**STRUTS**

**Easy to remember the differences between Struts 1. x and 2. x:**

|  |  |  |
| --- | --- | --- |
|  | **Struts 1.x** | **Struts 2.x** |
| **Configuration file** | Named as struts-config.xml | Named as struts.xml |
| **Tag Library** | Three tag libraries are required:  <%@ taglib uri = "/WEB-INF/struts-html.tld" prefix = "html" %>    <%@ taglib uri="/WEB-INF/struts-logic.tld" prefix = "logic" %>    <%@ taglib uri = "/WEB-INF/struts-bean.tld" prefix = "bean" %> | Only on tag library encapsulates all the features:  <%@ taglib uri = "/struts-tags"  prefix = "s" %> |
|  | Thread-safe methods should be declared. | Thread-safety is not an issue. |
| **Co-ordination of Beans and Action classes** | Java beans and action classes are separately defined | Getter-setter methods can be added in action classes. |
| **Servlet dependency** | Servlet dependent | Servlet independent |
| **Extending and implementing actions** | Action classes need to extend org.apache.struts.action.Action | Action classes can implement Action interface or can also extend ActionSupport class |
| **Execute() method** | Action classes have to provide body of execute() method | Implementing execute() method is not a compulsion. |
| **Front end** | Front end  is controlled by ActionServlet | Front end  is controlled by Filters |
| **Expression language** | JSTL’s EL (Expression language) is used here | For EL, OGNL (Object Graph Notation Language) is used. |
| **Validations** | Validations are to be performed in ActionForm manually | Validations are handled by XWork Validator which provides validations of certain fields automatically. |
| **Life cycle** | All the request handlers follow same life cycle. | interceptors in struts 2.x does not follow same life cycle. |

**What are interceptors and why to use them?**

The main use of interceptors is to pre-process or post-process a module in a web-site. That is, any pre-processing that is required before the action tag is executed from struts.xml and any post-processing required after the action tag is executed from struts.xml is specified by the interceptors.

Interceptors get executed before and after the action elements are called from struts.xml file.

The main purpose of using interceptors is to provide file validation, as we have seen above, validation of fields in our web page, logging, storing, handling exceptions, setting timers (time-out for some event) etc.

In nut-shell: Web-page restrictions and validations from user as well as server side can be provided by interceptors.

Also, a web page can have multiple interceptors. That is, n number of validations (pre as well as post) can be applied to a web form for different fields.

Types of interceptors in Struts 2:

The types of interceptors in Struts 2 are:

In-built

Custom

Inbuilt interceptors are predefined in struts configuration files. You just have to use it in action element when required. The types of inbuilt interceptors are:

fileUpload

execption

alias

checkbox

chain

datetimes

multiselect

i18n

params

Validation

conversionError

workflow

execAndWait

The Application server and web server in Java?

 Both are used to host Java web application. Though both application server and web server are generic terms, difference between application server and web server is a [famous J2EE interview question](http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html). On Java J2EE perspective main difference between web server and application server is support of EJB. In order to run EJB or host enterprise Java application (.ear) file you need an application server like JBoss, WebLogic, WebSphere or Glassfish, while you can still run your servlet and JSP or java web application (.war) file inside any web server like Tomcat or Jetty

1. Application Server supports distributed transaction and EJB. While Web Server only supports Servlets and JSP.

2. Application Server can contain web server in them. most of App server e.g. JBoss or WAS has Servlet and JSP container.

3. Though its not limited to Application Server but they used to provide services like Connection pooling, Transaction management, messaging, clustering, load balancing and persistence. Now Apache tomcat also provides connection pooling.

4. In terms of logical difference between web server and application server. web server is supposed to provide http protocol level service while application server provides support to web service and expose business level service e.g. EJB.

5. Application server are heavier than web server in terms of resource utilization.

Personally I don't like to ask questions like Difference between Application Server and Web Server. But since its been asked in many companies, you got to be familiar with some differences. Sometimes different interviewer expect different answer but I guess on Java's perspective until you are sure when do you need an application server and when you need a web server, you are good to go.

**JDBC Vs Hibernate**

JDBC stands for Java Database Connectivity allows developers to connect, query and update a database using the Structured Query Language. JDBC API standard provides Java developers to interact with different RDBMS and access table data through Java application without learning RDBMS details and using Database Specific JDBC Drivers.

Hibernate is an Object-Relational Mapping (ORM) solution for JAVA. It is a powerful, high performance object/relational persistence and query service. It allows us to develop persistent classes following object-oriented idiom – including association, inheritance and polymorphism.