SERVLET

-- How to run a servlet Basics
1. First write servlet program
2. Write web.xml (deployment descriptor)
3. In terminal .. build war file (Web Archive)
    - To deploy and run in any other server
        - JAva App - JAR file
        - Java Web App - WAR file
4. ./gradlew war in the project home directory path
5. Copy the war file into the Tomcat webapps directory
6. cp /Users/dhilli/codebase/hariniWebApplicationUsingServlets/build/libs/hariniWebApplicationUsingServlets-1.0-SNAPSHOT.war /Library/Tomcat/webapps/hariniWebApplicationUsingServlets.war
7. Start Tomcat and run the servlet program or html or jsp in the browser, give the project name with the url-pattern name given in web.xml file

```
war:war org.codehaus.mojo:wagon-maven-plugin:upload-single
-Dwagon.fromFile=D:\Training\lowes_javafullstack-dih-batch2\labs\examples\employee-mgmt-app03\target\employee-mgmt-app03.war
-Dwagon.url=file://C:\Programs\Tomcat\apache-tomcat-9.0.63\webapps
```

```
war:war org.codehaus.mojo:wagon-maven-plugin:upload-single
-Dwagon.fromFile=/Users/dhilli/codebase/lowes_javafullstack-dih-batch2/labs/examples/employee-mgmt-app03/target/employee-mgmt-app03.war
-Dwagon.url=/Library/Tomcat/webapps/
```

```
    cp /Users/dhilli/codebase/lowes_javafullstack-dih-batch2/labs/examples/employee-mgmt-app03/target/employee-mgmt-app03.war /Library/Tomcat/webapps/
```

— Servlet API

```
    - Servlet            -> GenericServlet -> HttpServlet
    - ServletConfig          -> GenericServlet -> HttpServlet
    - ServletContext
    - RequestDispatcher
    - ServletRequest     -> HttpServletRequest
    - ServletResponse -> HttpServletResponse
```

-- Client-Server Architecture

When a client computer submits a data request to the server over the internet, the server accepts the request and returns the digital
data required to the client.
-- Client: A client is a person or an organisation that sends a request and recieves the response for a service.
In the digital world, a client is capable of sending and receiving information using a specific service from the service providers.
-- Server: A server is a person or medium who serves something.
In the digital world, a server is a remote computer that provides data or access to specific services.
-- Servlet
A Java programme that runs on a Web server is known as a Servlet. They act as a link between a web browser's request and the server's response.
A servlet's life cycle consists of three primary methods.
The init() method
The service() method
The destroy() method
-- init() method: start of the servlet
There is only one call to the init method. It's only called once when the servlet is created, and it's never called again.
It's used for one-time initializations.
-- service() method: like a main method
The main method for performing the actual task is the service() method. To handle requests from browsers and return responses to them,
the servlet container invokes the service() method.
doGet(): This method is called by servlet service method to handle the GET request from the client.
doPost(): This method is called by servlet service method to handle the POST request from the client.
-- destroy() method: At the end of a servlet's life cycle, the destroy() function is only invoked once.
This method allows your servlet to conduct cleaning tasks like closing database connections.

— Deployment Descriptor: (web.xml)
It is the file used by the servlet container to define which servlets match up with which URLs.

-- The RequestDispatcher interface : To call one servlet from the other servlet. It can be s servlet or a jsp or html. to share the data between the resources(servlet or jsp or html)
It has 2 methods forward(), include()
forward() : This method forwards the request and the response to the 2nd servlet. Here, the request and the response objects are same as the 1st servlet . The

complete response from the 2nd servlet is sent to the client (browser)
Ex: RequestDispatcher rd = req.getRequestDispatcher("/servletpath");
    rd.forward(req,res); // sends the req, res objects to the 2nd Servlet
include() : This method is used to include the 2md servlet data into the 1st servlet itself .
The response of the 2nd servlet is included in the response of the 1st servlet and the response is sent back to the client(browser) from the 1st servlet itself.
Ex: req.getRequestDispatcher("login.html").include(request, response); // the response of the file login.html is included in the response of the 1st servlet

-- SendRedirect : Here, the response from 1st servlet is sent to the client(browser) with a data in the URL (URL rewriting). It redirects the response.
The client resends the request as a new request with this data to the 2nd servlet and the 2nd servlet process the request and sends
a new response to the client(browser).
Ex: res.sendRedirect("/servlet2Path?k=" + k) // the response sent from servlet 1 to client asking to redirect to the servlet2 along with the data in URL rewriting

SendRedirect can be done using URLRewriting or cookies

-- Difference between forwrd() and sendRedirect()
forward() method   sendRedirect() method
The forward() method works at server side.   The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.
It always sends a new request.
It can work within the server only.   It can be used within and outside the server.
Example: request.getRequestDispacher("servlet2").forward(request,response);
Example: response.sendRedirect("servlet2");

-- Attribute in Servlet
The information from one servlet to another can be passed using atributes.
It is just like passing object from one class to another so that we can reuse the same object again and again.
Scope: Life time or visibility of a variable or attribute is known as scope
An attribute in servlet is an object that can be set, get or removed from one of the following scopes:
1. HttpServletRequest scope
2. Session scope
3. ServletContext scope or application scope

4. Page Context

1.HttpServletRequest object scope:
The request data is available in an object of HttpServletRequest or ServletRequest in the form of (key, value) form.
The scope of HttpServletRequest object is limited to only service or doGet or doPost methods of a particular servlet.
Example: to pass a data temporarily to another servlet
2. Servlet Context scope:
Servletcontext object refers to the whole web application.
An object of ServletContext is used for having a common data for a group of servlets which belongs to same web application.
The data of ServletContext object can be accessed through out all the methods of the servlet of a particular web.xml
Example:Load-on-startup
Basically used for giving equal response for all the clients who are accessing a particular web application.
By default after making request the ServletContext object will be created by servlet container.
Because of this first response takes more amount of time and further responses will take minimum amount of time.

Real world Example of Attribute Scope:
In Gmail, each and every email request can be considered as in the Request Scope
If a User logins and sends emails and do some tasks ans then signs out, this comes under a complete session and the scope is session scope.
If all the users of gmail and the whole gamil application is need to be controlled (for example, setting only 100 emails must visible in a page),
this type of controlling can be done under Application scope.

-- Session Management
The Session management is a way to maintain state(data) of a user or to recognize the particular user.
Session means a particular interval of time.

HTTP is stateless that means each request is considered as the new request.
Hence we need to maintain state using session tracking techniques.
1. Cookies
2. Hidden Form Field
3. URL Rewriting

4. HttpSession

1. Cookies :
A piece of data that can be stored on the client machine of users who try to visit the web application. or
A small piece of information that is persisted between the multiple client requests.

How Cookie works
By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet.
So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default.
Thus, we recognize the user as the old user.
TYPES:
  Non-persistent cookie
  It is valid for single session only. It is removed each time when user closes the browser.

  Persistent cookie
  It is valid for multiple session . It is not removed each time when user closes the browser. It is removed only if user logout or signout.
Advantage of Cookies
  Simplest technique of maintaining the state.
  Cookies are maintained at client side.
Disadvantage of Cookies
  It will not work if cookie is disabled from the browser.
  Only textual information can be set in Cookie object.
Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

To add cookie:
```
Cookie c = new Cookie("cookiename","cookievalue");
response.addcookie(c);
```
To get the Cookie:
```
Cookie[] cookie = req.getCookies();
String uid = null;
for(Cookie c : cookie) {
if("uid".equals(c.getName())) {
uid = c.getValue();
}
}
```

2. Hidden form Field:
a hidden (invisible) textfield is used for maintaining the state of an user.

```
<input type="hidden" name="uname" value="Vimal Jaiswal">
```

Real application of hidden form field
It is widely used in comment form of a website.
In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Advantage :
It will always work whether cookie is disabled or not.

Disadvantage:
It is maintained at server side.
Extra form submission is required on each pages.
Only textual information can be used.

3. URL Rewriting:
We maintain the state of the user using link.For this purpose in URL rewriting,
we append a token or identifier to the URL of the next Servlet or the next resource.
We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??
here name and value separated by =
parameter pair name/value is separated by &

Advantage of URL Rewriting
It will always work whether cookie is disabled or not (browser independent).
Extra form submission is not required on each pages.
Disadvantage of URL Rewriting
It will work only with links.
It can send Only textual information.

4. HttpSession:
The container creates a session id for each user.
The container uses this id to identify the particular user.
An object of HttpSession can be used to perform two tasks:
  bind objects
  view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.
Note: For every request , the container creates a unique session with a id.

-- ServletContext or ServletConfig : (To get the initial values of the servlet or the applicaiton)

ServletContext: To get the initial values for all the servlets in the whole application.
In web.xml, specify as given below

```
<context-param>
<param-name> name <param-name>
<param-value> AppName  <param-value>
</context-param>
```

To get the context param values , write this code in the servlet

```
ServletContext cntx = getServletContext();
String str = cntx.getInitParameter("name");
out.println(str); // here it prints the AppName given in the web.xml
```

ServletConfig:
To get the initial values of the given servlet. It is Servlet specific.
If you have different values for different servlets in an application, we can get those values using ServletConfig
In web.xml, specify as given below

```
<Servlet>
<Servlet-name>servlet1 </Servlet-name>
<Servlet-Class>servlet1 </Servlet-Class>
<init-param>
<param-name> name <param-name>
<param-value> value  <param-value>
</init-param>
</Servlet>
```

To get the config values , write this code in the servlet

```
ServletConfig config = getServletConfig();
String str = config.getInitParameter("name");
out.println(str); // here it prints the value given in the web.xml inside the servlet1
```

