UNIT 3

Part - A

1. What is JSP?
   Ans: **JSP** stands for **Java Server Pages**. It is a server-side technology which is used for creating web applications. It is used to create dynamic web content. JSP consists of both HTML tags and JSP tags

2. What is MVC?

Ans: MVC stands for Model-View-Controller. It is a software design pattern that separates the application logic into three distinct layers:

- Model: The model represents the data and business logic of the application.
- View: The view represents the presentation of the data to the user.
- Controller: The controller acts as an intermediary between the model and the view. It receives user input and updates the model accordingly. It then generates a view based on the updated model and sends it to the user.

3. What is ASP?

Ans: ASP stands for Active Server Pages. It is a server-side scripting technology that was developed by Microsoft. ASP pages are written in HTML and VBScript, and they are executed on the server before being sent to the client.

4. What is Model Layer?
   Ans: The model layer in advanced Java is a layer of abstraction that encapsulates the data and business logic of an application. It provides a clean and efficient way to interact with the data. The model layer is typically implemented using object-oriented programming (OOP) principles. This allows the data and business logic to be encapsulated in reusable classes.

5. What is View Layer?
   Ans: The view layer in advanced Java is a layer of abstraction that encapsulates the presentation of the data to the user. the view layer is typically implemented using user interface (UI) frameworks such as Swing, JavaFX, or Android UI toolkit.

6. What is Controller Layer?
   Ans: The controller layer in advanced Java is a layer of abstraction that acts as an intermediary between the view layer and the model layer. It receives requests from the view layer, interacts with the model layer to fulfill those requests, and then returns the results to the view layer.

7. What is Implicit Objects?
   Ans: implicit objects are objects that are automatically created and made available to all JSP pages. They provide a convenient way to access common functionality, such as the request, response, and session objects.

Part - B

1. What are the advantages of JSP?

Ans : **Advantages of JSP over Servlet**

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet
In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc

.

Advantages of JSP

Following table lists out the other advantages of using JSP over other technologies − **vs. Active Server Pages (ASP)**

The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers. **vs. Pure Servlets**

It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

vs. Server-Side Includes (SSI)

SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like. **vs. JavaScript**

JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

vs. Static HTML

Regular HTML, of course, cannot contain dynamic information.

---

2. Explain the life cycle of a JSP.

Ans : A JSP life cycle is defined as the process from its creation till the destruction. This is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

The following are the paths followed by a JSP –

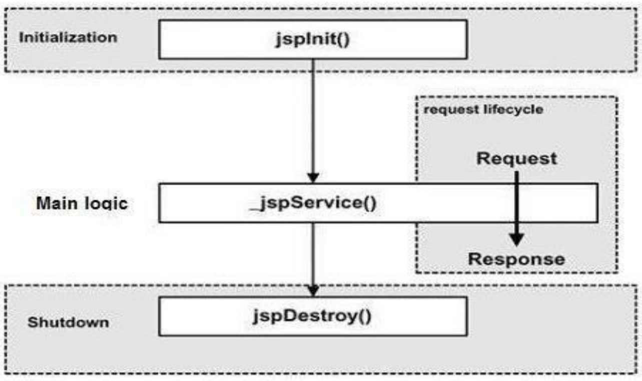
1.Compilation

2.Initialization

3.Execution

4.Cleanup

The four major phases of a JSP life cycle are very similar to the Servlet Life Cycle. The four phases have been described below –



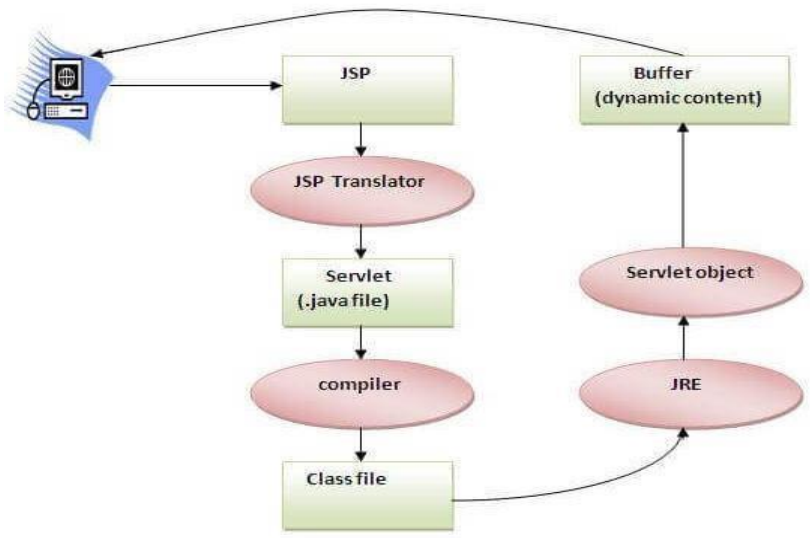


Figure3.1:JSP Architecture

JSP Compilation

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.

The compilation process involves three steps − ¢ Parsing the JSP.

- Turning the JSP into a

servlet.

- Compiling the servlet
- .

JSP Initialization

When a container loads a JSP it invokes the jspInit() method before servicing any requests. If you need to perform JSP-specific initialization, override the jspInit() method − public void jspInit(){

// Initialization code...

}

Typically, initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

JSP Execution

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.

Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the _jspService() method in the JSP.

The _jspService() method takes an HttpServletRequest and an HttpServletResponse as its parameters as follows −

void _jspService(HttpServletRequest request, HttpServletResponse response) { // Service handling code...

}

The _jspService() method of a JSP is invoked on request basis. This is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods, i.e, GET, POST, DELETE, etc.

JSP Cleanup

The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.

The jspDestroy() method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files.

The jspDestroy() method has the following form

```
− public void jspDestroy() {

   // Your cleanup code goes here.

}
```

----

3. Explain the JSP processing?

Ans: JSP processing in advanced Java is the process of compiling and executing a JSP page. JSP pages are server-side scripting pages that are used to generate dynamic web content. JSP pages are written in HTML, XML, and Java code.

The JSP processing lifecycle consists of the following steps:

1. Translation: The JSP page is translated into a Java servlet.
2. Compilation: The Java servlet is compiled into a Java bytecode file.
3. Loading: The Java bytecode file is loaded into the Java Virtual Machine (JVM).
4. Initialization: The servlet is initialized.
5. Service: The servlet service method is called to handle the request.
6. Destruction: The servlet is destroyed.

Translation:

The JSP page is translated into a Java servlet using the JSP translator. The JSP translator converts the HTML, XML, and Java code in the JSP page into Java code.

Compilation:

The Java servlet is compiled into a Java bytecode file using the Java compiler. The Java compiler converts the Java code in the servlet into bytecode.

Loading:

The Java bytecode file is loaded into the JVM using the class loader. The class loader loads the bytecode file into memory so that it can be executed by the JVM.

Initialization:

The servlet is initialized by calling the init() method. The init() method is used to initialize the servlet and its state.

Service:

The servlet service method is called to handle the request. The service() method is used to process the request and generate the response.

Destruction:

The servlet is destroyed by calling the destroy() method. The destroy() method is used to clean up the servlet and its state.

JSP processing in advanced Java can be used to implement a variety of features, such as:

- Dynamic content generation: JSP pages can be used to generate dynamic content, such as the current date and time, or the results of a database query.
- User interaction: JSP pages can be used to interact with users, such as collecting user input or displaying personalized content.
- Data access: JSP pages can be used to access data, such as data stored in a database or a file system.
- Business logic: JSP pages can be used to implement business logic, such as validating user input or processing a transaction.

----

4. Explain types of JSP Implicit Objects

Ans: These Objects are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared.

JSP Implicit Objects are also called pre-defined variables.

The request Object

The request object is an instance of a javax.servlet.http.HttpServletRequest object. Each time a client requests a page the JSP engine creates a new object to represent that request. The request object provides methods to get the HTTP header information including form data, cookies, HTTP methods etc.

The response Object

The response object is an instance of a javax.servlet.http.HttpServletResponse object. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes, etc.

The out Object

The out implicit object is an instance of a javax.servlet.jsp.JspWriter object and is used to send content in a response.

The session Object

The session object is an instance of javax.servlet.http.HttpSession and behaves exactly the same way that session objects behave under Java Servlets.

The session object is used to track client session between client requests.

The application Object

The application object is direct wrapper around the ServletContext object for the generated Servlet and in reality an instance of a javax.servlet.ServletContext object.

This object is a representation of the JSP page through its entire lifecycle. This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the jspDestroy() method.

The config Object

The config object is an instantiation of javax.servlet.ServletConfig and is a direct wrapper around the ServletConfig object for the generated servlet.

This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc.

The following config method is the only one you might ever use, and its usage is trivial

− config.getServletName();

This returns the servlet name, which is the string contained in the <servlet-name> element defined in the WEB-INF\web.xml file.

The pageContext Object

The pageContext object is an instance of a javax.servlet.jsp.PageContext object.

The pageContext object is used to represent the entire JSP page.

This object is intended as a means to access information about the page while avoiding most of the implementation details.

This object stores references to the request and response objects for each request. The application, config, session, and out objects are derived by accessing attributes of this object.

The page Object

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page.

The page object is really a direct synonym for the this object.

The exception Object

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

----

5. Explain the JSP form processing.

Ans: **1. GET method**

This method is the default method used by the browser. This method sends the information attested to the requested URL separated by question mark sign (?). As this method sends data in an appended format, it is not recommended to use for sensitive information like passwords. However, this method has a limitation of characters i.e., 1024 bytes only in a request string.

Example:

```
<html>
<body>
<form action = "get1.jsp" method = "GET">
First Name: <input type = "text" name = "first_name">
<br />
Last Name: <input type = "text" name = "last_name" />
<input type = "submit" value = "Submit" />
</form>
</body>
</html>
```

When you run this file, you will get a form window. Fill in the credentials and click on the submit button.

The form action will be passed to the jsp file to process the information. Here is the jsp code

```
<html>
<head>
<title>GET using form</title>
</head>
<body>
<h2>--DataFlair--</h2>
<h2>GET Method to Read Form Data Using form</h2>
<ul>
<li><p><b>First Name:</b>
<%= request.getParameter("first_name")%>
</p></li>
<li><p><b>Last Name:</b>
<%= request.getParameter("last_name")%>
</p></li>
</ul>
</body>
</html>
```

2. POST method

This method of passing information is more reliable than the method above. This method is used for sensitive information. The POST method sends the information as a separate text message rather than appending the message in the URL. Thus it becomes more dependable. The information that comes from this method comes as standard input to the backend

program.
Form Processing in JSP is handled using the following methods:

1.getParameter():
This method is used to get the value of the specified parameter.
2. getParameterValues():
This method returns multiple values of the specified parameter. In the case of form, this situation can arise when we use checkboxes.
3. getParameterNames()
This method returns the name of the parameters.
4. getInputStream()
This method reads the binary data sent by the client.
Example:

post.htm
```
<html>
<body>
<form action = "post.jsp" method = "POST">
First Name: <input type = "text" name = "first_name">
<br />
Last Name: <input type = "text" name = "last_name" />
<input type = "submit" value = "Submit" />
</form>
</body>
</html>
```

This html form file will take the input from the user and pass the action to the jsp file to handle the data. The code for .jsp file will be:

```
<html>
<head>
<title>Using POST Method</title>
</head>
<body>
<h1>POST Method to Read Form Data</h1>
<b>First Name:</b>
<%= request.getParameter("first_name")%>
</br>
<b>Last Name:</b>
<%= request.getParameter("last_name")%>
</body>
</html>
```

6.Explain the JSP Database access with example(skipped)

7.Define JSP Standard Tag Libraries and Custom Tag

Ans: The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications.

JSTL tags are divided into three categories:

- Core tags: These tags provide basic functionality, such as conditional logic, loops, and output.
- Formatting tags: These tags provide formatting functionality, such as escaping HTML, internationalization, and date formatting.
- SQL tags: These tags provide database access functionality, such as retrieving data from a database and inserting, updating, and deleting data in a database.

S.No.	Tag & Description
1	<c:out> Like <%= ... >, but for expressions.
2	<c:set > Sets the result of an expression evaluation in a 'scope'
3	<c:remove > Removes a scoped variable (from a particular scope, if specified).
4	<c:catch> Catches any Throwable that occurs in its body and optionally exposes it.
5	<c:if> Simple conditional tag which evalutes its body if the supplied condition is true.

Custom Tag

A custom tag is a JSP tag that is written by the developer. Custom tags can be used to encapsulate reusable functionality and make JSP pages more maintainable and readable.   To create a custom tag, the developer must create a Java class that extends the javax.servlet.jsp.tagext.TagSupport class. The custom tag class must implement the doStartTag(), doEndTag(), and doAfterBody() methods.

The doStartTag() method is called when the custom tag is first encountered in the JSP page. The doEndTag() method is called when the custom tag is finished executing. The doAfterBody() method is called after the body of the custom tag has been evaluated.Custom tags can be used to perform a variety of tasks, such as formatting data, validating user input, and generating dynamic content.

There are several benefits to using JSP Standard Tag Libraries and Custom Tags, including:

- Increased productivity: JSTL tags and custom tags can save developers a lot of time by providing reusable functionality.
- Improved maintainability: JSTL tags and custom tags can make JSP pages more maintainable by encapsulating reusable functionality.
- Increased readability: JSTL tags and custom tags can make JSP pages more readable by making it easier to understand the flow of execution.
- Reduced code duplication: JSTL tags and custom tags can help to reduce code duplication by providing reusable functionality

----

8).Define JSP expression Language ?

Ans: JSP EL allows you to create expressions both (a) arithmetic and (b) logical. Within a JSP EL expression, you can use integers, floating point numbers, strings, the built-in constants true and false for boolean values, and null.

Simple Syntax

Typically, when you specify an attribute value in a JSP tag, you simply use a string. For example − <jsp:setProperty name = "box" property = "perimeter" value = "100"/>

Basic Operators in EL

JSP Expression Language (EL) supports most of the arithmetic and logical operators supported by Java. Following table lists out the most frequently used operators −

S.No.	Operator & Description
1	. Access a bean property or Map entry
2	[] Access an array or List element
3	( ) Group a subexpression to change the evaluation order
4	+ Addition
5	- Subtraction or negation of a value
6	* Multiplication
7	/ or div Division

8	% or mod Modulo (remainder)
9	== or eq Test for equality
10	!= or ne Test for inequality
11	< or lt Test for less than
12	> or gt Test for greater than
13	<= or le Test for less than or equal
14	>= or ge Test for greater than or equal
15	&& or and Test for logical AND
16	|| or or Test for logical OR
17	! or not Unary Boolean complement
18	Empty Test for empty variable values

----

9).Define JSP Exception handling with example ?
Ans: When you are writing a JSP code, you might make coding errors which can occur at any part of the code. There may occur the following type of errors in your JSP code − Checked exceptions

A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation. Runtime exceptions

A runtime exception is an exception that probably could have been avoided by the programmer. As opposed to the checked exceptions, runtime exceptions are ignored at the time of compliation. Errors

These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error
Example:

index.html

- HTML

```
<html>
<head>
<body>
<form action="a.jsp">
Number1:<input type="text" name="first" >
Number2:<input type="text" name="second" >
<input type="submit" value="divide">
</form>
</body>
</html>
```

A.jsp

- Java

```
// JSP code to divide two numbers
<% @page errorPage = "error.jsp" %> < %

                                   String num1
    = request.getParameter("first");
String num2 = request.getParameter("second");

// extracting numbers from request
int x = Integer.parseInt(num1);
int y = Integer.parseInt(num2);
int z = x / y; // dividing the numbers
out.print("division of numbers is: " + z); // result

% >
```

error.jsp

- Java

```
// JSP code for error page, which displays the exception
<% @page isErrorPage = "true" %>

    <h1> Exception caught</ h1>

        The exception is : <%= exception %> // displaying the exception
```

Part - C

1. What are the steps to execute JSP page?

Ans : To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page. **index.jsp**

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page. We will learn scriptlet tag later.

```
<html>

<body>

<% out.print(2*5); %>

</body>

</html>
```

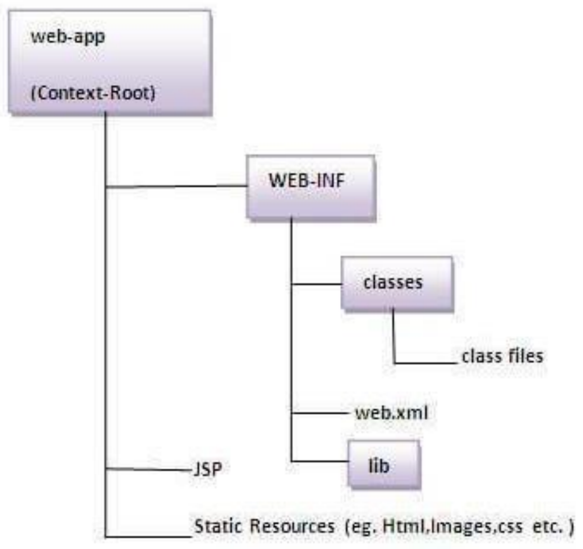
Follow the following steps to execute this JSP page:

- Start the server
- Put the JSP file in a folder and deploy on the server
- Visit the browser by the URL http://localhost:portno/contextRoot/jspfile, for example, http://localhost:8888/myapplication/index.jsp

There is no need of directory structure if you don't have class files or TLD files. For example, put JSP files in a folder directly and deploy that folder. It will be running fine. However, if you are using Bean class, Servlet or TLD file, the directory structure is required.

The Directory structure of JSP

The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.



2).Explain the MVC Architecture in JSP.

Ans : MVC is an architecture that separates business logic, presentation and data. In MVC, M stands for Model V stands for View C stands for controller.

MVC is a systematic way to use the application where the flow starts from the view layer, where the request is raised and processed in controller layer and sent to model layer to insert data and get back the success or failure message.

Model Layer:

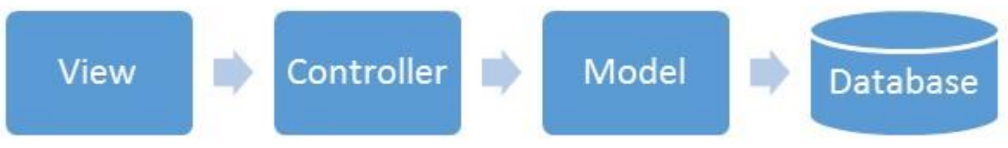
This is the data layer which consists of the business logic of the system. It consists of all the data of the application It also represents the state of the application.
It consists of classes which have the connection to the database.
The controller connects with model and fetches the data and sends to the view layer.
The model connects with the database as well and stores the data into a database which is connected to it.

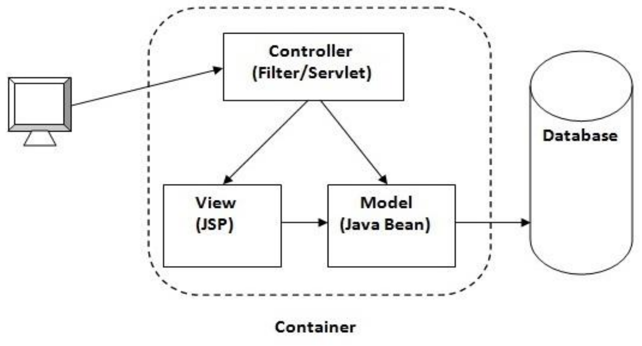
View Layer:

This is a presentation layer.
It consists of HTML, JSP, etc. into it.
It normally presents the UI of the application.
It is used to display the data which is fetched from the controller which in turn fetching data from model layer classes.
This view layer shows the data on UI of the application.

Controller Layer:

It acts as an interface between View and Model.
It intercepts all the requests which are coming from the view layer.
It receives the requests from the view layer and processes the requests and does the necessary validation for the request.
This request is further sent to model layer for data processing, and once the request is processed, it sends back to the controller with required information and displayed accordingly by the view.
The diagram is represented below:





The advantages of

Easy to maintain
Easy to extend
Easy to test
Navigation control is centralized

----

3).Explain the JSP Session and Cookies Handling in JSP ?

Ans : HTTP is a "stateless" protocol which means each time a client retrieves a Webpage, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

Maintaining Session Between Web Client And Server

Let us now discuss a few options to maintain the session between the Web Client and the Web Server

<u>Cookies</u>

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the received cookie. This may not be an effective way as the browser at times does not support a cookie. It is not recommended to use this procedure to maintain the sessions.

<u>Hidden Form Fields</u>

A web server can send a hidden HTML form field along with a unique session ID as follows −

<input type = "hidden" name = "sessionid" value = "12345">

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or the POST data. Each time the web browser sends the request back, the session_id value can be used to keep the track of different web browsers. This can be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

<u>URL Rewriting</u>

You can append some extra data at the end of each URL. This data identifies the session; the server can associate that session identifier with the data it has stored about that session. For example, with http://tutorialspoint.com/file.htm;sessionid=12345, the session identifier is attached as sessionid = 12345 which can be accessed at the web server to identify the client. URL rewriting is a better way to maintain sessions and works for the browsers when they don't support cookies. The drawback here is that you will have to generate every URL dynamically to assign a session ID though page is a simple static HTML page.