**Descriptive module-8**

**Chapter-01**

1. **What are the advantages of JSP over servlet? Page-1**

* Servlet embed content into logic, whereas JSP pages embed logic into content.
* JSP pages contain markup interlaced with special JSP elements that provide logic for controlling the dynamic content.
* JSP pages are preferred for presenting dynamic content in the presentation tier due to their greater readability, maintainability, and simplicity.

1. **What are the advantages of Sevlet technology? Page-2**

Some of most basic advantages of servlet technology are as follows:

* **Simplicity:** Servlets are easy to write, and all the complicated threading and request delegating is managed by the servlet container.
* **Extensibility**: The Servlet API is completely protocol independent.
* **Efficiency**: Unlike CGI scripts, the execution of a servlet doesn’t require a separate process to be spawned by the web server each time.
* **Performance**: Servlets are persistent, and their life cycle extends beyond that of each HTTP request.

1. **What are the phases of JSP life cycle? Page-5**

The life cycle of JSP page can be split into four phases:

* **Translation**: When a request is first made for a JSP and if the syntax is correct, the JSP engine will translate the JSP into its page implementation class, which takes the form of standard java servlet and compile into a class file.
* **Initialization:** After the translation phase has been completed, the JSP engine loads the generated class file and creates an instance of the servlet. Then the container initializes the servlet Instance with a call to a method named jspInit().
* **Execution:** In this phase the initial request is serviced and for that, the web container calls the \_jspService() method of the implementation servlet. This method provides all the functionality for handling a request and returning a response to the client.
* **Finalization:** In this last phase, the jspDestroy() method is called to free up the resources and to shut down the servlet container. Then the servlet can no longer serve any request.

1. **Where can the tag library are reused? Page-8**

After the tags are bundled into a tag library, that tag library can be reused across the following:

* A single page.
* The pages of a web application.
* Different web applications.

1. **What are the JSP scripting elements? Page-19**

Scripting elements are used within a JSP page to manipulate objects and perform computations that enable the generation of dynamic content. Scripting elements can be classified into the following catagories:

* Comments
* Declarations
* Scriptlets
* Expressions
* Expression Language expressions

1. **What are the JSP implicit Objects? Page-23**

All JSP scripting elements have access to a number of useful objects that are provided by the JSP container and are known as JSP implicit objects. Each of these implicit objects are classes or interfaces as defined by either the servlet or JSP specifications. The objects are:

* request
* response
* out
* session
* config
* application
* page
* pageContext
* exception

1. **What are the JSP Directives? Page-25**

Directives are used for passing important information to the JSP engine. JSP page authors have the following three types of directives at their disposal:

* page directives
* include directives
* taglib directives

1. **What are the action elements? Page-34**

In JSP 2.1 there are three types of action elements:

* Standard actions
* Custom actions
* JSTL actions

1. **How many tag libraries have in JSTL?**

There are four tag libraries in JSTL:

1. Core
2. Internationalization (I18n) and Formating
3. XML
4. SQL
5. **Define the followings?**

* **JSP**: A JSP page is simply a regular text file that contains markup (usually HTML) suitable for display inside a browser. JSP technology is an extension of Servlet technology. The sole purpose of JSP technology is to produce dynamic web-based content. **Page-3**
* **Servlet**: Servlet can most simply be described as custom web**-**server extensions, whose jobs are to process requests and dynamically construct appropriate responses. **Page-2**
* **Template Text:** Any non-JSP code located inside a JSP page is known as template text. It can take any form as long as it’s text based. The most common form of template text is the markup such as HTML or XML. **Page-18**
* **Scriptlets**: Scriptlets are small block of source code contained within the <% and %> delimiters that can used to provide programming style language functionality around a page’s content, thus making their output dynamic. **Page-20**
* **Expressions**: Expressions are similar to scriptlets, but as their name suggests they evaluate a regular java expression and return a result. This result must be a string or be convertible to a string; otherwise an exception will be raised during the translation phase. The general syntax is:<%= your code or text%>. **Page-21**
* **request**: It is an instance of the javax.servlet.http.ServletRequest interface. The request object provides access to all the available information about the user request and may be used in exactly the same way that the HttpServletRequest parameter is used in the service() method of a servlet. **Page-23**
* **response**: The response object is an instance of the javax.servlet.http.HttpServletResponse interface. It represents the current response to be return to the user. It can be used in exactly the same way as the HttpServletResponse parameter received by the service() method of servlet. **Page-23**
* **out**: The implicit out object represent an instance of the javax.servlet.jsp.JspWriter class that can be used to write character data to the response stream in a similar manner to that seen by the java.io.PrintWriter class. **Page-23**
* **session**: The implicit session object provides a reference to an implementation of the client’s individual javax.servlet.http.HttpSession object, which can be used to store and retrieve data. **Page-24**
* **config**: The config object simply provides the JSP developer with access to the ServletConfig object that is used by the web container to configure the JSP page. **Page-24**
* **application**: This object provides a reference to the javax.servlet.ServletContext interface of the web application. **Page-24**
* **page**: This object references an instance of the JSP’s page implementation class and is declared of type Object. **Page-24**
* **pageContext**: A pageContext instance provides the JSP developer with access to all the available JSP scopes and to several page attributes, such as the current request and response, the ServletContext,HttpSession, and ServletConfig to name but a few. **Page-24**
* **exception**: The implicit exception object is available only to those JSP pages that declared themselves as error pages by using the following page directives: <%@ page isErrorPage=”true” %>. This object itself is an instance of java.lang.Throwable and will be represent a runtime error that occurred during the request process. **Page-25**
* **page directive**: The page directive is used to define any page dependent properties that a JSP page may have, such as library dependencies, buffering, or error-handling requirements. Example of page directive is: <%@ page session=”true” %>. **Page-25**
* **include directive**: The include directive executes at translation time and enables the contents of a separate resource to be statically merged inside the original page, thus radically affecting the generated servlet. Syntax of include directive is: <%@ include file=”ralativeURL” %>. **Page-28**
* **taglib directive**: A tag library contains a collection of actions (also known as tags) that can be grouped together to perform some form of logic. These actions are XML based, so their use is considerably easier for a non-java-speaking UI designer.
* **TLD**: It stands for Tag Library Descriptor which is an XML base file. It is used to provide general descriptive information about the custom tag library, such as a description of its usage and the JSP version that the tag supports. **Page-32**
* **URI**: It stands for Uniform Resource Identifier that identifies the TLD and therefore, the tag library that is associated with the prefix. **Page-32**
* **Standard actions:** The JSP standard actions provide the JSP page author with a (relatively small) selection of useful actions. They are:

1. The <jsp:include> action
2. The <jsp:useBean> action
3. The <jsp:getProperty> action
4. The <jsp:setProperty> action

* **Custom actions**: Custom actions are another mechanism for encapsulating functionality into reusable components for use inside JSP pages.

**Chapter-02**

|  |  |
| --- | --- |
| **Q1:** | **What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?** |
| **A:** | The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.  The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root. |
| **Q2:** | **Explain the directory structure of a web application.** |
| **A:** | The directory structure of a web application consists of two parts.  A private directory called WEB-INF A public resource directory which contains public resource folder.  WEB-INF folder consists of  1. web.xml 2. classes directory 3. lib directory |
| **Q3:** | **What are the common mechanisms used for session tracking?** |
| **A:** | Cookies SSL sessions URL- rewriting |
| **Q4:** | **Explain ServletContext.** |
| **A:** | ServletContext interface is a window for a servlet to view it's environment. A servlet can use this interface to get information such as initialization parameters for the web applicationor servlet container's version. Every web application has one and only one ServletContext and is accessible to all active resource of that application. |
| **Q5:** | **What is the difference between HttpServlet and GenericServlet?** |
| **A:** | A GenericServlet has a service() method aimed to handle requests. HttpServlet extends GenericServlet and adds support for doGet(), doPost(), doHead() methods (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() methods (HTTP 1.1).  Both these classes are abstract. |
| **Q6:** | **What is the difference between ServletContext and ServletConfig?** |
| **A:** | ServletContext: Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.The ServletContext object is contained within the ServletConfig object, which the Web server provides the servlet when the servlet is initialized   ServletConfig: The object created after a servlet is instantiated and its default constructor is read. It is created to pass initialization information to the servlet. |

**Q7. What are the uses of Servlet?**

**A:** The important used of HTTP Servlets are:  
  
- Storage and processing of data submitted by an HTML form.   
- Providing dynamic content to the client for example outputting the result of a query.   
- Improving the system performance by handling multiple requests at a time.  
- Managing state information on top of the stateless HTTP.

### Q8: Explain the differences between Jsp and Servlet.

**A:**

• The JSP is used mainly for presentation purpose  
• But the Servlets are not used for this purpose only.  
• JSP can be only HttpServlet.   
• HTTP is the only supported protocol in JSP   
• But a servlet is supported by any protocol . e.g HTTP, FTP, SMTP etc.

### Q9: What is session?

**A:**

• The session may be said as an object.  
• It is used by a servlet to track a user’s interaction.  
• It interacts with the Web application.  
• It works across multiple HTTP requests.  
• The sessions are mainly stored in the server.

### Q10: Define the servlet mapping.

**A:**

• The servlet mapping is defined as an association between the URL pattern and a servlet.  
• The mapping is used in mapping requests.  
• It maps in the Servlets only.

**Chapter-03**

1. **What is literal El language?**

**Answer:-**

Just as in any programming language, the El provides several literals for developers to use and Simply devalued to the text of the expression. A literal expression can be used anywhere a value expression can be used.

1. **What is the purpose of using El language? Where we use it?**

**Answer:**

Purpose->

1. Short hand notation for bean properties

2. Simple access to collection elements

3. Automatic type conversion

4. Conditional output

Where we use->

We can use the El in the same places as we would have used a scriplet, for example-

1. Within attribute values for JSP standard and custom tag
2. Within template text(that is in the body of the page)
3. **What is El Function?**

**Answer:-**

An El function is mapped to a static method of a java class and can appear either in temple text or in the attribute of a custom tag. A function in El can take any number of parameters and these are again declared in a deployment descriptor.

1. **What is the namespaces in El function?**

**Answer:-**

Functions must always have a namespaces. Function are assigned a namespace that is used to

Access a function similar to package specifications in java classes. For example

${myfunctions:function(“param”)}

The namespace in this example is myfunctions.

1. **What are implicit objects in Expression language?**

**Answer:-**

Like PSP implicit objects, El implicit objects also allow us to access things such as request, session, pageContext and a lot more beside objects are->

1. ApplicationScope

2. Cookie

3. header

4. initParam

6. pageContext

7. pageScope

8. param

9. paramValues

10. requestScope

11. sessionScope

1. **El Function VS custom tags?**

**Answer:-**

|  |  |
| --- | --- |
| El Function | Custom Tags |
| 1.Function do not provide easy access to pageContext and variable | 1.A tag provide easy access to pageContext and other variable |
| 2. Function do not provide functionality to process a body | 2. Provides iterative behavior over a body |
| 3. 1.Function are must simpler to write | 3. Tag are less simpler to write than function |
| 4. Function provide to reuse existing java code in a web content | 4. |

**Chapter-04**

1. **What is the difference between custom tag and scriptlet based web development?**

Using too many scriptlets on a page reduces its readability and therefore its maintainability and generally makes a JSP page look ugly. When using scriptlets, it’s all too easy to forget a closing brace. Custom tags are far better than the scriptlet-based approach, because the use of a custom tag places no assumptions on the skills of the page author—custom tags are simply XML based. Java developers can create their own custom actions (often known as custom tags) and make them available to web designers via tag libraries.

1. **What is JSTL ?**

JSTL denotes **JavaServer Pages Standard Tag Library.** JSTL is action element of JSP. It is first released in June 2002. The JSTL specification outlines a number of custom actions that are designed to handle the vast majority of common tasks needed by JSP page authors that avoids many of the problems associated with scriptlets.

1. **How many tag libraries make JSTL tag library?**

The JSTL is often referred to as a single tag library when in fact it’s a collection of four tag

libraries. Each tag library provides useful actions (or tags) based on the following functional

areas:

• Core

• Internationalization (I18n) and formatting

• Relational database access

• XML processing

1. **What do you mean by Core Tag Library?**

As mentioned previously, the Core tag library provides JSP page authors with a set of reusable

actions to cater to the simple “core” requirements that almost every JSP application has in some shape or form, such as object manipulation, looping, and so on.

1. **What do you mean by Internationalization and Formatting Tag Library(FMT).**

Preparing an application so it’s ready for the global marketplace is known as internationalization (or I18n for short). A related term, localization (or l10n), refers to the process of customizing an application for a particular language or region.

1. **What is Formatting Actions ?**

Ensuring that your clients view your JSP pages in their own language is just the tip of the iceberg

with regard to building a fully internationalized and localized application. Such as..

• Date and time formats

• Number formats

• Currency formats

• Colors

• Page layouts

• Address standards (zip codes)

1. **What do you mean by SQL Tag Library.**

The primary job of the <sql:setDataSource> action is therefore to configure a data source. The configured data source is used by the remaining actions in the SQL library to source database connections so they may perform queries and updates, and so on.

1. **What do you mean by XML Processing Tag Library.**

Increasingly, XML is also being used internally by web applications to represent data retrieved from the business or database layer, which is then rendered into an appropriate format (HTML, WML, and so on) by the presentation layer that requiring specific programming skills of the page author. To address this problem, the JSTL provides an XML processing tag library that is designed to solve many of the common tasks met by page authors using XML data.

9. **Write the functional area of XML Processing Tag Library.**

The XML processing tag library can be split into the following functional areas concerned

with XML data:

• XML core actions

• XML flow control actions

• XML transformation actions.

**Chapter-5**

**1.What is the Managed Beans? Page-200.**

**Ans:** JavaBeans use to separate presentation from business logic. They are simply Java classes that conform to a certain coding style. Two aspect of javaBeans are

1. The JavaBean used in the web application must have a no-argument constructor.
2. Any property to be exposed must have a get or set method.

**Q2.What are the bean scopes?page204**.

Ans: If we configure a java bean to be used in a jsp page, we can configure with one of the four scopes. They are

1.Request

2.Session

3.Application

4.none

**Q3.what is page Navigation? Page 2017**

Ans : Page navigation in jsf application is handled by configuration file. The navigation can specify which web component initiates request , handles the response. Two Navigations are

1.Static Navigation

2.Dynamic Navigation

**Q4.Why is jsf used for?**

Ans: Jsf is used for

1.MVC for web application

2.Easy to use

3.Extensible component

4.support for the client device independent

5.standard

6.Huge vendor and industry support

**Q6.Describe the relationship between jsf and other java technologies.page 184**

Ans: Java technologies such as jsp, servlet, jstl are stand-alone technologies. Jsf is different because it is supporting technology. The prime design pattern of jsf is the Model-View-Controller(MVC) pattern.MVC separates application into three categories which is easy to maintain and control process is reliable.

**Q7. What are the purpose of value-Binding expression jsp pages ?** page205

Ans : Value-Binding is the process that the jsf implementation can set or get the value of javaBean properties. If we want to get the value from bean file, we can use the bean file object .(dot) and the specific property. Example:  
<h:inputText value=”#{flight.fname}” size=”35”/>

**Q8.What are the Jsf Method binding ?**

Ans: Just like Value-Binding process to model, we can also bind a method to expression. The syntax of method binding is same as value binding. We can call by using .(dot) or bracket notation .They are

1.Action

2.Validators

3.action listeners

4.Value change listeners

**Q09.Why we use validations? Page 236**.

Ans: Validation is ensured that data entered by user is correct. Jsf provides to simplify data validation through the use of standard validation and custom validation. These validations are used to confirm data is entered correctly or not. Actually these are used for user input.

**Chapter-6**

**TAG FILES AND SIMPLE TAGS**

**(1) What are the Valid Body-Content Values for Tag Extensions?**

**Ans:** The Valid Body-Content Values for Tag Extensions are:-

1. Empty
2. JSP
3. Scriptless
4. Tagdependent.

**(2) Write down the Differences Between Simple Tags and Classic Tags?**

Ans: There are several important differences between simple and classic tags.

To build classic tags, we write the functionality provided by the custom tag as a Java class

that implements the javax.servlet.jsp.tagext.Tag interface. This class containing the func-tionality provided by the tag is called the tag handler.

Tag files and simple tags allow the functionality of custom tags to be implemented by using JSP fragments and Java code, respectively. Althoughthey use different paradigms, they both greatly simplify the way in which custom tags can be built and have changed the way that JSP-based web applications are built.

Other changes to custom tags introduced in JSP 2.0 include the ability of a custom tag to

take any number of undefined (dynamic) attributes, and the ability to use the JSP EL with cus-

tom tags. both changes affect classic tagsand simple tags.

**(3) Why do we need custom tags or what are the key factors benefit of using custom tag?**

Ans: Expe-rience has shown us that three key factors benefit from this practice:

• Reusability

• Readability

• Maintainability

**(4)How you can use tag in JSP page?**

Ans: A defined and deployed tag library must be made available to a JSP page if it’s to be used. After the tag library has been imported, the tags within that library can be used with

XML syntax. For example, if the tag library contained a tag called copyright, you simply use

the tag as follows:

<tags:copyright></tags:copyright>

that tag can be

used in its shortened format as follows:

<tags:copyright/>

**(6) What is scriptless body content value?**

Ans: scriptless body content is effectively the same as JSP body content with

the restriction that it cannot contain any Java code wrapped up as

scriptlets.

**(7) How attributes are written within tag?**

Ans: Custom tags can be customized through the use of attributes

in the same way that methods can be customized through the use of parameters. Attributes are written as name=value

pairs within the tag itself, as shown here:

<prefix:myTag attributeName="attribute value"/>

**(8) Difference between JavaBeans and Custom Tags?**

Ans:

JavaBeans are good general-purpose objects that encapsulate state in a portable “bucket.”

Tags are a web-specific technology. Tags are primarily for generating presentation ele-

ments, and as such they primarily encapsulate behavior.

Use JavaBeans for representing and storing information and state. An example is build-

ing JavaBeans to represent the business objects in your application.

Use custom tags to represent and implement actions that occur on those JavaBeans, as

well as logic related to the presentation of information. An example from JSTL is iterating over a collection of objects or conditional logic.

**(9) What is tag files and how it is used?**

Ans: Tag files provide a very simple way for content and functionality to be abstracted away from

JSP pages and into reusable components. a tag file is simply a JSP fragment containing some content or JSP code that you would like to use over and over again.

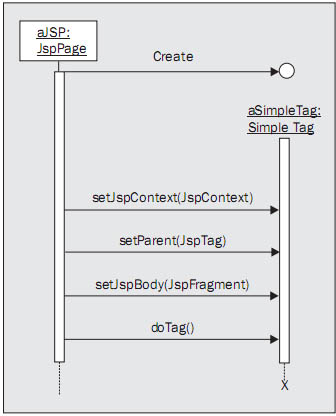
Tag files are used from within a JSP page as a custom tag is used in the same way. Before using tag files on the page, we have to indicate that the tag file is a tag and import the tag library containing that tag.

**(10) Why Use Tag Files?**

Ans: Tag files provide a much cleaner way to buildand subsequently use templates on JSP pages. Using tag files to build templates is a great way to separate the content from the presentation of that content, and this makes it a natural progression from simply mixing the two togetherin JSP pages.

**(11) What is The Basic Tag Life Cycle**

Ans: The Life Cycle of a simple tag is:



**(12) What is a Custom Tag?**

Ans: A custom tag is a user-defined JSP language element. When a JSP page containing a custom tag is translated into a servlet, the tag is converted to operations on a tag handler. The web container then invokes those operations when the JSP page’s servlet is executed.

**(13) Write the syntax of Deploying the Tag Library?**

Ans: If the TLD file located at WEB-INF\tlds\ch06.tld then we can directly import this on a JSP page by

using the following directive:

<%@ taglib uri="/WEB-INF/tlds/ch06.tld" prefix="ch06" %>

**(14) What is tag handler?**

Ans: When a custom tag is used on a JSP page, the first thing that the JSP container must do is create a new instance of the tag handler class. Tag Handler is the Java class that will embody the functionality that the custom tag will provide.