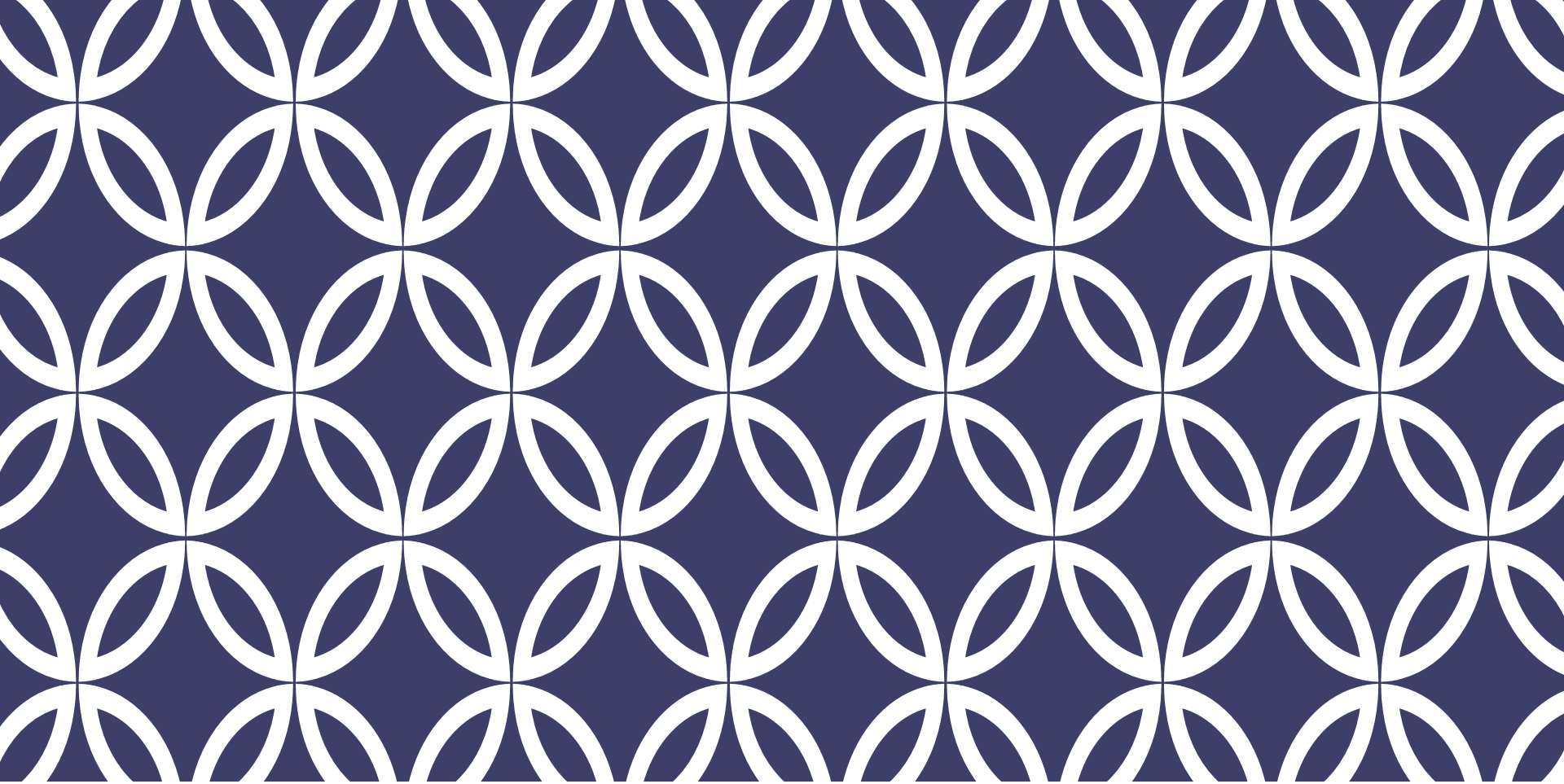
ACADGILD

Presents

Introduction to Advance Java J2EE



Session 19 – Introduction to Web

Agenda – Introduction to Web

1. Introduction to Web
2. Understanding URL
3. Why we need Servlet and JSPs?
4. Web Project Structure
5. Web Container
6. Web Application Directory Structure
7. Deployment Descriptor
8. Java Servlets
9. Servlet Life Cycle
10. Cookies
11. Sessions

Introduction to Web

- **Web Applications** are used to create dynamic websites. Java provides support for web application through**Servlets** and **JSPs**. We can create a website with static HTML pages but when we want information to be dynamic, we need web application.

- **Web Server** is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

- **A web client** is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

Understanding URL

- URL is acronym of Universal Resource Locator and it's used to locate the server and resource. Every resource on the web has it's own unique address. Let's see parts of URL with an example.

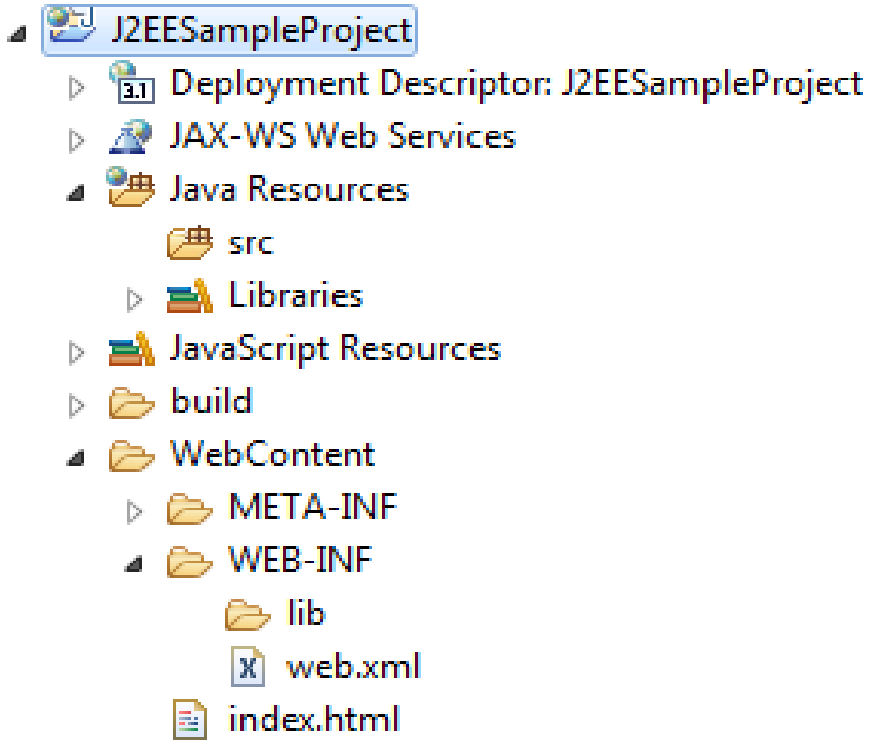
  http://localhost:8080/FirstServletProject/jsps/hello.jsp

- **http://** – This is the first part of URL and provides the communication protocol to be used in server-client communication.
- **localhost** – The unique address of the server, most of the times it's the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.
- **8080** – This is the port on which server is listening, it's optional and if we don't provide it in URL then request goes to the default port of the protocol. Port numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.
- **FirstServletProject/jsps/hello.jsp** – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

Why we need Servlet and JSPs?

- Web servers are good for static contents HTML pages but they don't know how to generate dynamic content or how to save data into databases, so we need another tool that we can use to generate dynamic content. There are several programming languages for dynamic content like PHP, Python, Ruby on Rails, Java Servlets and JSPs.
- Java Servlet and JSPs are server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

Web Project Structure



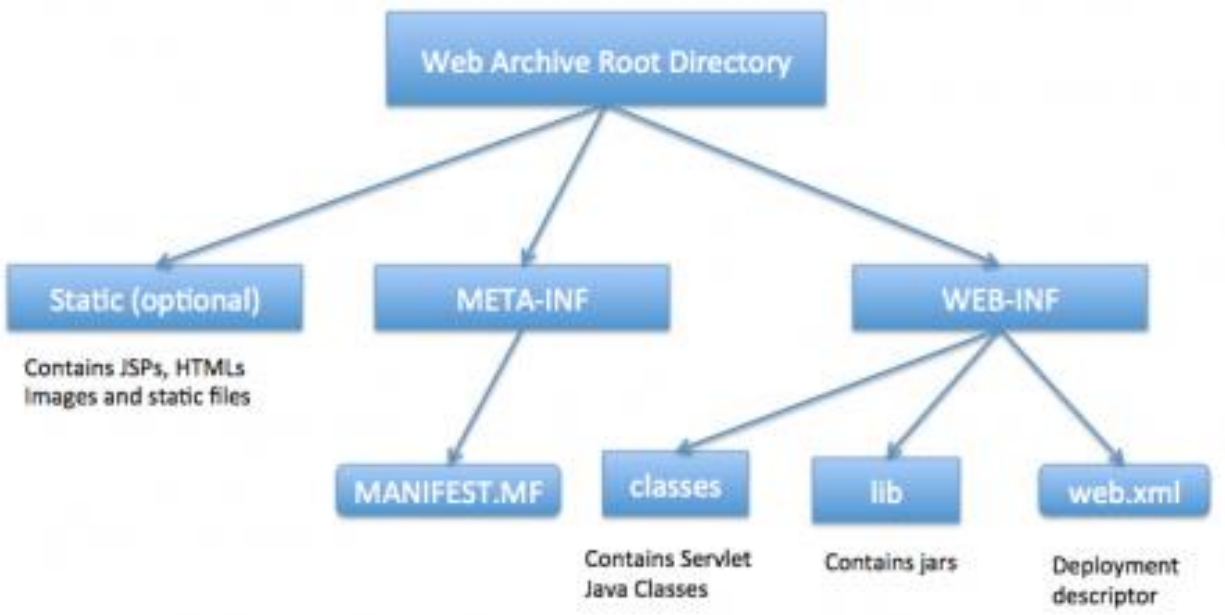
Web Container

1. Tomcat is a web container, when a request is made from Client to web server, it passes the request to web container.
2. It's web container job is to find the correct resource to handle the request (servlet or JSP) and then use the response from the resource to generate the response and provide it to web server.
3. Then web server sends the response back to the client.

Web Application Directory Structure

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure.

You can export above dynamic web project as WAR file and unzip it to check the hierarchy. It will be something like below image.



Deployment Descriptor

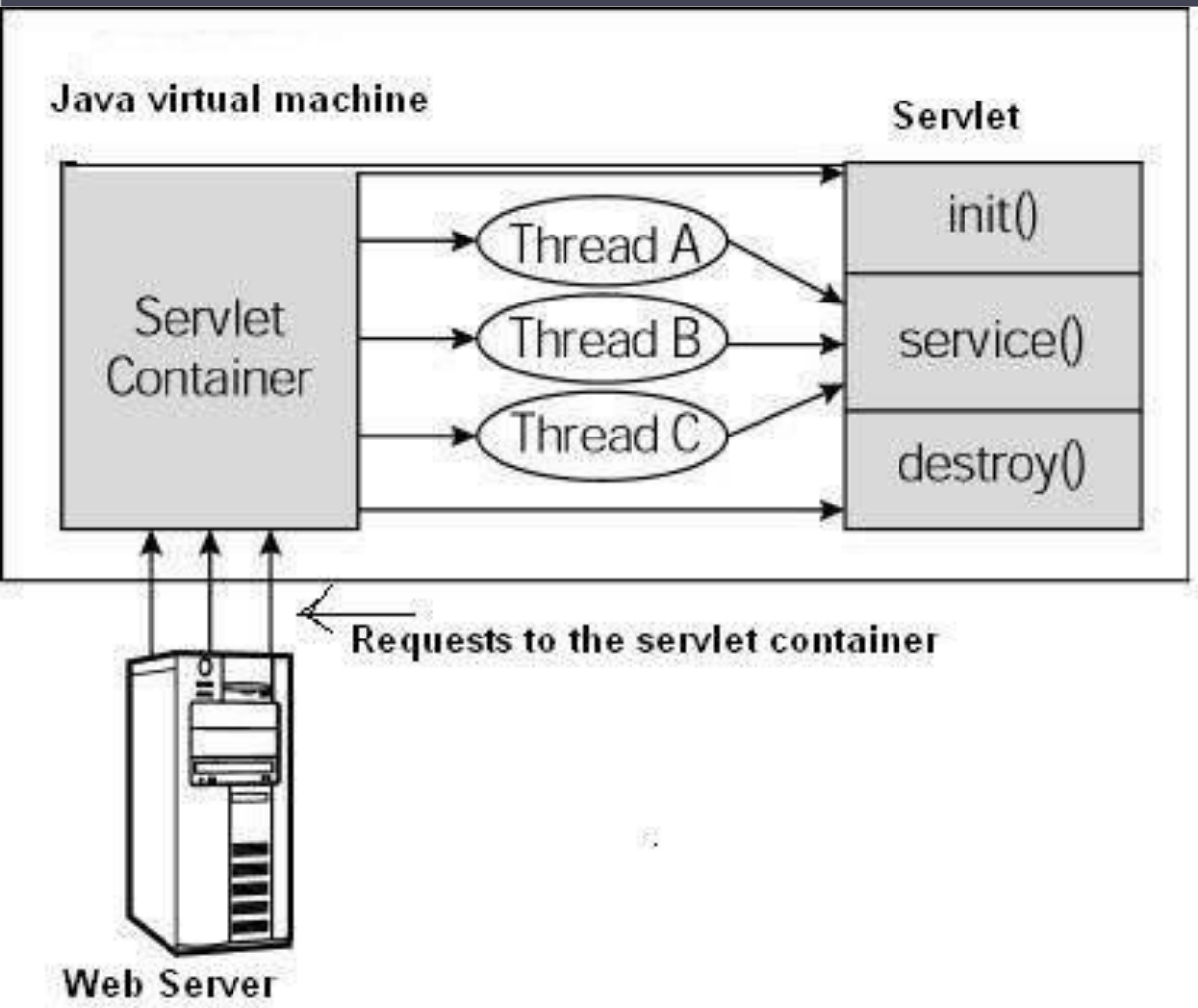
web.xml file is the deployment descriptor of the web application and contains mapping for servlets, welcome pages, security configurations, session timeout settings etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>ServletCookies</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>s1</servlet-name>
        <servlet-class>com.acadgild.servletcookiesdemo.FirstServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>s1</servlet-name>
        <url-pattern>/servlet1</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>s2</servlet-name>
        <servlet-class>com.acadgild.servletcookiesdemo.SecondServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>s2</servlet-name>
        <url-pattern>/servlet2</url-pattern>
    </servlet-mapping>

</web-app>
```

What is Servlet?

- Java Servlets are programs that run on a Web or Application server.
- They act as a middle layer between a request and response.
- Request can be from Web browser or other HTTP client.
- Response can be databases or application on HTTP server.

Pictorial representation of Java Servlet



1. Server gets HTTP requests
2. Server redirects them to Servlet container
3. Servlet container loads Servlet & then calls it's service() method
4. Servlet container handles multiple requests by spawning multiple threads.
5. Each thread executing service() method is single instance of Servlet.

Java Servlet Packages

1. Java Servlets are created and compiled just like any Java class.

2. Technically talking, Servlet is an interface in Java.

3. Packages in which Servlets are found are **javax.servlet** & **javax.servlet.http**

4. Servlet interface needs to be implemented for creating any Servlet

5. Servlets service HTTP requests and implement **javax.servlet.Servlet** interface.

6. Servlets written in web applications typically extends javax.servlet.http.HttpServlet

7. **javax.servlet.http.HttpServlet** class is abstract class that implements Servlet interface and is specially designed to handle HTTP requests.

Servlet Life Cycle

- Servlet life cycle can be defined as the entire process from its creation to destruction.

Servlet takes following path in it's life cycle.

- Servlet is initialized by calling the **init ()** method
- Servlet calls **service()** method to process a client's request
- Servlet is terminated by calling the **destroy()** method
- Finally, Servlet is garbage collected by the garbage collector of the JVM

Now let's see each of phase in details…

Phases of Servlet Life Cycle

init phase

- init method is designed to be called only once.
- It is called when Servlet is first created and not called again for each user request
- So, it is used for one-time initializations, just as with the init method of applets.
- Servlet is normally created when a user first invokes a URL corresponding to the Servlet
- When a user invokes a Servlet, a single instance of each Servlet gets created
- init() method simply creates or loads some data that will be used throughout it's life

Method signature:

```
public void init() throws ServletException
{
  // Initialization code
}
```

Phases of Servlet Life Cycle

Service phase :

- service() method is the main method to perform the actual task
- Servlet container (i.e. web server) calls service() method for 2 purposes
- For handling requests coming from client & for writing formatted response back to client
- Each time server receives request for Servlet, server spawns a new thread & calls service()
- The service() method checks HTTP request type (GET, POST, PUT, DELETE etc) & calls doGet, doPost, doPut, doDelete methods accordingly.

Method signature:

```
public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException {
    // Implementation of business logic
}
```

doGet() Method

A GET request results from a HTML form that has NO METHOD specified in it.

In other words, default request is GET request which has to be handled by doGet() method

Method signature:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
   throws ServletException, IOException
{
   // Servlet code
}
```

doPost() Method :

A POST request resulting from HTML form that lists POST specified in it.

Such request has to be handled by doPost() method

Method signature:

```
public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
   throws ServletException, IOException
{
   // Servlet code
}
```

destroy() Method :

- The destroy() method is called only once at the end of the life cycle of a Servlet.
- This method gives your Servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities
- After the destroy() method is called, the Servlet object is marked for garbage collection

Method signature:

```
public void destroy()
{
   // Finalization code
}
```

Sample Servlet Code

For creating first hello world project using Servlet, follow these instructions In eclipse

File -> New -> Project -> Dynamic web project
Create new project with some good name & click finish.

Once project is created, expand it. You will see a folder named **Java Resources** there.
Now, **right-click on Java Resources -> New -> Servlet**

Give your Servlet a name & click **finish.**

Eclipse will create a .java file for you & will open it for coding.
Here is the place where you can write your business logic for implementing it in Servlet.

NOTE : When you create new Servlet by wizard, eclipse makes its entry in web.xml.
If you are creating new class & then adding rest of code then you will have to make its entry explicitly in web.xml file.

Sample Servlet Code

```
import java.io.*;
import javax.servlet.*;

public class HelloWorld implements Servlet
{

    public void service(ServletRequest req,ServletResponse res)
                        throws IOException,ServletException
    {

            res.setContentType("text/html");
            PrintWriter out=res.getWriter();
            out.print("<html><body>");
            out.print("<b>Hello Servlets</b>");
            out.print("</body></html>");

    }

}
```

Sample Servlet Code

- We can also override other life cycle methods.
- To override them in your code just add following snippet to your code

```
public void init() throws ServletException
{
    System.out.println("Servlet initialized!");
}

public void init() throws ServletException
{
    System.out.println("Servlet destroyed!");
}
```

Output of Servlet Code

Expect output something of the tune of:

Cookies

What is cookie?

- HTTP Cookies are little pieces of data that web application can store on client machine of user visiting web application.
- It can store up to 4 kilo bytes of data.
- Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.
- If browser is configured to store cookies, it will keep this information
- Cookies are usually set in an HTTP header

Cookies Method

- **setDomain(String pattern)** : sets the domain to which cookie applies
- **getDomain()** : gets the domain to which cookie applies
- **setMaxAge(int expiry)** : sets time (in seconds) should elapse before the cookie expires
- **getMaxAge()** : returns maximum age of the cookie
- **getName()** : returns the name of cookie
- **setValue(String newValue)** : sets value associated with cookie
- **getValue()** : gets value associated with cookie
- **setPath(String uri)** : sets path to which this cookie applies
- **getPath()** : gets the path to which this cookie applies
- **setSecure(boolean flag)** : if set cookie will be sent over encrypted connections (i.e. SSL)
- **setComment(String purpose)** : specifies comment that describes a cookie's purpose
- **getComment() :** returns comment of cookie

Code snippet to illustrate usage of cookie methods

```
Cookie first_name = new Cookie("first_name","Andy");
Cookie last_name = new Cookie("last_name","Rubin");
```

Above code will create 2 cookies for you.

```
firstName.setMaxAge(60*60*24);
lastName.setMaxAge(60*60*24);
```

These 2 lines will set their ages

```
first_name.setComment("Sample cookie created for demo!");
```

This will set comment for first cookie

You can set cookies in one Servlet & can read them back from other Servlet Cookies once set, can be read any time from other Servlet as:

```
Cookie[] cookies = request.getCookies();
```

Where request is parameter of HttpServletRequest type. Once you get all cookies in array, you can search for specific cookie by key.

Session

- HTTP is a "**stateless**" protocol
- Each time client retrieves web page, client opens a separate connection to the server
- Server automatically does NOT keep any record of previous client requests.
- There are few ways to maintain session between client & server
- One of those ways is **Session**!
- **HttpSession** identifies user across more than one page request & also stores information about that user
- Servlet container uses HttpSession to create session between an HTTP client & server
- The session persists for specified time period, across many requests from the user
- We can get HttpSession object by calling **getSession()** of **HttpServletRequest**

Session Methods

- **getAttribute(String name):** returns object bound with specified name in this session
- **getAttributeNames():** returns names of all objects bound to this session
- **getCreationTime():** returns time when this session was created
- **getId():** returns a string containing the unique identifier assigned to session
- **getLastAccessedTime():** returns last time client sent request associated with session
- **invalidate():** invalidates session and unbinds any objects bound to it
- **isNew()**: returns true if the client does not yet know about the session
- **removeAttribute(String name)**: removes specified bound object from session
- **setAttribute(String name, Object value):** binds object to session by name specified
- **getMaxInactiveInterval()**: returns time for which container will keep session open

Code snippet to illustrate usage of session methods

HttpSession session = request.getSession(true);

Session got created.

session.getCreationTime();

This will get it’s creatin time

session.setAttribute(“username”, “java”);

Sets username for maintaining over session

session.getId();

Fetches id of your session

Lets Discuss Assignments