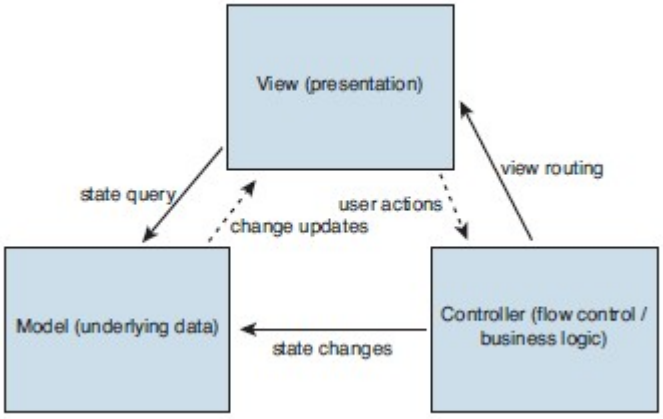
LIYANA ARACHCHIGE RANIL

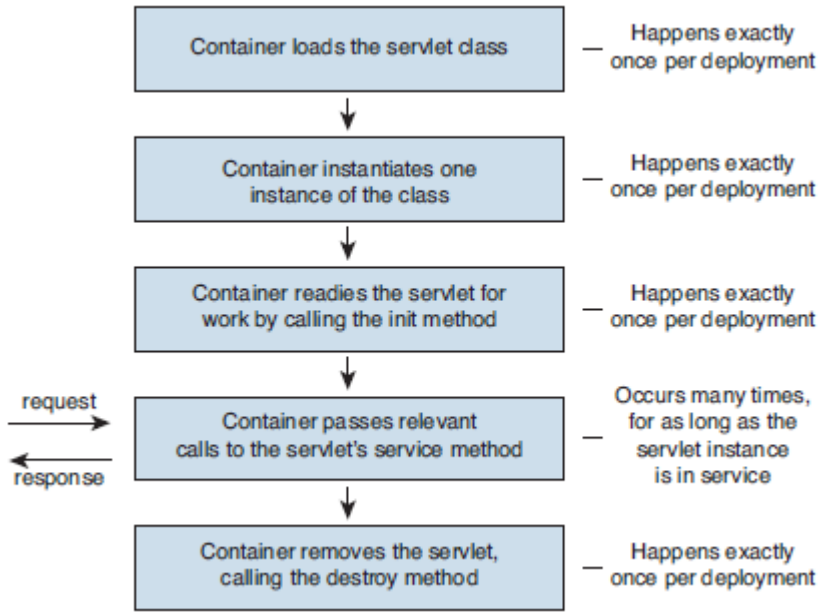
SHORT NOTES / HUMPHY SHEIL / MARK CADE – WEB TECHNOLOGIES

1. **WEB TECHNOLOGEIS** cover the area of **PRESENTATION LAYER**
2. Model View Controller (MVC) architecture is de-facto architecture by which **PRESENTATION** ,**BUSINESS LOGIC** and **DATA** are separated
3. 

4. MODEL = represents business data along with business logic or operations, VIEW = renders the content of a MODEL , CONTROLLER = defines application behavior
5. The WEB CONTAINER is ANALOGOUS to EJB CONTAINER , but provide support in WEB TIER
6. **CONTAINERS** are basically **APPLICATION INFRASTRUCTURE** and it relieves developers from lot of **INFRASTRUCURE** related issues
7. CONTAINERS usually provide
    a. CONSURRENCY CONTROL
    b. USER MANAGED TRANSACTIONS
    c. CONFIGURATIONS
    d. SECURITY MANAGEMENT

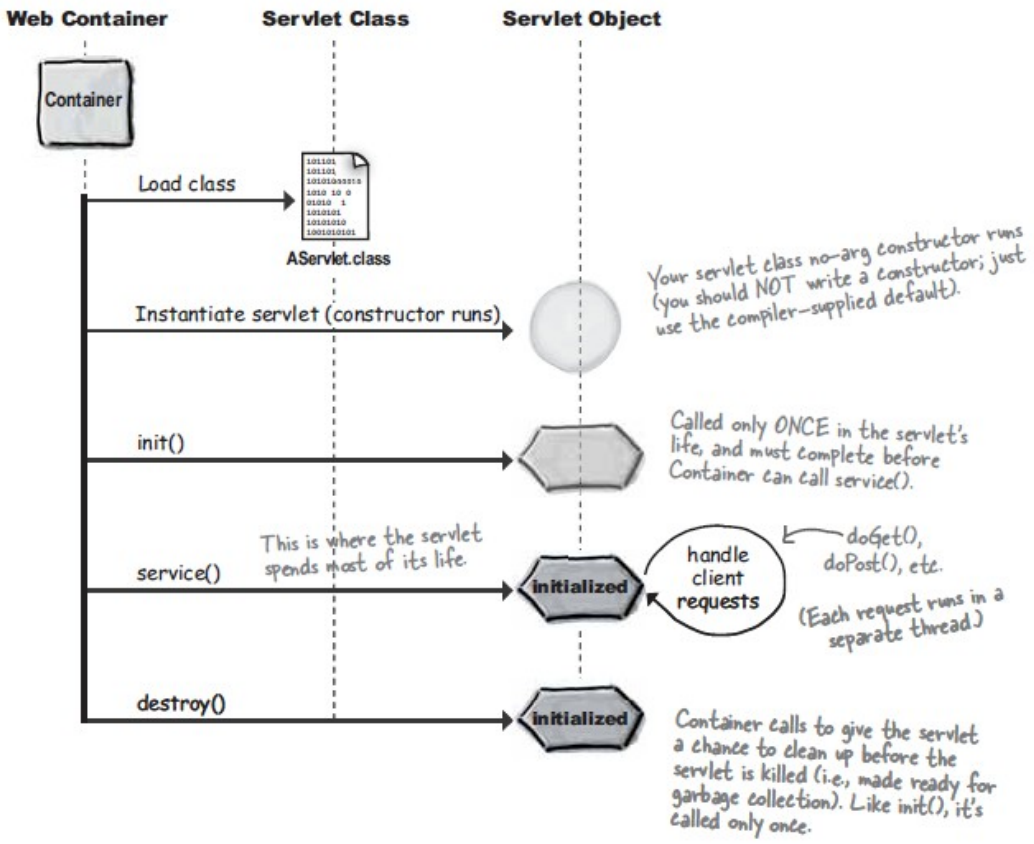
e. Etc…

8. SERVLET is a server-side component designed to handled inbound service requests from remote clients
9. SERVLET model is designed to accommodate any protocol that is predicated around REQUEST/RESPONSE model
10. A SERVLET is HOSTED by the CONTAINER and multiple threads use it in order to provide a SCALABLE system unless developer explicitly chooses not to do so by using SINGLETHREADMODEL (Deprecated)tagging interface
11. **SingleThredModel** tagging interface is deprecated since it results systems that **DO NOT SCALE**
12. SERVLET LIFE CYCLE



13.

14.



15. if a SERVLET calls response.sendRedirect(urlstring) , then the BROWER who would do the JOB. But id RequestDispatcher.forward('somefile') is used then the SERVER will do the work

16. When the initialized Servlet Code is given the SERVLETCONTEXT while invoking INIT method , then only the SERVLET is a true servlet
17. Servlet INIT parameters are READ ONLY once the server is started.
18.

```
<servlet>
    <servlet-name>BeerParamTests</servlet-name>
    <servlet-class>TestInitParams</servlet-class>

    <init-param>
      <param-name>adminEmail</param-name>
      <param-value>likewecare@wickedlysmart.com</param-value>
    </init-param>

</servlet>
```

You give it a param-name and a param-value. Simple. Just make sure it's INSIDE the <servlet> element in the DD.

19.

```
out.println(getServletConfig().getInitParameter("adminEmail"));
```

20. SERVLET INIT parameters are only available to the SERVELT given, but CONTEXT init parameters are available to the WHOLE WEB APP
21.

```
<context-param>
    <param-name>adminEmail</param-name>
    <param-value>clientheaderror@wickedlysmart.com</param-value>
</context-param>
```

You give it a param-name

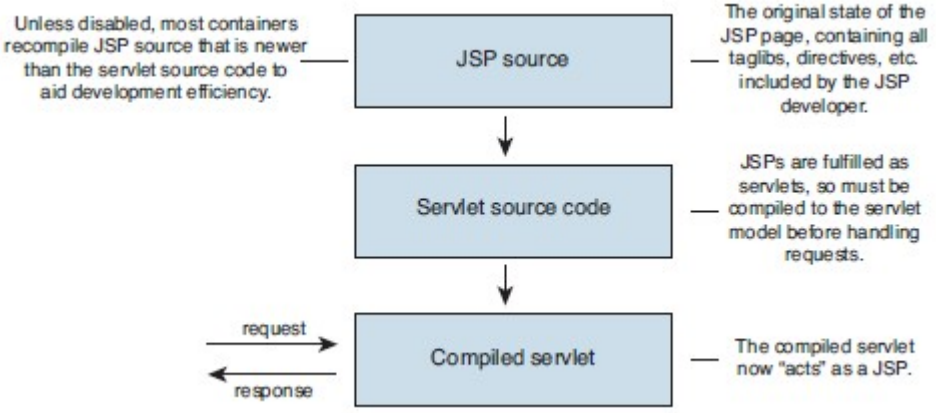
22.

```
out.println(getServletContext().getInitParameter("adminEmail"));
```

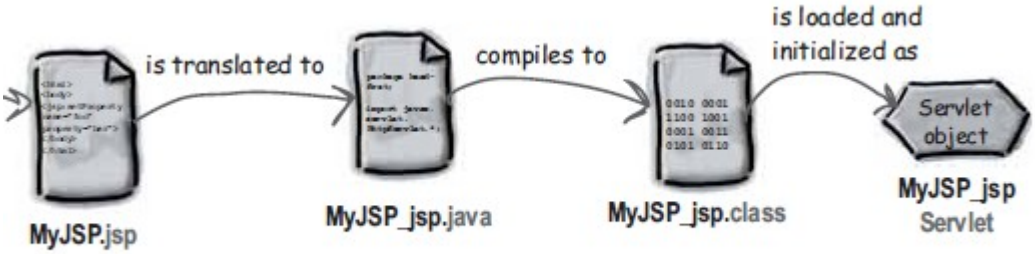
23. SERVLET CONFIG only ONE PER SERVLET , SERVLET CONTEXT one PER WEB APP

24. **ServeletContextListener interface** can be used to listen to CONTEXT initialization and context destruction. Add the context listener in web.xml under <listener><listener-class>class of listener</listener-class></listener>
25. There are other listener types , **HttpSessionListener , HttpRequestListener, HttpSessionAttributeListener,HttpRequestAttributeListener,HttpContextListener,HttpContextAttributeListener,HttpSessionBindingListener,HttpSessionActivationListener**
26. HttpSessionBindingListener helps ATTRIBUTE itself to get notified when it is being bound to a session or unbround from a session
27. In a SERVLET , only REQUEST ATTRIBUTE and LOCAL VARIABLES are THREAD SAFE
28. WEB CONTAINER keep track of USER SESSIONS with the help of COOKIES or URL REWRITING
29. FILTERS are server side components hosted by WEB CONTAINER that receive INBOUND request BEFORE it is received by any other COMPONENT
30. FILTERS are used to PRE-PROCESS requests (ex : LOG EVENTS , PERFORM SECURITY CHECKS etc..)
31. FILTERS can also be used to POST-PROCESS requests
32. LISTENERS are server side components hosted by the WEB CONTAINER that are NOTIFIED about specific events that OCCUR during a SERVLET'S LIFE CYCLE
33. JAVA SERVER PAGES (JSP) are HTML pages with embedded MARK-UP that is evaluated at runtime by the WEB CONTAINER to create complete HTML pages

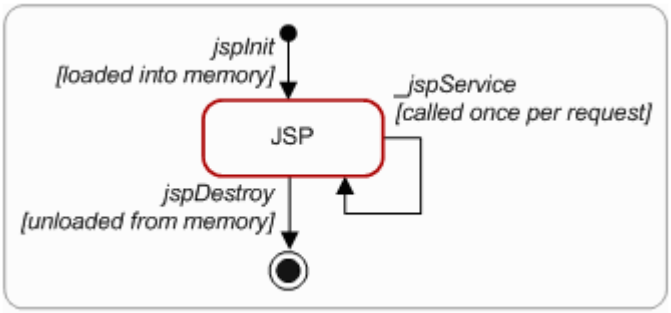
34. JSP LIFE CYCLE
35.



36. JSP is translated to JAVA code , which represent a SERVLET and then SERVLET code is compiled to class files and then this follows SERVLET life cycle
37.



38. TRNSLATION PHASE -> COMPILATION PHASE -> INSTANTIATION AND LOADING PHASE -> CALL **jspInit()** -> CALL **_jspService()** [more than once] -> Call **jspDestroy()**

39. 

Lifecycle of a JSP Document

40. In JSP's to import a package, packages write DIRECTIVES. Page DIRECTIVE is helpful in this case**, < %@page import="foo.*,java.*"%>**

Scriptlet:	<% %>
Directive:	<%@ %>
Expression:	<%= %>

41. Scriptlets , expressions ,directives
42. All SCRIPLETS , EXPRESSIONS land in _jspService method
43. All the variables declared in SCRIPLET are LOCAT variables in the SERVICE method
44. To declare CLASS LEVEL variables, use **<%! Int I = 0 ;%>**
45. There are several IMPLECIT objects that a JSP offers , they are
    a. JspWrite = out
    b. HttpServletRequest = request
    c. HttpServletResponse = response
    d. HttpSession = session
    e. ServletContext = application
    f. ServletConfig = config
    g. JspException = exception
    h. PageContext = pageContext
    i. Object = page

46. To make SCRIPTING invalid , add <scripting-invalid>true</scripting-invalid> tag to Deployment Descriptor (web.xml). This would essentially invalidate all the scripting in all the JSPs
47. EL (Expression Language) is ENABLED by DEFAULT, you can disable it by adding <el-ignored>true</el-ignored> in the DD
48. EL would look like ${someexpression} always
49. By using <%@ page isELIgnored=”true”%> , you can turn off /On EL for particular JSP page
50. EL implicit Object ,
    a. pageScope
    b. requestScope
    c. sessionScope
    d. applicationScope
    e. param
    f. paramValues
    g. header
    h. headerValues
    i. cookie
    j. initParam – MAP of CONTEXT INIT PARAMS
    k. pageContext – not a Map / all other are a MAP
51. EL can INVOKE FUNCTIONS as well, for that write a TLD , write a CLASS which has a public static method which you want to invoke , use the @page directive in JSp to get the TLD in to the page and use the function name in the EL block
52. Ex : ${mine:rollIt()} , mine=”tag lib prefix” , rollIt=”function name in TLD”

53. Include DIRECTIVE <@ include file=”Header.jsp”%> happens TRANSLATION TIME while <jsp:include page=”Header.jsp”/> happens at RUNTIME
54. JSTL is a set of TAG LIBRARIES that forms part of JSP specification
55. JSTL brought the MUCH NEEDED standardization to the tag library space
56. JSTL allows you to employ a single standard set of tags
57. JSTL SQL tags are supposed to be used whenever doing PROTOTYES and whenever overhead of creating BEAN is not possible
58. It is always better to ENCAPSULATE calls to a DATABASE in a BEAN
59. JSTL contains , **CORE , XML , Internationalization , SQL , Functions** related tag libraries
60. If JSTL tags are also not enough, CUSTOME TAGS can be used. Custom tags are written by the developers
61. Tag libs were declared in DD in the OLD way , new way no need to specify the Tag Lib in the DD
62. The container will look for TAGLIBS in following locations , Directly inside WEB-INF ,Inside a SUB DIRECTORY of WEB-INF , INSIDE META-INF of a JAR file , Inside a SU DIRECTOYR Of META-INF of a JAR file
63. <url-pattern> in web.xml which maps to a SERVLET must start with a forward slash <url-pattern>/something.do</url-pattern> , in the form <form action = ‘something.do”></form>. Or else use ASTERIC enabled mappings such as <url-pattern>*.do</url-pattern> [do not put forward slash]

64. EXPRESSION LANGUAGE (EL) was introduced in the JSP 2.0 and JSF 1.1 introduced it’s own EL. In J2EE5 both these combined and made UNIFIED EL
65. EL helps developers to remove JAVA SCRIPLETS from JSP , JSF
66. ${exp} = Expression is evaluated immediately [from JSP 2.0], #{exp} = evaluation would be differed [from JSF]
67. UNIFIED EL defines two EXPRESSIONS , VALUE EXPRESSION and METHOD EXPRESSION
68. VALUE EXPRESSIONS can either set or yield values
69. METHOD EXPRESSIONS reference method that can be INVOKED
70. CONTAINER also makes number of IMPLICIT EL object available to developers
71. CUSTOM JAVA SCRIPTLETS are not necessarily bad , but it becomes headache when developers intermix PRESENTATION with BUSINESS LOGIC
72. JSF is a UI Framework for web applications based on the J2EE platform
73. JSF is designed to be easy to use by Developers
74. It allows developers to stop thinking in terms of HTTP request and responses and instead to think about UI development in terms of user generated EVENTS
75. JSF components are REUSABLE , improves DEVELOPER PRODUCTIVITY, SOFTWARE QUALITY , SYSTEM MAINTAINABILITY
76. The clear intent of JSF is that technology should be TOOLABLE or provided with DEEP and MATURE support from IDE(s)
77. JSF maps closely to .NET

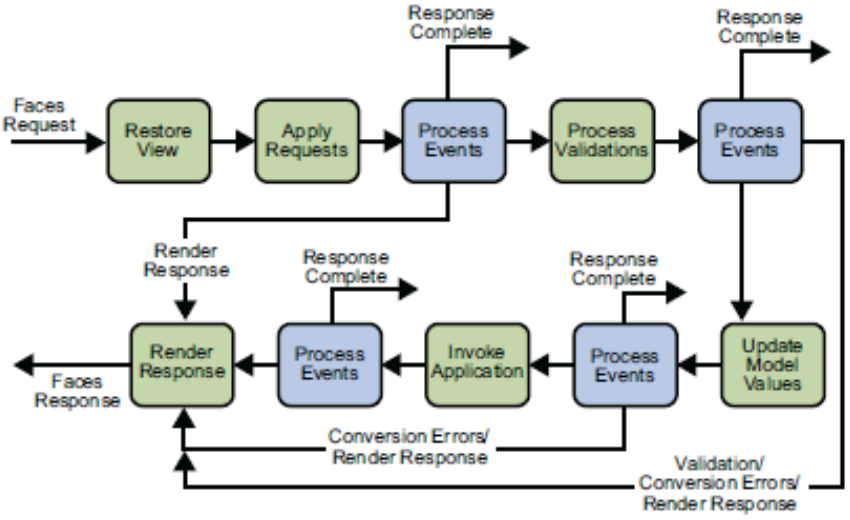
78. A WEB FRAMEWORK FILL THE GAP between JSP/SERVLET/JSF specifications
79. A good framework provides the architect and developers with a clear ROAD MAP on exactly how to implement CORE FEATURES such as ACTION HANDLERS , CLIENT side , SERVER side VALIDATIONS , how to HANDLE TRANSACTIONS in a sensible manner, integrate SECURITY, mange SESSION STATE and build a MAINTAINABLE , UNDESTANDABLE web UI
80. **JSPs and Servlets** , STANDARD USES
    a. JSP handles the presentation of data to the end user
    b. JSP should not contain BUSINESS LOGIC
    c. Good approach is to replace all JAVA Code with EL , JSTL or CUSTOM,THIRDPARTY TAGS
    d. JSP is the V in MVC
81. **JSF** STANDARD USES
    a. Standard uses of JSF are the same as JSP
    b. You have a choice , either to use JSP with JSTL and a good MVC framework or use JSF
    c. JSP and JSF are not MUTUALLY EXCLUSIVE, It is perfectly possible to make a JSP+JSP hybrid UI
82. WEB CENTRIC implementations mean that the implementation does not use EJBs at all
83. Scenarios where strong MESSAGING , TRANSACTION, SECURITY MANAGEMENT are all candidate for an EJB centric implementation
84. Scenarios where EASE OF DEVELOPMENT , EXISTING APPLICATION is WEB CENTRIC , TRANSACTIONS are not KEY to business (READ only or READ mostly) would be good for a WEB CENTRIC approach

85. If the functionality contained in the WEB CONTAINER for CONCURRENCY CONTROL , SECURITY and SESSION MANAGEMENT is sufficient , if there are no ASYNCHRONOUS messaging requirement , no JMS queue or topics to access , WEB CENTRIC approach is good enough
86. System Correctness, Reliability, Security are critical factors to select EJB centric implementation. Also it matters if the systems with which your system interact is also EJB
87. CHOOSING between EJB and WEB CENTRIC
    a. TRANSACTION REQUIREMENT – The more onerous , the bigger the reason to select EJB
    b. SECURITY REQUIREMENT – The more onerous , the bigger the reason to select EJB
    c. MESSAGING REQUIREMENT – need to integrate with ASYNCHROUNOUS messaging then EJB if present
    d. PERFORMANCE
    e. EASE OF DEVELOPMENT
    f. SCALABILITY
    g. EXISTING TEAM SKILLS / EXISTING PROJECT IMPLEMENTATIONS
88. Assuming an EFFICIENT container , STATELESS SESSION beans are as EFFICIENT ass SERVLET/ACTION handlers
89. The obvious EXCEPTION is STATEFULL SESSION BEANS. POOR SCALING and PERFORMANCE ISSUES
90. STATEFULL SESSION BEANS only suitable for very SPECIFIC cases
91. AJAX is used to give user a feel of RICH APPLICATION (web 2.0)
92. AJAX is ASYNCHRONOUS JAVA SCRIPT and XML to update web pages ASYNCHRONOUSLY

93. AJAX enhances USERS experience
94. WEB COMPONENTS are either JSP , SERVLET , JSF or WEB SERVICE ENDPOINTS
95. There are TWO types of WEB APPLICATIONS
    a. PRESENTATION ORIENTED (generated interactive web pages containing various mark up language HTML , XML and so on)
    b. SERVICE ORIENTED (implements the END POINT of a WEB SERVICE)
96. @Resource annotation is used to INJECT a DATA SOURCE , ENTERPRISE BEAN or and ENVIRONMENT ENTRY
97. Adopting a WEB FRAMEWORK will introduce ADDITIONAL COMPLEXY and PERFORMACE effects
98. JSF can be used to build a server side UI with MINIMUL EFFORT
99. One of the greatest ADVANTAGE of JSF is CLEAN SEPARATION of BEHAVIORS and PRESENTATION
100. JSF can manage UI as STATEFUL components in the WEB CONTAINER while JSP can not
101. JSF allows FINER GRAINED separation of BEHAVIORS and PRESENTATION
102. JSF UI components are CONFIGURABLE
103. JSF provides set of UI COMPONENTS , RENDERING MODEL,EVENT and LISTENERS , a CONVERSION MODEL, A VALIDATION MODEL , NAVIGATION MODEL
104. In JSF , developers can user DIFFERENT RENDER KITS to render on DIFFERENT DEVICES
105. JSF CORE TAGS are tags that are designed to perform actions that are INDEPENDENT of any RENDER KIT.

106. JSP tags defined in standard HTML tag lib represents HTML form components and other basic HTML elements
107. In JSF it provides , JSF UI Components , RENDERERS , CONVERTERS , BACKING BEANS , EVENT HANDLERS , VALIDATORS etc
108. in JSF you can create JSF resource bundle for customized messages as well
109. INTERNATIONALIZATION is also easy to achieve using JSF resource bundles
110. JSF life cycle is spit in to multiple phases to support the sophisticated UI component model
111. JSF page is different from JSP , JSF is represented as TREE OF UI components called VIEW
112. JSF life cycle is as follows

FIGURE 10–3 JavaServer Faces Standard Request-Response Life Cycle



113. JSF life cycle handle both types of **REQUEST , INITIAL REQUEST(S)** and **POST BACK (SUBMIT a page)**

114. JSF life cycle works as bellow

a. [**RESTORE VIEW**]
b. When a request to JSF is sent (Button click / Link) **RESTORE VIEW** starts
c. During this phase JSF **builds the VIEW of the PAGE,WIRES EVENT HANDLERS and VALIDATORS to COMPONENTS in the VIEW , SAVES the VIEW in the FACESCONTEXT**
d. If the **REQUEST** is an **INITIAL REQUEST** , the JSF would create an **EMPTY VIEW** and life cycle goes to **RENDER RESPONSE** phase, during which **EMPTY** view is **POPULATED** with the **COMPONENTS** referenced by the **TAGS** in the **PAGE**
e. If the **REQUEST** is a **POSTBACK**, a **VIEW** already **EXISTS**. During this phase JSF would **RESTORE** the view by using the **STATE** information saved on the **CLIENT** or the **SERVER**
f. **[APPLY REQUEST VALUES Phase]**
g. After the **COMPONENT TREE** is **RESOTED**, each component in the tree **EXTRACTS** values from the **REQUEST PARAMS** by using **DECODE** method
h. If **CONVERSION** of values fails , **ERROR** messages will be **GENERATED** and **QUEUED** to **FACESCONTEXT**
i. Those **ERROR** messages would be displayed during the **RENDER RESPONSE PHASE**
j. If some components on the page have **IMMEDIATE** attribute **TRUE** , then **VALIDATION** , **CONVERSION** and

EVENTS ASSOCIATED with the **COMPONENT** will be **PROCESSES** during this **PHASE**

k. If at this point the **APPLICATION** needs to **REDIRECT** to **DIFFERENT WEB APPLICATION** or **GENERATE** a **RESPONSE** that **DOES NOT** contain JSF components it can call **FACESCONTEXT.RESPONSECOMPLETE**

l. At this phase the **COMPONENTS** are set to their **NEW VALUES** and **MESSAGES** and **EVENTS** have been **QUEUED**

m. **[PROCESS VALIDATION phase]**

n. During this phase the JSF processes all **VALIDATORS REGISTED** on the **COMPONENTS** in the **TREE**

o. If any **ERRORS** then **ERROR MESSAGES** would be added to **FACES CONTEXT** and the JSF lifecycle would advance to **RENDER RESPONSE** phase

p. **[UPDATE MODEL VALUES phase]**

q. After JSF determines **DATA** is **VALID** , it can walk the **COMPONENT TREE** and **SET** the corresponding **SERVER SIDE OBJECT** properties

r. If any **ERRORS** then JSF lifecycle would **ADVANCE** to **RENDER RESPONSE** phase to show the **PAGE** with **ERRORS**

s. **[INVOKE APPLICATION phase]**

t. This phase JSF handles any **APPLICATION-LEVEL EVENTS**, such as **SUBMITTING** a **FORM** or **LINKING to ANOTHER PAGE**

u. If at this point if the **APPLICATION** needs to **REDIRECT** to a **DIFFERENT WEB APPLICATION RESOURCE** or generate a response that **DOES NOT CONTAIN JSF**

components , it can **CALL FacesContext.responseComplete**

v. **[RENDER RESPONSE phase]**

w. At this phase JSF implementation **DELEGATES AUTHORITY** for **RENDERING the page to the JSP CONTAINER if the APPLICATION is using JSP pages**

x. If there were **ERRORS** in previous phase during this phase the **ORIGINAL PAGE** is rendered **AGAIN** with **ERROR MESSAGES**

115. JSF supports **LOCALIZATION** for, **STATIC TEXT [Labels]**, **ALTERNATE TEXT [tooltips]**, **ERROR MESSAGES [validation etc...], and DYNAMIC DATA [backing bean related.]**

116. JSF support **CUSTOME UI** creation , **CUSTOM CONVERTERS** , **CUSTOM VALIDATORS** , **CUSTOME TAGS , CUSTOM RENDERER**

117. JSF allows **STATE saving** either on **CLIENT** or **SERVER**

118. JSF by **DEFAULT SAVES** the **STATE** on **CLIENT**. The state of **the ENTIRE VIEW** is **RENDERED** to a **HIDDEN FILED** on the **PAGE** in that case

119. To **PREVENT OTHERS SEEING** information sent to **CLIENT** , state can be **ENCYPTED** while being **TRANSFERRED from SERVER to CLIENT**

120. If the **DEPLOYMENT DESCRIPTOR is not having required information** to do **STATE ENCRYPTION** , the **STATE** would be **TRANFERRED** as **PLAIN**

121. **Access** to **JSF** pages can be **RESTRICTED** in a **SIMILAR** way that is done to **JSP using web-resource-collection** tags