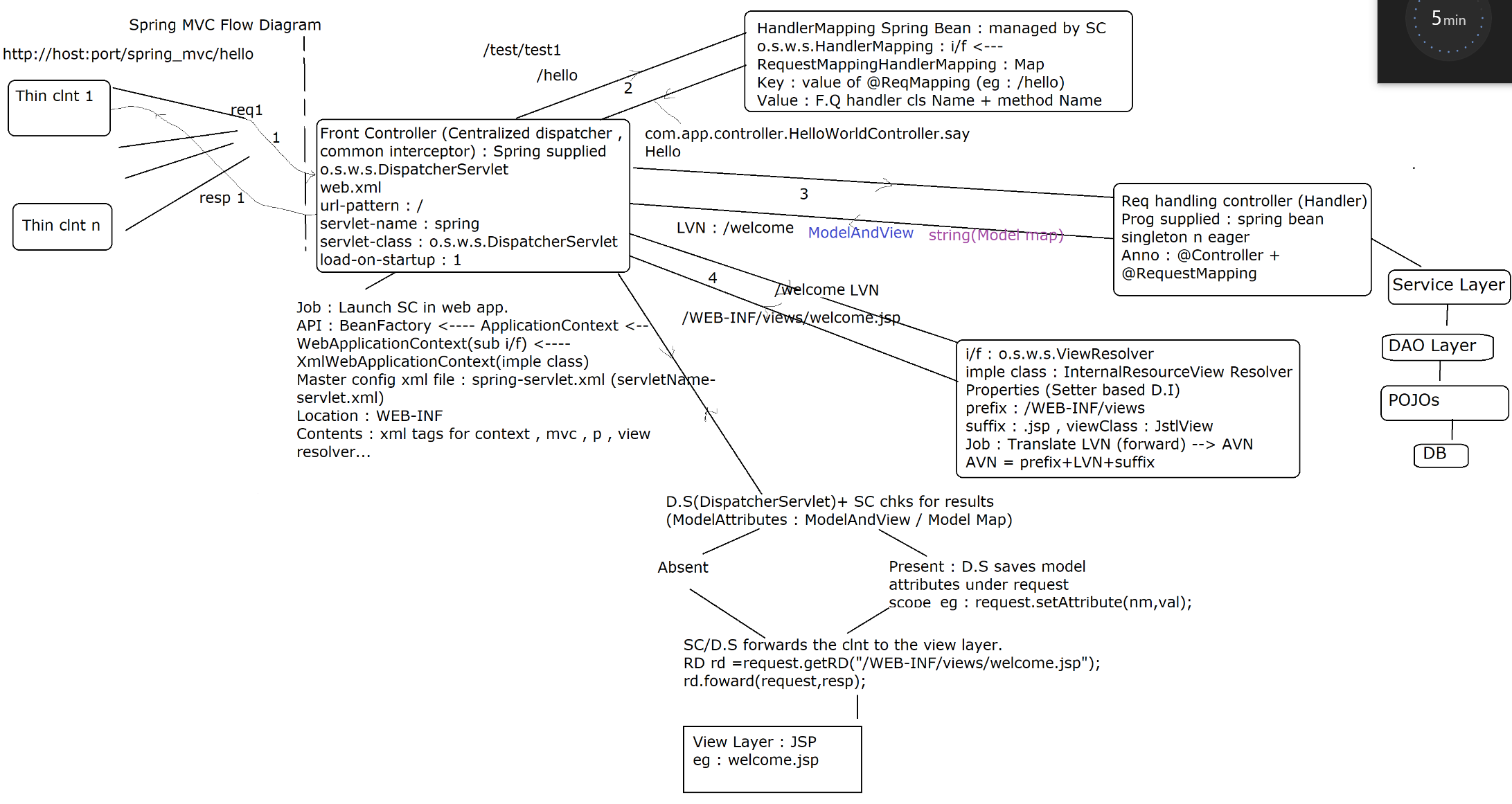
<load-on-startup>1</load-on-startup> => Web container (WC) will start the life cycle of DispatcherServlet , at the web app deployment time.

Job of D.S (DispatcherServlet)

To start the SC (spring container) in web app.

spring-servlet.xml => master configuration xml file for starting SC.

D.S(DispatcherServlet) reads this config file , @ web app deployment time , to start SC , within a web app. (represented by i/f WebApplicationContext --sub i/f of ApplicationContext)



# Paper cdac acts

1. How to deploy a servlet context listener within web application ?

1 point

A: Using <listener> tag in web.xml

B: Using <web-listener> tag

C: Using @WebListener annotation

D: A & C

2. How many ServletConfig & ServletContext objects will be created by web container?

1 point

A Only single one per entire web server.

B One per servlet & one per web application

C. One per context & one per application

D One per every servlet.

3. Consider following 2 servlets.

@WebServlet(value="/s1",loadOnStartup=1)

public class FirstServlet extends HttpServlet {...}

@WebServlet(value="/s2")

public class SecondServlet extends HttpServlet {...}

Which of the following is true ?

1 point

A. Web container doesn't invoke init method of the second servlet.

B Web container invokes init method of the second servlet at web app start up.

C. Web container invokes init method of the first servlet at web app start up.

D. Web Container starts life cycle of the first servlet , after clint request.

4. Which of the following attributes are inherently thread un safe ?

1 point

A. Page Scoped attributes

B. Request scoped attributes

C. Session scoped attributes

D. Application scoped attributes

5. What will be output , if client sends the following request to test1.jsp ?

http://host:port/test\_web/test1.jsp?pid=101&type=electronic&name=tv

test1.jsp

<body>

Product Details are : <br>

<jsp:forward page="test2.jsp"/>

</body>

test2.jsp

<body>

Name : ${param.name} <br>

Type : ${requestScope.type}

</body>

1 point

A. Product Details are : Name : tvType : electronic

B. Name : tv Type : electronic

C. Product Details are :

D. Name : tv Type :

6. While using request chaining scenario in page navigation , which should be minimum scope of server side attributes?

1 point

A: page scope

B: request scope

C: session scope

D: application scope.

7. Which of the following is not accessible via EL syntax

1 point

A. pageContext

B. sessionScope

C. param

D. session

8. Consider following JSPs. What will be displayed on client browser?

test1.jsp

<%

request.setAttribute("age",25);

session.attribute("height","160cm");

%>

<c:redirect url="test2.jsp"/>

test2.jsp

Age : ${requestScope.age}

Height : ${sessionScope.height}

1 point

A. Blank Page

B. Age : 25 Height : 160cm

C. Age : Height : 160cm

D. Age :25 Height :

9. Which of the following is true ?

1 point

A: In request dispatcher's scenario , minimum scope of attributes must be session scope.

B: In redirect scenario from one.jsp to two.jsp , client browser will see the URL of two.jsp.

C: In request dispatcher scenario from one.jsp to two.jsp , client browser will see the URL of two.jsp.

D: Request dispatching technique will be slower than send redirect.

10. Assume TestBean is a valid java bean class , under beans package.

Consider following code.

index.jsp

<jsp:useBean id="test" class="beans.TestBean" scope="application"/>

<c:redirect url="login.jsp"/>

login.jsp

<jsp:useBean id="test" class="beans.TestBean" scope="application"/>

How many times TestBean will be instantiated, if there are 10 clients accessing index.jsp of this web site ?

1 point

A.10 times.

B. 20 times.

C. 1 time

D. 2 times.

11. In Front Controller based MVC , what is the role of front controller?

1 point

A.Business Logic

B. Presentation Logic

C Centralized dispatcher

D UI

12. What will happen in the following hibernate code, if user id is annotated with @Id & @GeneratedValue?

User u=new User("Ravi","Seth","ravi@gmail.com");

u.setId(10);

//session : org.hibernate.Session

session.persist(u);

//tx : org.hibernate.Transaction

tx.commit();

1 point

A. A new record will be saved with user id=10;

B. Throws PersistentObjectException

C. A new record will be inserted using auto incremented id.

D. No change in DB.

13 . In a bi directional one-to-many association , what will happen if mappedBy attribute is not specified?

1 point

A. Hibernate will not create the tables automatically.

B. Hibernate will throw MappingException

C. Hibernate will create additional table , having PKs of both the tables.

D. None of the above.

14. What will happen for the following code , when you bootstrap hibernate?

@Entity

public class Stock

{

@Id

private Integer id;

@Column(unique=true)

private String code;

@Transient

private String stockDesc;

//all getters n setters are provided.

}

1 point

A. Table creation will fail.

B SessionFactory creation will fail

C. Table will be created with only id & code columns.

D. Table will be created with id ,code & stockDesc columns

15. In many-to-many relationship between Book & Author,

Consider JPQL = "select a from Author a where a.name=:name";

Suppose session was opened using getCurrentSession , query is executed, author object is returned in a1 & transaction is committed.

Afterwards , if you access System.out.println(a1.getBooks()) , what will happen?

1 point

A. select query will be fired on author & books table

B. Hibernate throws LazyInitializationException

C. Will display the book details by a specified author.

D. Will only display author details.

16. Which of the following is true?

1 point

A. L2 cache is implicitly enabled in hibernate.

B. Query level cache is implicitly enabled in hibernate.

C L1 cache is implicitly enabled in hibernate.

D. All of above

17. What can be said regarding saveOrUpdate method of hibernate?

1 point

A. It can cause a transition from transient to persistent entity.

B. It cause a transition from detached to persistent entity

C A & B

D It causes a transition from persistent to detached entity.

18. What will be happen in the following hibernate code sample, if account a with id=100 exists in DB , with balance=5000?

Session hs=factory.getCurrentSession();

BankAccount a=hs.get(BankAccount.class,100);

a.setBalance(7000);

tx.commit();

a.setBalance(8000);

1 point

A. Account balance will still be 5000

B. balance will be 7000

C. balance will be 8000

D. Runtime exception is thrown

19. Which default fetching policy JPA uses for one-to-one & one-to-many?

1 point

A. Eager & Lazy

B. Lazy & eager

C Eager & eager

D. Lazy & Lazy

20. In spring hibernate based web application , what can you say regarding Session's load method ?

1 point

A: Returns a proxy by default

B: Returns null , if entity is not found

C: load method is un supported

D: load method will always use eager fetching policy.

2 फेब्रुवरी (शुक्रवार)

तिरूपति ते बंगलोर **(टर्मिनल -1)**

सकाळी **10:50 ते 11:50**

फ्लाइट नंबर 🡪 **Indigo 6E-7738**

PNR नं 🡪 **EV68MD**

सीट नं 🡪 **4A, 4C, 4D, 4F, 5A**

बंगलोर **(टर्मिनल -1)** ते औरंगाबाद

दुपारी **02:30 ते 04:00**

फ्लाइट नंबर 🡪 **Indigo 6E-516**

PNR नं **🡪 EV68MD**

सीट नं 🡪 **14A, 14B, 14C, 15A, 15B**