JSP INTERVIEW QUESTION

1. WHAT IS JSP AND IMPORTANCE OF IT.

JSP stands for JavaServer Pages. It is a technology that allows developers to create dynamic web pages using the Java programming language. JSP technology is built on top of the Java Servlet API, which provides the underlying functionality for handling requests and responses between the web server and the client.

JSP is important because it allows developers to create dynamic web pages that can be customized based on user input or other external factors. For example, a JSP page can be used to display data from a database, process user input from a form, or generate content based on the user's preferences or location.

JSP pages are also highly reusable, allowing developers to separate presentation logic from application logic and build complex web applications more easily. Additionally, JSP pages can be used in conjunction with other Java technologies such as Servlets, JavaBeans, and Enterprise JavaBeans (EJBs) to create powerful and scalable web applications.

Overall, JSP technology is an important tool for building dynamic, interactive, and scalable web applications using the Java programming language.

2. WHAT IS JSP LIFE-CYCLE. (SERVLET LIFECYCLE+ CONVERSION)

The JSP lifecycle is the sequence of phases that a JSP page goes through from its creation to its destruction. The JSP lifecycle can be broken down into the following phases:

1. Translation: The JSP container first translates the JSP page into a Servlet, which can be executed by the server. During this phase, the container also

checks the syntax of the JSP page and generates any necessary code for handling requests and responses.

2. Compilation: The translated Servlet code is then compiled into bytecode by the container. This bytecode can be executed by the Java Virtual Machine (JVM).

3. Initialization: During the initialization phase, the container creates an instance of the Servlet class and calls its `init()` method. This method is used to initialize any resources that the JSP page may require, such as database connections or other objects.

4. Request Processing: When a request is received for the JSP page, the container creates a new thread and calls the `service()` method of the Servlet class. This method is responsible for handling the request and generating the response.

5. Destruction: When the JSP page is no longer needed, the container calls the `destroy()` method of the Servlet class. This method is used to release any resources that the JSP page may have allocated, such as database connections or other objects.

In terms of conversion, the JSP lifecycle is closely related to the Servlet lifecycle. When a JSP page is translated into a Servlet, the resulting code follows the Servlet lifecycle. This means that the JSP page is treated as a Servlet by the container and goes through the same phases as a Servlet, such as initialization, request processing, and destruction. The main difference is that JSP pages provide a simpler and more convenient way to generate dynamic web content using Java code and HTML markup.

3.WHAT ARE IMPLICIT OBJECT IN JSP

In JSP (JavaServer Pages), implicit objects are objects that are automatically created by the JSP container and are available to developers without needing to be explicitly declared or instantiated. These objects provide developers with easy access to various resources and information, such as HTTP requests and responses, session information, and application context.

Here is a list of some of the commonly used implicit objects in JSP:

1. request: The `request` object provides access to the HTTP request that is sent by the client to the server.

2. response: The `response` object provides access to the HTTP response that is sent by the server to the client.

3. session: The `session` object provides access to the session data that is associated with the current user's session.

4. application: The `application` object provides access to the servlet context, which represents the entire web application.

5. out: The `out` object is an instance of the JSPWriter class and is used to send output to the client.

6. pageContext: The `pageContext` object provides access to various JSP-related objects, such as the `request`, `response`, and `session` objects.

7. config: The `config` object provides access to the configuration information for the current JSP page.

8. exception: The `exception` object provides access to any exception that occurs during the execution of the JSP page.

By using these implicit objects, developers can quickly and easily access various resources and information within their JSP pages, without needing to write a lot of boilerplate code.

4.WHAT IS EXPRESSION LANGUAGE.

Expression Language (EL) is a scripting language that is used to simplify the process of accessing data and manipulating values within JSP pages and other Java-based web applications. It provides a simple and concise syntax for evaluating and manipulating data, without the need for complex Java code.

EL is based on the Unified Expression Language (UEL) standard, which was introduced in Java EE 5. It allows developers to access data from various sources, such as request parameters, session attributes, and application-scoped variables, using a simple syntax. EL also provides a set of operators and

functions for manipulating values and performing operations, such as arithmetic, comparisons, and string concatenation.

Here are some examples of how EL can be used in JSP pages:

1. Displaying a value from a request parameter:

```
<p>The user's name is ${param.username}</p>
```

2. Calculating a value based on session attributes:

```
<p>The user's age is ${sessionScope.userAge + 10}</p>
```

3. Using conditional statements to display content:

```
<c:if test="${param.username == 'admin'}">
  <p>Welcome admin!</p>
</c:if>
```

EL is designed to simplify the process of accessing and manipulating data within JSP pages and other Java-based web applications. By providing a simple and concise syntax for evaluating and manipulating data, EL allows developers to create more powerful and dynamic web applications with less code.

5. NAME 4 BASIC JSP TAGS. (ANS: SCRIPLET TAG (<% %>), EXPRESSION TAG (<%= %>), DECLERATION TAG (<%! %>), DIRECTION TAG OR PAGE DIRECTIVE(<%@ %>))

Yes, you are correct. Here are the four basic JSP tags:

1. Scriptlet tag (`<% %>`): The scriptlet tag is used to embed Java code directly into the JSP page. Any code within the scriptlet tags is executed when the JSP page is processed by the server.

2. Expression tag (`<%= %>`): The expression tag is used to evaluate and output the value of a Java expression within the JSP page. The expression is evaluated and the result is included in the output HTML page.

3. Declaration tag (`<%! %>`): The declaration tag is used to declare instance variables and methods within the JSP page. This tag is typically used to define reusable code that can be called from other parts of the JSP page.

4. Directive tag or Page directive (`<%@ %>`): The directive tag is used to provide instructions to the JSP container on how to handle the JSP page. The most commonly used directive is the `page` directive, which is used to set various attributes of the JSP page, such as the language used, error page, and import statements.

6. WHAT IS THE FUNCTION OF isELIgnored?
The `isELIgnored` attribute is a boolean attribute of the JSP `page` directive that is used to indicate whether the use of Expression Language (EL) is enabled or disabled in the current JSP page.

By default, EL is enabled in JSP pages, which means that any EL expressions in the JSP page will be evaluated by the JSP container. However, in some cases, you may want to disable EL for a particular JSP page, such as when using custom scripting languages or when performance is a concern.

The `isELIgnored` attribute can be set to either `true` or `false`. If it is set to `true`, EL expressions are ignored and not evaluated in the JSP page. If it is set to `false`, EL expressions are evaluated and processed by the JSP container as usual.

Here is an example of how to use the `isELIgnored` attribute in a JSP page:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isELIgnored="true" %>

<html>
<head>
    <title>Example JSP page</title>
</head>
<body>
    <p>The use of EL is disabled in this JSP page.</p>
</body>
```

</html>
```

In this example, the `isELIgnored` attribute is set to `true`, which means that any EL expressions in the JSP page will be ignored and not evaluated by the JSP container. Instead, the JSP page simply displays a message indicating that the use of EL is disabled.

## 7.WHAT IS NAMING CONVENCTION OF CONVERTING JSP TO SERVLET CLASS

When a JSP file is compiled, the JSP container automatically generates a corresponding servlet class that implements the behavior of the JSP page. The naming convention for the generated servlet class is as follows:

- The servlet class name is derived from the JSP file name.
- The `.jsp` file extension is replaced with `.java`.
- Any special characters in the JSP file name, such as hyphens or periods, are converted to underscores in the servlet class name.
- The servlet class is placed in the `work` directory of the JSP container, typically under a subdirectory named `org/apache/jsp`.

For example, if you have a JSP file named `my_page.jsp`, the corresponding servlet class generated by the JSP container will be named `my_page_jsp.java`.

It's worth noting that in modern JSP development, it's generally recommended to separate the presentation logic (i.e., the JSP page) from the business logic (i.e., the servlet). This approach is known as the Model-View-Controller (MVC) design pattern, and it involves using servlets to handle requests and responses, and using JSP pages to display data and generate HTML output. By separating the presentation logic from the business logic, you can achieve greater flexibility and maintainability in your web application.

## 8.WHAT IS JSTL.

JSTL stands for JavaServer Pages Standard Tag Library. It is a collection of custom tags that provide common functionality for JSP pages. JSTL is included in the Java EE platform and provides a convenient way to perform common tasks such as looping over collections, conditional processing, internationalization, and accessing databases.
```

JSTL includes four core tag libraries:

1. Core tags: Provides functionality for iteration, conditional processing, URL management, and manipulating variables.

2. XML tags: Provides functionality for processing XML documents.

3. SQL tags: Provides functionality for interacting with relational databases.

4. Functions tags: Provides a set of utility functions that can be used in JSP expressions.

Using JSTL can make JSP pages more concise and easier to read, as well as more maintainable and portable across different web containers. It also promotes good design practices, such as separating presentation logic from business logic.

Here is an example of using the JSTL `core` tag library to iterate over a collection and display its elements:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>
<head>
   <title>Example JSTL page</title>
</head>
<body>
   <ul>
      <c:forEach var="item" items="${items}">
         <li>${item}</li>
      </c:forEach>
   </ul>
</body>
</html>
```

In this example, the `c:forEach` tag is used to iterate over a collection of items, which is passed as an attribute using the Expression Language (EL) syntax `${items}`. The `var` attribute specifies the name of the loop variable, which is `item` in this case. The body of the `c:forEach` tag contains the HTML code

that is executed for each item in the collection. In this case, it simply displays the item value using `${item}` syntax.