Name:- Zaveri Rishit N.
Sub:- Enterprise Java
Roll No.:- 75 semester:- 8th

1. Java EE servlet & JSP are the heart Java EE. All the Java EE frameworks we built top of servlets & JSPs. A comprensive post with more than 20 tutorials to help you learn servlets & JSP.

=> new in java EE 6:
- managed Bean
- POJO based managed component
- Provides common set of service such as lifecycle resource, injection callback & interception
- Enterprise JavaBeans.
- An EJB can be packaged in for local access using @Local & ejb-jar for local & remote access
- EJB can be accessed using a Protol' global JNDI named servlet
- Annotations driven servlet and listeners. The web.xml descriptor became optional in a most of the common case frameworks libraries can be integrated in a modular way using web.xml.
- servlet filter & listener can be programmatically
- Restful web service
- client-side content negotiation support using HTTP Accept headers.

=> Java EE 7:-
- Java API for web socket
- Enables a websocket client & server endpoint to be defined declaratively via annotations on a POJO.

or programmatically via interface implementations.
- offers client-specific configurations such as providing custom configuration algorithms.
- Allow for integrations with existing Java EE technologies.
- Java API for JSON processing.

=> New Java EE 8:
-) JSON-B providing a binding layer for converting java objects to & from JSON message.
- Updates to JSON-P, improving the object-model
- Java EE security API, supporting cloud
- HTTP/2 support in java servlet, servlet provides a programming class to extend server capabilities.
- JAX-RS 2.1 section client API.
- ~~EE~~ JSF 2.3 for building server side user interface.

2. (i) stateless session Bean:-
-) This is business object that represent business logic only. It doesn't have state.
-) The stateless bean object use pooled by the EJB container to service that request on demand.
- It can be access by one client at a time.
- In case of concurrent access, EJB container routes each request in a different instance.
- Annotation used in stateless bean.
(i) stateless

(ii) PostConstruct (iii) Predestroy

- EJB container creates & maintains a pool of session Bean first. It injects the dependency if then calls the @PostConstruct method if any
- now Actual business logic method is invoked by the client then container calls @predestroy method. if any now bean ready for garbage collection

eg.:- @Remote
Public interface DemoR
{
int add (int a, int b); }

-> Demo.java:-
@stateless (mappedName: "st1")
Public class Demo implements DemoR
{
Public int add (int a, int b)
{ return a+b; } }

-> Demo.java
Public class test{
public static void main (String[] ar) {
Context c = new InitialContext();
DemoR r = (DemoR) c.lookup ("st1");
System.out.print (r.add (2, 3));
} }

(ii) statefull session Bean:- This is business object that represent business logic like stateless session bean. but it maintain state.

- annotation used in this bean.
@ stateful @ PostConstruct @ Predestroy
@ prepassivate @ PostActivate.

eg.

-> BankR.java :-

@Remote

```
Public interface BankR {
    boolean withdraw(int amt);
    void deposit(int amt);
    int getBalance();
}
```

- Bank.java :-

@Stateful(mappedName="stateful")

```
Public class Bank implements BankR {
    Private int amt = 0;
Public Boolean withdraw(int amt)
    {
    if (amt <= this.amt)
    { this.amt = amt;
        return true; }
    }else {
    return false; } }
Public void deposit(int amt) {
    this.amt += amt; }
Public int getBalance() {
    return amt; } }
```

=> index.jsp :-

```
<a href = "openAC"> OpenAccount </a>
```

=> file operation.jsp :-

```
<form action = "operationProcess.jsp" >
```

```
Amount <input type="text" name="amt" >
operation : <input type="radio" name="rd"
            value="dep" > Deposite
<input type="radio" name="rd" value="wit">withdraw

Check Balance : <input type="radio" name="rd"
                value="chk" >
<input type="submit" value="submit" >
    </form>
=> operationprocess.jsp
<%.
Bank R re=session.getAttribute("Remote");
  String ope = req.getParameter("rd");
String amt = request.getparameter("amt");
    if (ope!=null)
    {
if (ope.equals("dep"))
{ remote.deposite(Integer.parseInt(amt));
  else
{ if (operation.equals("wit"))
{
boolean flag = remote.withdraw(Integer.parseInt
                                (remote));
if (flag) {
out.print("Amount withdraw");
} else {
  out.print("enter less amount");
  }} else {
  out.print (remote.getBalance());
  }}  %>
```

<jsp:include page="operation.jsp" />

-> OpenAccount.java :-

```
@webservice("/openAccount");
public class openAccount extend HttpServlet {
Protected void doGet(HttpServletRequest req,
HttpServlet Responce Res) {
    try {
initialContext c = new InitialContext();
Bank b = c.lookup("stateful");
    req.getSessionc().setAttribute("remot", b);
    rea.getRequestDispatcher("/operation.jsp");
    include (req, res);
    } catch (Exception e) {     }
    }}
```

3. SOAP	REST
- SOAP is a protocol	- REST is a architecture.
- SOAP stands for simple Object access protocol	-> It is stands for Representational state-transfer.
- SOAP continues REST because it's protocol	-> REST can use SOAP based service b'z it's conceptual & can use any protocol like (HTTP, SOAP.
- SOAP defines standards to be strickly followed.	- REST does not define too much standard like SOAP.
- SOAP defines it's own secur	- RESTful service inherits secur

	measure from the undeploying transfer.
- SOAP requires more bandwidth & resource than REST	- REST requires less bandwidth & resource than SOAP.
- SOAP permits xml data format only.	- REST Permits different data format such as Plain text, html, XML, JSON etc.
- JAX-WS is the java API for SOAP web service.	-> JAX-RS is the java API for restful web service.

6. OAuth 2.0 is set of defined process flow for delegated authorization.
- OpenId connect is a set of defined process flow for federated authentication. OpenId Connects flow are built using the OAuth 2.0 Process flow as the basic & there adding a few additional step over it to flow for federates authentication.
:> OAuth 2.0 :- Consider the resource owner since java owns the resource. The server on which the resource Resides is called 'REST' Server. The app that is trying to access its resources on the resource serve is called 'client'. The server that authorize to the resource is called the 'authorization' server.

Resource Joe's contactlist
Resource owner - Joe
client - Yelp app
Resource server - google. contact server
Thus joe is 'delegating' the responcibility to authorize access to his resource hosted the resource server in the authorization server.
-> The oAuth 2.0 Process flow:- The following are the two most commonly used authorization code flow most common used Process flow.
- Implicit flow used in pure JSonApplication.

=> OpenId connect Process flow:-
The Process flow is the same as oAuth 2.0 authorization Process flow with following addition
-> In addition to the access token, an Id token is returned by the authorization server

- UserInfo and Point for getting more user informator.
- openId connect is built on the process flow of oAuth 2.0 typically uses JWT formet for the Id-token.