Deloitte.

Java Servlets

USI CBO CR LAUNCHPAD TRAINING PROGRAM

Web Applications

Context, Objectives, Agenda

Context

- *Servlets are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web server.*
- *Servlets work on the server side and sends response to the front end so as to show user specific data.*
- *Servlets connect front end to the database.*

Objectives

- To learn
  - What is web application and where/how is it used in real time
  - How to create a Web application using Java Servlets
  - Different ways in which we can create servlets
  - What is a session and how it can be used to authenticate a user

Agenda

Topic	Content
Web Applications	• Introduction • Process Flow
Servlets	• Features & Introduction • Servlet API & Interface • Life Cycle
Servlet Collaboration	• RequestDispatcher • SendRedirect • Example
Servlet – Hands on session	• Servlet Registration Example • Servlet CRUD Example
Session Tracking	• What is Session? • Session Techniques
EOD Activities	• Assignment of the day

Web Applications

Web Applications

Objectives

01 **Introduction**

02 **Process Flow**

03 **Steps to implement Web Apps**

04 **Knowledge Check**

Web Applications

Introduction

What are Web applications?

01

Definition

- A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.
- Web applications use a combination of server-side scripts (PHP and ASP) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users.
- This allows users to interact with the company using online forms, content management systems, shopping carts and more. In addition, the applications allow employees to create documents, share information, collaborate on projects, and work on common documents regardless of location or device.

02

Web application implementation

- Web applications are usually coded in browser-supported language such as JavaScript and HTML as these languages rely on the browser to render the program executable.
- Some of the applications are dynamic, requiring server-side processing. Others are completely static with no processing required at the server.
- The web application architecture describes the interactions between applications, databases, and middleware systems on the web. It ensures that multiple applications work simultaneously.

03

Examples

- Web applications include online forms, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning, and email programs such as Gmail, Yahoo and AOL.
- Popular applications include Google Apps and Microsoft 365.

Web Applications

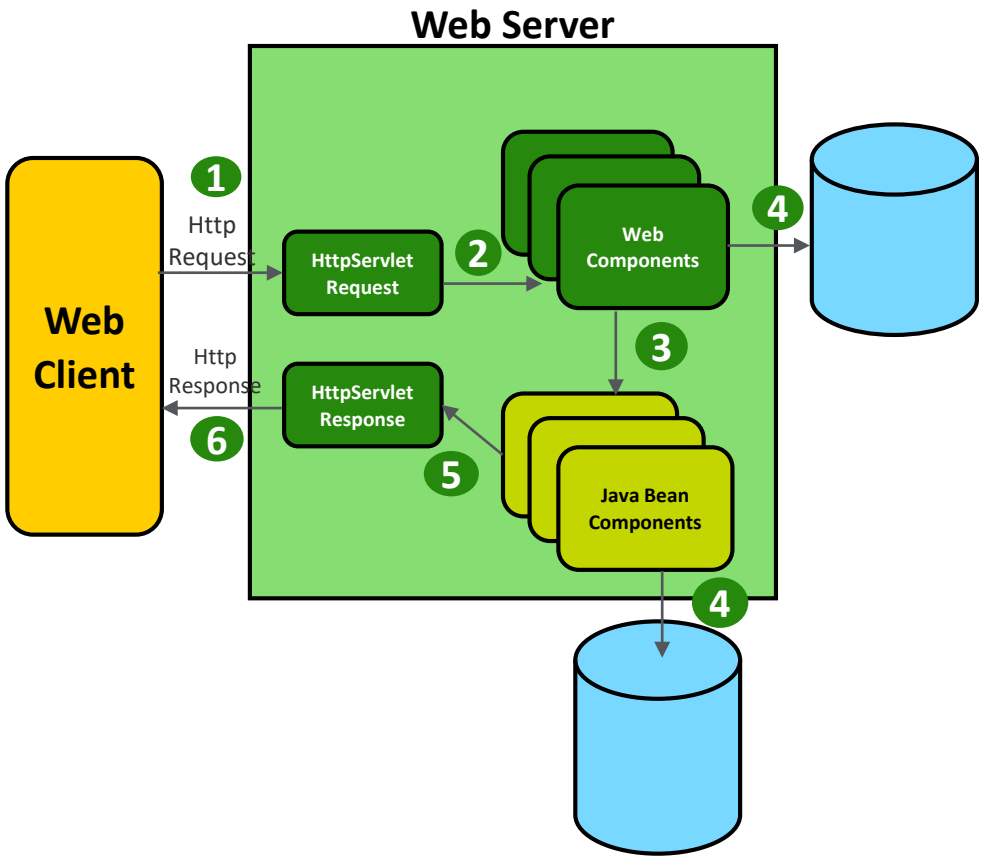
Process Flow

Web Client

- The client, or user, side of the Web.
- It typically refers to the Web browser in the user's machine.
- It may also refer to plug-ins and helper applications that enhance the browser to support special services from the site.

HttpServletRequest / Response

- The request and response communications over the internet between client and server are held using the HTTP protocol.
- That's why the server is often called as HTTP server.
- When a server answers a request, the server usually sends some type of content to the browser so that the browser can display it.



Web Server

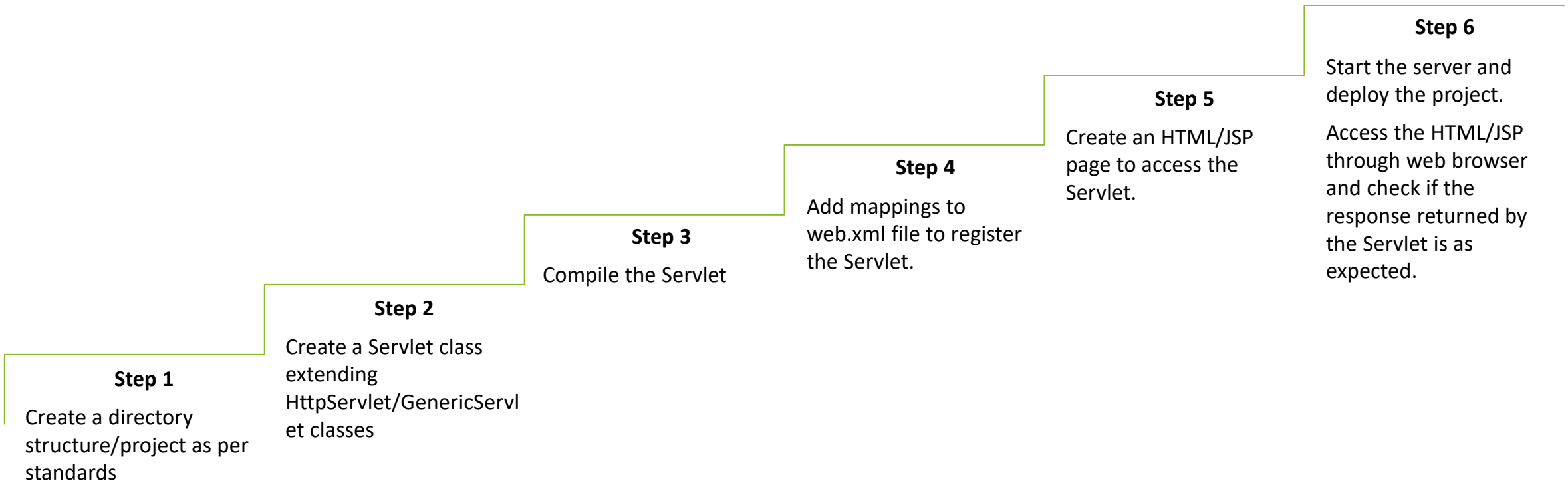
- Web server that implements Java Servlet and Java Server Pages technology converts the request into an HTTPServletRequest object.
- This object is delivered to a web component, which can interact with JavaBeans components or a database to generate dynamic content.

Web Component

- The web component can then generate an HTTPServletResponse or it can pass the request to another web component.
- Eventually a web component generates a HTTPServletResponse object.
- The web server converts this object to an HTTP response and returns it to the client.

Web Applications

High Level steps involved in implementing Web App



Knowledge Check

Q1. Servlets are used to program which component in a web application?

- Client
- Server
- Tomcat
- Applet

Q2. What type of protocol is HTTP?

- Stateless
- Stateful
- Transfer protocol
- Information protocol

Q3. A program that accepts requests from a Web browser and sends back results in the form of HTML documents, is known as

- Web host
- Web server
- Web interface
- Web application

Q4. Communication between application program and database server, takes place through:

- ODBC
- JDBC
- HTTP
- ODBC or JDBC

Servlets

Servlets

Objectives

01 **Server side programming - Overview**

02 **Introduction to Servlets**

03 **Web Terminology**

04 **Servlet API & Interface**

05 **GenericServlet & HttpServlet**

06 **HttpServlet Life Cycle**

Servlets

Server-side Programming - Overview

1. Server-side programming is very useful because it allows us to efficiently deliver information tailored for individual users and thereby create a much better user experience.

2. Server-side programming allows us to instead store the information in a database and dynamically construct and return HTML and other types of files (e.g. PDFs, images, etc.).
   - It is also possible to simply return data (JSON, XML, etc.) for rendering by appropriate client-side web frameworks (this reduces the processing burden on the server and the amount of data that needs to be

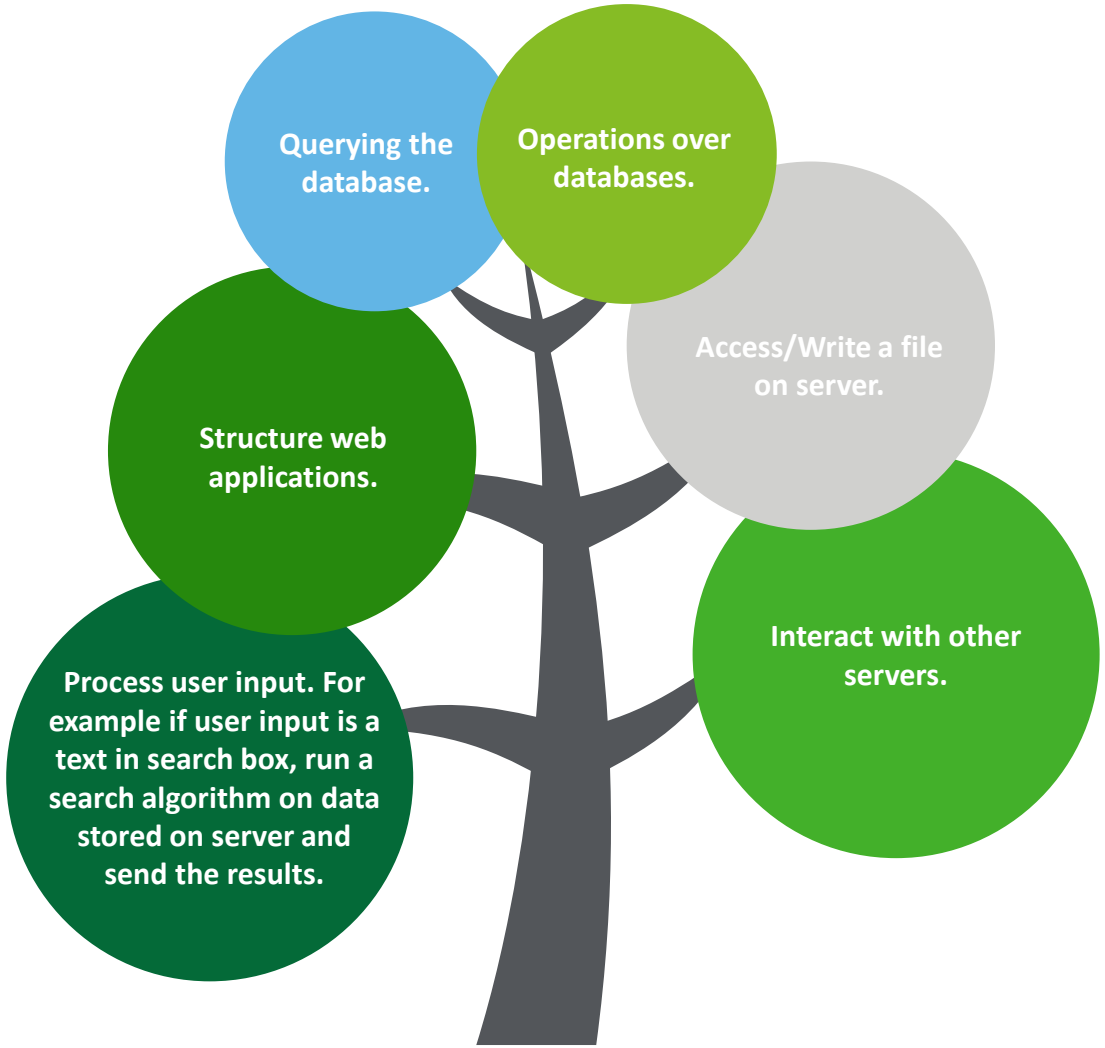
3. The server is not limited to sending information from databases, and might alternatively return the result of software tools, or data from communications services.
   - The content can even be targeted for the type of client device that is receiving it.

4. Servers can store and use information about clients to provide a convenient and tailored user experience. For example, many sites store credit cards so that details don't have to be entered again.
   - Sites like Google Maps can use saved or current locations for providing routing information, and search or travel history to highlight local businesses in search results.

5. Server-side programming allows developers to make use of sessions — basically, a mechanism that allows a server to store information on the current user of a site and send different responses based on that information.

Servlets

Features of Server-side Programming

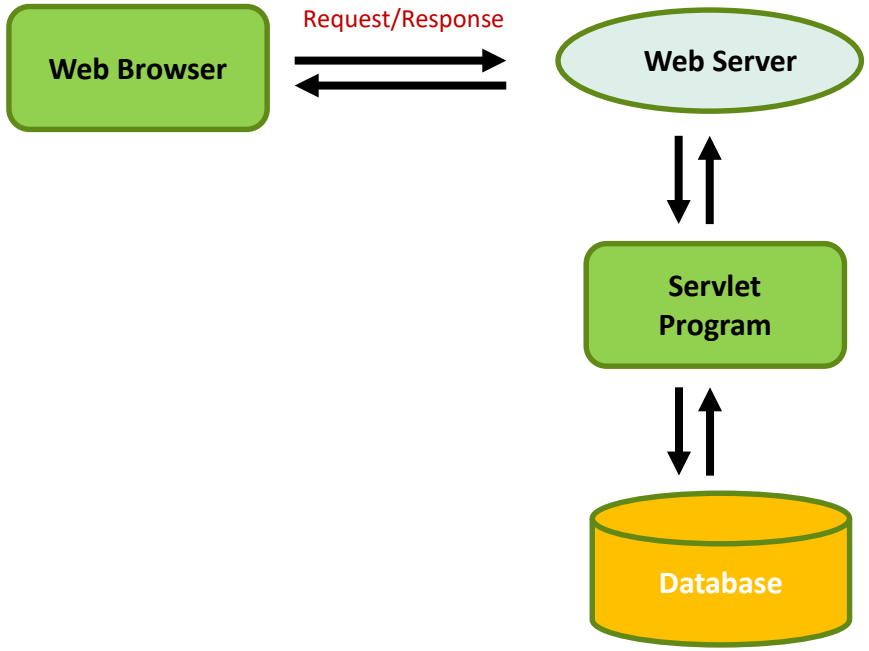


Servlets

Introduction to Servlets

- Servlets are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web server.
- Servlets work on the server-side.
- Execution of Servlets involves six basic steps:
  - The clients send the request to the web server.
  - The web server receives the request.
  - The web server passes the request to the corresponding servlet.
  - The servlet processes the request and generates the response in the form of output.
  - The servlet sends the response back to the web server.
  - The web server sends the response back to the client and the client browser displays it on the screen.

Servlet Architecture:



Servlets

Web Terminology

Term	Details
✓ Web Server	• A web server is a computer that runs websites. It's a computer program that distributes web pages as they are requisitioned. • The basic objective of the web server is to store, process and deliver web pages to the users.
✓ Static and Dynamic Website	• A static website contains web pages with fixed content. • A Dynamic website can have dynamic information based on the user entered data.
✓ URL	• A URL(Uniform Resource Locator) incorporates the domain name, along with other detailed information, to create a complete address (or “web address”) to direct a browser to a specific page online called a web page
✓ Request ✓ Response	• Request is the data passed from the front end(browser) to the web server for processing. • Response is the data passed from the web server to front end after processing of the request.
✓ Session	• In computer systems, a user session begins when a user logs in to or accesses a particular computer, network, or software service and ends after a certain time period or when the user logs out.
✓ HTTP	• HTTP(HyperText Transfer Protocol) is the protocol used to transfer data over the web. • It is part of the Internet protocol suite and defines commands and services used for transmitting webpage data.

Servlets

Servlet API

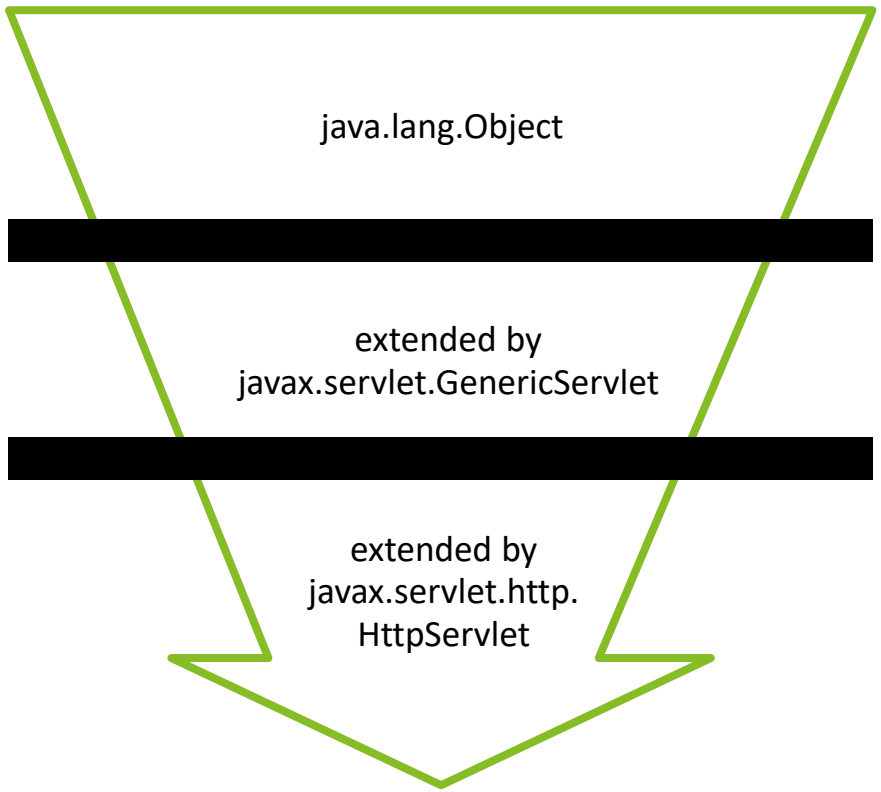
You need to use Servlet API to create servlets. There are two packages that you must remember while using API:

- the javax.servlet package that contains the classes to support generic servlet (protocol-independent servlet)
- the javax.servlet.http package that contains classes to support http servlet.

There are many classes in javax.servlet.http package. They are as follows:

- HttpServlet
- Cookie
- HttpServletRequestWrapper
- HttpServletResponseWrapper
- HttpSessionEvent
- HttpSessionBindingEvent
- HttpUtils (deprecated now)

Package Hierarchy:



Servlets

Servlet Interface

- Servlet interface provides common behavior to all the servlets. Servlet interface defines methods that all servlets must implement.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides **3 life cycle methods** that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.
- There are 5 methods in Servlet interface:
  - public void init(ServletConfig config)
  - public void service(ServletRequest request,ServletResponse response)
  - public void destroy()
  - public ServletConfig getServletConfig()
  - public String getServletInfo()

The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Example:

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

public class First implements Servlet{
ServletConfig config=null;

public void init(ServletConfig config){
    this.config=config;
    System.out.println("servlet is initialized");
}

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 simple servlet</b>");
    out.print("</body></html>");

}
public void destroy(){System.out.println("servlet is destroyed");}
public ServletConfig getServletConfig(){return config;}
public String getServletInfo(){return "Servlet info";}

}
```

Servlets

Servlet Interfaces

public void init(ServletConfig config)	• Initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	• Provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	• Is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	• Returns the object of ServletConfig.
public String getServletInfo()	• Returns information about servlet such as writer, copyright, version etc.

Servlets

Classes: GenericServlet

- **GenericServlet class**
  - Implements Servlet, ServletConfig and Serializable interfaces.
  - It provides the implementation of all the methods of these interfaces except the service method.
  - Can handle any type of request so it is protocol-independent.
- **Pros of using GenericServlet**:
  - Generic Servlet is easier to write
  - Has simple lifecycle methods
  - To write Generic Servlet you just need to extend javax.servlet.GenericServlet and override the service() method.
- **Cons of using GenericServlet**:
  - Working with Generic Servlet is not that easy because we don't have convenience methods such as doGet(), doPost(), doHead() etc in Generic Servlet that we can use in Http Servlet.

Example:

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

public class First extends GenericServlet{
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 generic servlet</b>");
out.print("</body></html>");

}
}
```

Servlets

Classes: HttpServlet

- Unlike Generic Servlet, the HTTP Servlet doesn't override the service() method. Instead it overrides the **doGet()** method or **doPost()** method or both.
- The **doGet() method** is used for getting the information from server while the **doPost() method** is used for sending information to the server.
- HttpServlet class is **protocol-dependent**.
- In Http Servlet there is no need to override the service() method because this method dispatches the Http Requests to the correct method handler,
- for example if it receives HTTP GET Request it dispatches the request to the doGet() method.

Example:

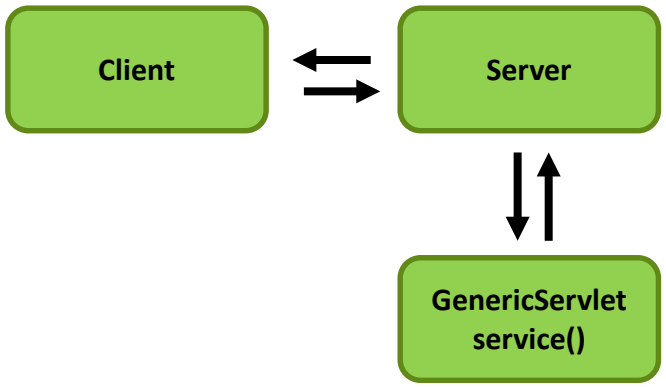
```
// Import required java libraries
import java.io.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

   private String message;

   public void init() throws ServletException {
      // Do required initialization
      message = "Hello World";
   }

   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      // Set response content type
      response.setContentType("text/html");

      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
    }

   public void destroy() {
      // do nothing.
   }
}
```

Servlets

Process: GenericServlet & HttpServlet

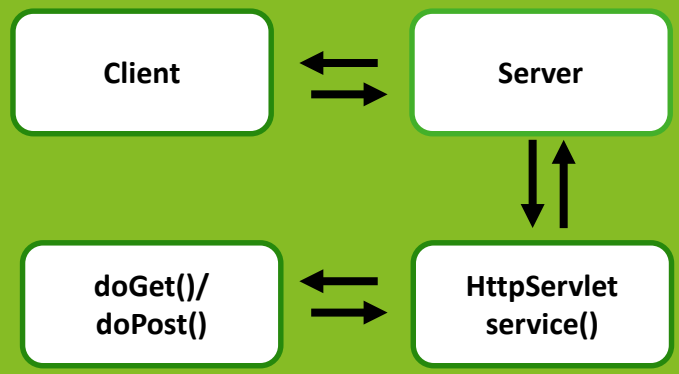
GenericServlet Process:

- Client sends request to the Server.
- Server calls the GenericServlet service() method. Service() method processes the request and sends the response object to the server.
- Server sends response to Client for user to see the data.



HttpServlet Process:

- Client sends response to the Server.
- Server calls the HttpServlet service() method. Service() method then dispatches the request to doGet() or doPost() method according to the request type.
- Server sends response to Client for user to see the data.

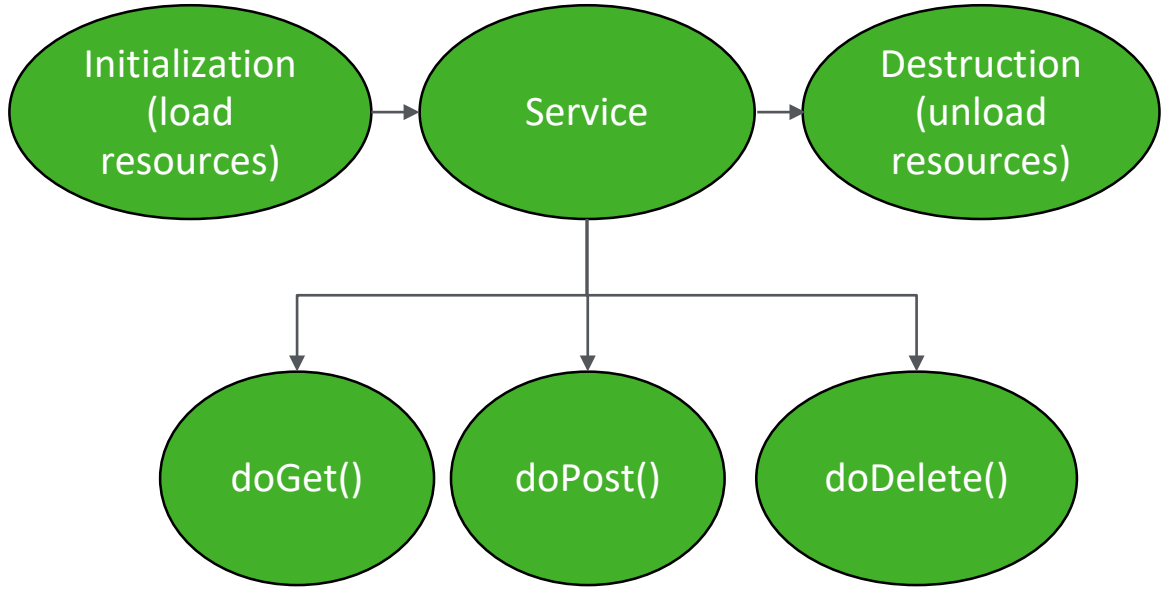


Servlets

HttpServlet Life Cycle

- The life cycle begins as soon as it is called by the Web sever to load into the container.
- It has a three-phase life: instantiation and initialization, service, and destruction.
- **Instantiation and Initialization Phase:**
  - In the *initialization phase* - the servlet container starts by locating servlet classes either in the local file system or from a remote location.
  - The classes critical to start application must be located at startup. Initialization process begins after servlet objects are instantiated.
  - It includes reading persistent configuration data, JDBC connection parameters. The container invokes the init() method of the Servlet interface for the purpose of initialization.
- **Service Phase:**
  - The service phase represents the interaction with the request until the duration of the service is actually destroyed.
  - The request and response are mapped to the service method, which in turn delegates the handle to the doXXX() methods.
  - The request and responses are mapped to the objects of ServletRequest and ServletResponse, respectively.

Flow Diagram:



Servlets

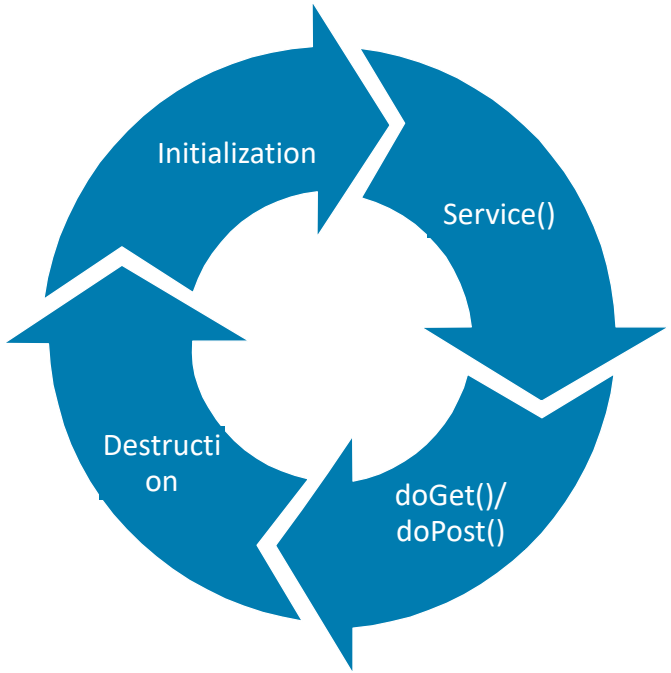
HttpServlet Life Cycle

- **Destruction Phase:**
  - In this phase, the servlet is removed from the container. The container invokes the destroy method of the Servlet interface.
  - This method ensures that the servlet is terminated gracefully and cleans up all the resources used or created during its life cycle.
- **Conclusion:**
  - The proper use of life cycle phases—

    - Initialization
    - Service
    - Destruction

    ensure that the servlet manages the application resources efficiently.
  - **During initialization**, the servlet loads up all resources it needs to process requests in the later phase.
  - **The service phase** uses these resources to respond as per requests made and then
  - **In the destruction phase**, the memory is freed up with the cleanup process.

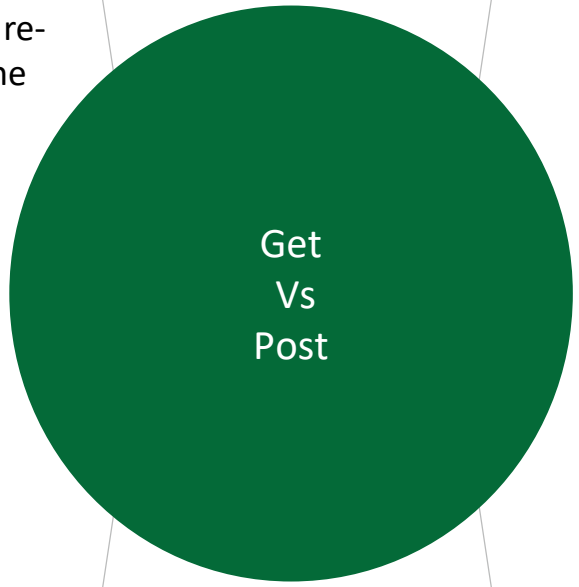
Flow Diagram:



Servlets

Http request methods GET vs POST

- Parameters remain in browser history because they are part of the URL
- Can be bookmarked
- GET requests are re-executed but may not be re-submitted to server if the HTML is stored in the browser cache
- Can send but the parameter data is limited to what we can stuff into the request line (URL). Safest use less than 2K of parameters, some servers handle up to 64K
- Only ASCII characters allowed
- GET is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in plaintext
- GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send
- Can be cached



- Parameters are not saved in browser history
- Cannot be bookmarked
- The browser usually alerts the user that data will need to be re-submitted
- Can send parameters, including uploading files, to the server
- No restrictions. Binary data is also allowed
- POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
- POST method variables are not displayed in the URL
- Not cached

Servlets

Servlet Example : Steps to create a Servlet

Steps to create a servlet are as follows:

- Create a class.
- Extend the **HttpServlet/GenericServlet** class according to the requirement.
- Override the **init(), service(), doGet(), doPost(), destroy()** method if required.
- In the service we will get the HttpRequest object from the front end.
- The service() method gets the particular objects from the request object and performs different operations and will call the DAO methods for database operations.
- After the DAO method returns the results, the Servlet will set the result in the HttpResponse object for users to see.

Here is an example of a servlet for saving user input data into the database.

***Problem Statement**: It is first fetching the parameters like name, password, email, country and calling the DAO method save() and setting the response object with appropriate response.*

Example:

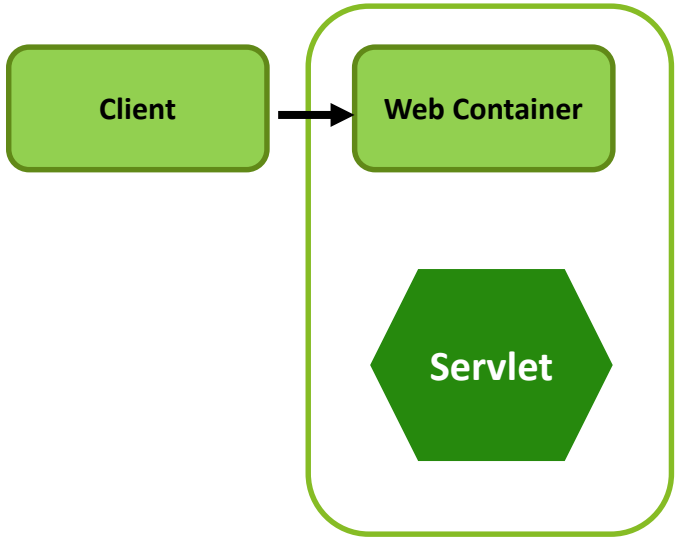
```
public class SaveServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        String name=request.getParameter("name");
        String password=request.getParameter("password");
        String email=request.getParameter("email");
        String country=request.getParameter("country");

        Emp e=new Emp();
        e.setName(name);
        e.setPassword(password);
        e.setEmail(email);
        e.setCountry(country);

        int status=EmpDao.save(e);
        if(status>0){
            out.print("<p>Record saved successfully!</p>");
            request.getRequestDispatcher("index.html").include(request, response);
        }else{
            out.println("Sorry! unable to save record");
        }

        out.close();
    }
```

Servlets

How a servlet works

Step 1

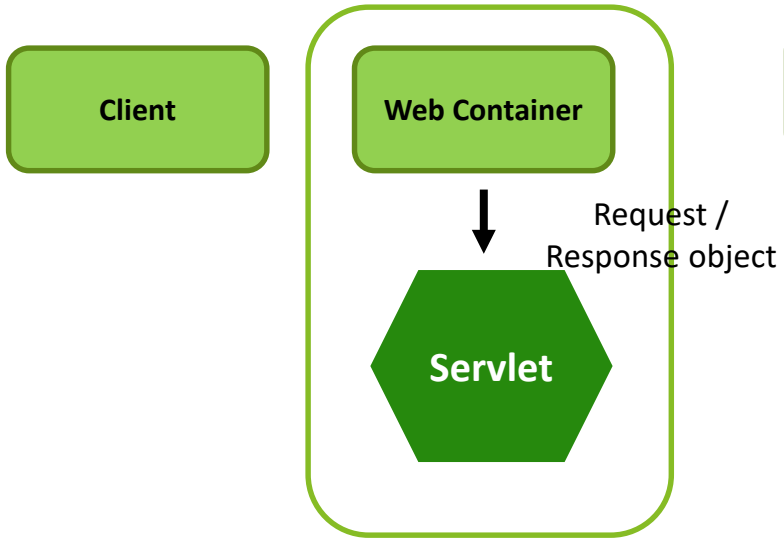
User sends request for a servlet by clicking a link that has the URL to the servlet.



Step 2

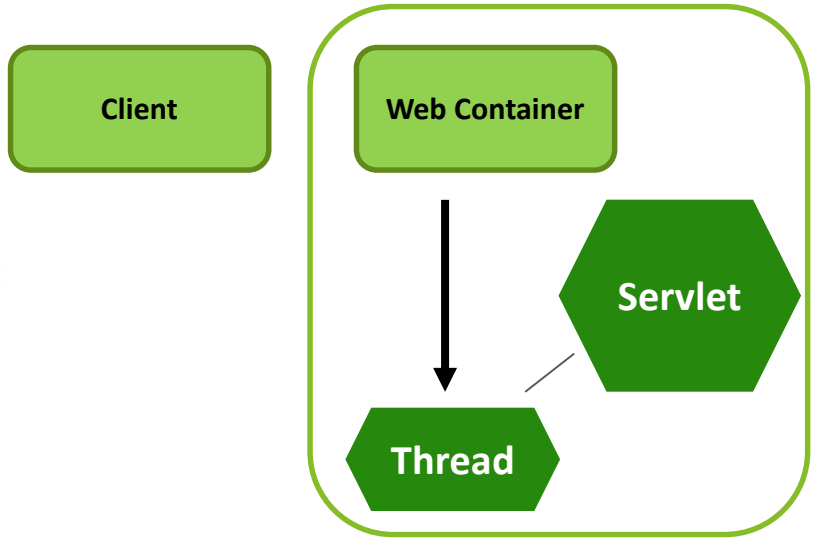
The container finds the servlet using deployment descriptor and creates two objects :

- HttpServletRequest
- HttpServletResponse



Step 3

Then the container creates or allocates a thread for that request and calls the Servlet's service() method and passes the request, response objects as arguments.

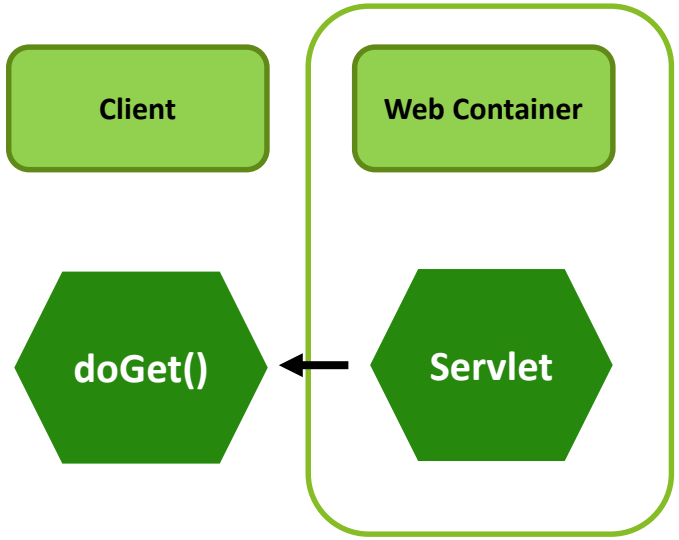


Servlets

How a servlet works

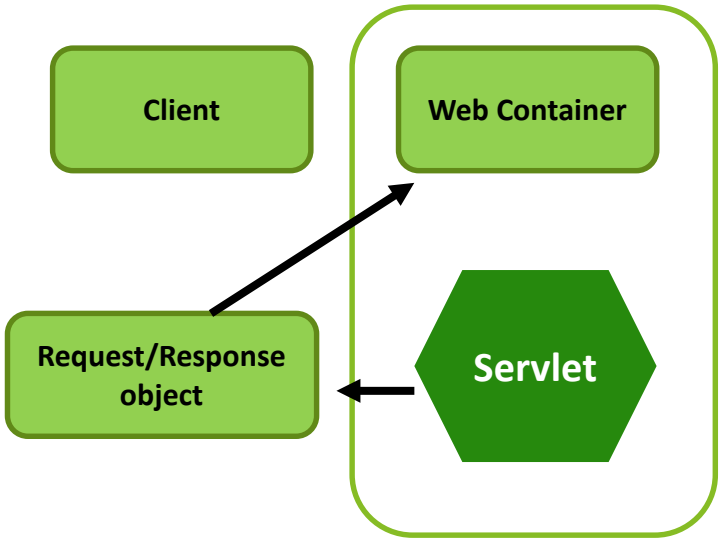
Step 4

The service() method, then decides which servlet method, doGet() or doPost() to call, based on HTTP Request Method(Get, Post etc) sent by the client.



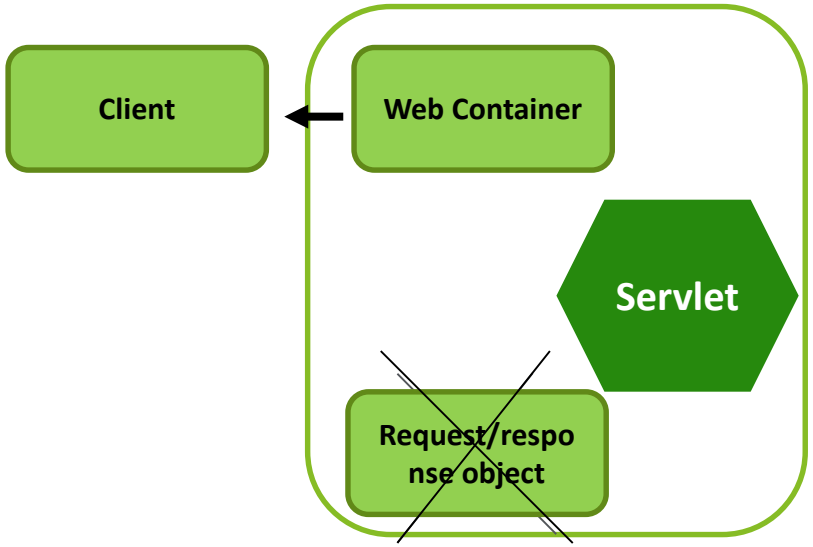
Step 5

Then the Servlet uses response object to write the response back to the client.



Step 6

After the service() method is completed the thread dies. And the request and response objects are ready for garbage collection.



Break – 15 min

Servlet Collaboration

Servlet Collaboration

Objectives

01 **RequestDispatcher**

02 **SendRedirect**

03 **Example**

04 **Knowledge Check**

Servlet Collaboration

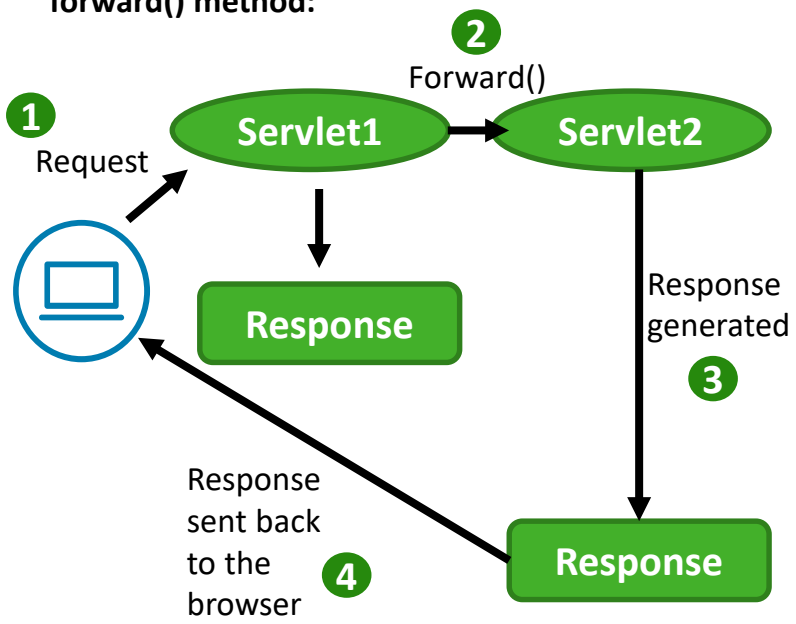
RequestDispatcher

- ✓ The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.
- ✓ This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.
- ✓ public void forward(ServletRequest request,ServletResponse response) :Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- ✓ public void include(ServletRequest request,ServletResponse response) :Includes the content of a resource (servlet, JSP page, or HTML file) in the response.
- ✓ Both methods throw IOException.
- ✓ The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

Servlet Collaboration

RequestDispatcher

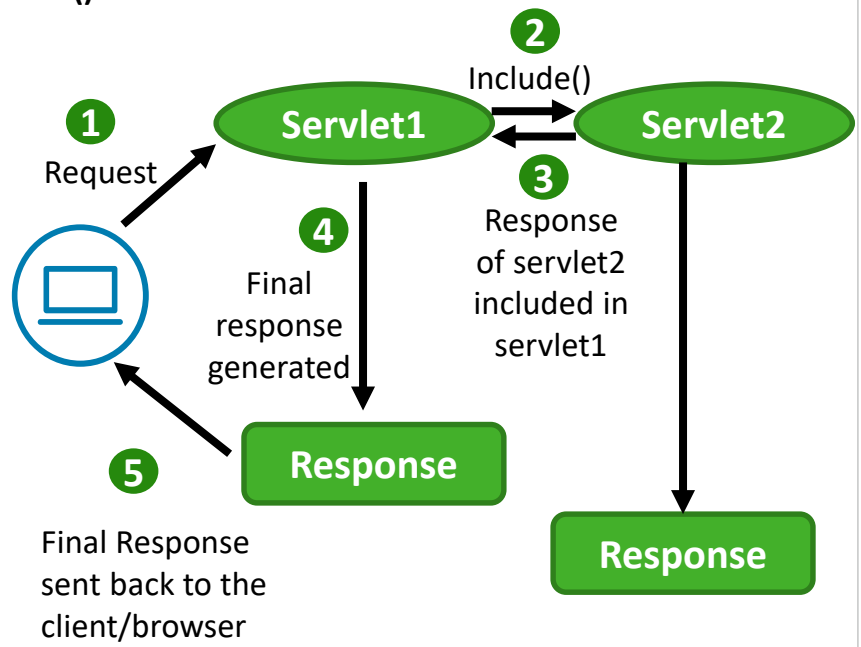
forward() method:



- Response of second servlet is sent to the client.
- Response of the first servlet is not displayed to the user.

Forward()
Vs
Include()

include() method:



- Response of second servlet is included in the response of the first servlet that is being sent to the client.

Servlet Collaboration

forward() vs sendRedirect()

- The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file.
- It accepts relative as well as absolute URL.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

forward()	sendRedirect()
• 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");

Servlet Collaboration

Example

The below Login class extends HttpServlet class and shows implementation of both forward() and include() method:

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


public class Login extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String n=request.getParameter("userName");
    String p=request.getParameter("userPass");

    if(p.equals("servlet")){
        RequestDispatcher rd=request.getRequestDispatcher("servlet2");
        rd.forward(request, response);
    }
    else{
        out.print("Sorry UserName or Password Error!");
        RequestDispatcher rd=request.getRequestDispatcher("/index.html");
        rd.include(request, response);

        }
    }

}
```

Knowledge Check

Q1. Which of the following code is used to get an attribute in a HTTP Session object in servlets?

- session.getAttribute(String name)
- session.alterAttribute(String name)
- session.updateAttribute(String name)
- session.setAttribute(String name)

Q2. When destroy() method of a filter is called?

- only once at the end of the life cycle of a filter
- after the filter has executed doFilter method
- only once at the begining of the life cycle of a filter
- after the filter has executed

Q3. Which of the following is true about servlets?

- Servlets execute within the address space of web server
- Servlets are platform-independent because they are written in java
- Servlets can use the full functionality of the Java class libraries
- Servlets execute within the address space of web server, platform independent and uses the functionality of java class libraries

Q4. Which of the following is true about init() method of servlet?

- The init() method simply creates or loads some data that will be used throughout the life of the servlet.
- The init() method is not called again and again for each user request.
- Both of the above.
- None of the above.

Knowledge Check

Q5. Which of the following is true about HTTP Get method?

- The GET method sends the encoded user information appended to the page request.
- The GET method is the defualt method to pass information from browser to web server.
- Both of the above.
- None of the above.

Q6. Which of the following is the correct order of filter life cycle phase methods?

- init(), service(), destroy()
- initialize(), service(), destroy()
- init(), doFilter(), destroy()
- init(), service(), delete()

Q7. Which of the following code is used to get a HTTP Session object in servlets?

- request.getSession()
- response.getSession();
- new Session()
- None of the above.

Q8. How do we send data to get method?

- We cannot
- Through URL
- Through payload
- None of these

Servlet Collaboration

Hands on Session

Create a Servlet and add logic to check if the user name and password entered by the user is correct or not. If correct, redirect the user to a Home page using below methods:

- **forward()**
- **sendRedirect()**

Lunch Break – 45 min.

Servlet – Hands on Session

Servlet – Hands on Session

Objectives

01 **Servlet Registration Example**

02 **Servlet CRUD Example**

03 **Key Take Away**

04 **Knowledge Check**

Servlet – Hands on Session

Servlet - *web.xml* Configuration

Via web.xml:

- The most common way to register a servlet is to add it to your web.xml file.
- As you can see, this involves two steps:
  - adding our servlet to the servlet tag, making sure to also specify the source path to the class the servlet resides within,
  - specifying the URL path the servlet will be exposed on in the url-pattern tag.
- The web.xml file is usually found in WebContent/WEB-INF.

Example:

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<servlet>
    <servlet-name>Example</servlet-name>
    <servlet-class>com.ExampleServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ExampleServlet</servlet-name>
    <url-pattern>/ExampleServlet</url-pattern>
</servlet-mapping>
```

Servlet – Hands on Session

Servlet – Annotation Configuration (contd..)

Via Annotations:

- Now let's register our servlet using the @WebServlet annotation on our custom servlet class.
- This eliminates the need for servlet mappings in the server.xml and registration of the servlet in web.xml:
- The code demonstrates how to add that annotation directly to a servlet.
- The servlet will still be available at the same URL path as before.

Example:

```
import java.io.IOException;


@WebServlet(
          name = "AnnotationExample",
          description = "Example Servlet Using Annotations",
          urlPatterns = {"/AnnotationExample"}
        )
public class ExampleServlet extends HttpServlet {

    @Override
    protected void doGet(
      HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<p>Hello World!</p>");
    }
}
```

Servlet – Hands on Session

Servlet CRUD Example

Use Case:

- ❑ To create, update, delete and read a user in an application.

Steps:

- Create a table in database to store user information.
- Create an html/jsp page to get user input from screen.
- Create a DAO class to perform database operations like create/delete/update.
- Create Servlets for each operation to read request from client and send appropriate response to client.

HTML File:

```
<form action="SaveServlet" method="post">
<table>
<tr><td>Name:</td><td><input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td><input type="password" name="password"/></td></tr>
<tr><td>Email:</td><td><input type="email" name="email"/></td></tr>
<tr><td>Country:</td><td>
<select name="country" style="width:150px">
<option>India</option>
<option>USA</option>
<option>UK</option>
<option>Other</option>
</select>
</td></tr>
<tr><td colspan="2"><input type="submit" value="Save Employee"/></td></tr>
</table>
</form>

<br/>
<a href="ViewServlet">view employees</a>
```

Servlets – Hands on Session

Servlet CRUD Example (contd..)

POJO File:

- Create a POJO class with all getter setter methods of the defined variables.

Emp POJO

```
public class Emp {
        private int id;
        private String name,
        private String password;
        private String email;
        private String country;
}
```

DAO File:

- Create a DAO class to perform all CRUD operations. An example of save is shown

```
public class EmpDao {
  public static int save(Emp e){
    try{ Connection con=EmpDao.getConnection();
      PreparedStatement ps=con.prepareStatement(
            "insert into user(name,password,email,country) values (?,?,?,?)");
      ps.setString(1,e.getName());
      ps.setString(2,e.getPassword());  ps.setString(3,e.getEmail());
      ps.setString(4,e.getCountry()); ps.executeUpdate();
      con.close();  }catch(Exception ex){ex.printStackTrace();}
    return status;  }  }
```

Servlets – Hands on Session

Servlet CRUD Example

(contd..)

Edit Servlet:

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


@WebServlet("/EditServlet")
public class EditServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out=response.getWriter();
      String sid=request.getParameter("id");
      int id=Integer.parseInt(sid);
      String name=request.getParameter("name");
```

Contd..

```
      String password=request.getParameter("password");
          String email=request.getParameter("email");
          String country=request.getParameter("country");

          Emp e=new Emp();
          e.setId(id);
          e.setName(name);
          e.setPassword(password);
          e.setEmail(email);
          e.setCountry(country);
          int status=EmpDao.update(e);
          if(status>0){
             response.sendRedirect("ViewServlet");
          }else{
             out.println("Sorry! unable to update record");
          }
          out.close();
      }
   }
```

Servlets – Hands on Session

Servlet CRUD Example

(contd..)

View Servlet:

```
import java.io.*

import javax.servlet.*

@WebServlet("/ViewServlet")

public class ViewServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request,
HttpServletResponse response)

            throws ServletException, IOException {

      response.setContentType("text/html");

      PrintWriter out=response.getWriter();

      out.println("<a href='index.html'>Add New
Employee</a>");

      out.println("<h1>Employees List</h1>");

      List<Emp> list=EmpDao.getAllEmployees();

      out.print("<table border='1' width='100%'");
```

Contd..

```
out.print("<tr><th>Id</th><th>Name</th><th>Password</
th><th>Email</th><th>Country</th>

                  <th>Edit</th><th>Delete</th></tr>");
      for(Emp e:list){

      out.print("<tr><td>"+e.getId()+"</td><td>"+e.getNam
      e()+"</td><td>"+e.getPassword()+"</td>
      <td>"+e.getEmail()+"</td><td>"+e.getCountry()+"</td
      ><td><a
      href='EditServlet?id="+e.getId()+"'>edit</a></td>
            <td><a
      href='DeleteServlet?id="+e.getId()+"'>delete</a></td
      ></tr>");
      }
      out.print("</table>");
      out.close();
   }
}
```

Servlets – Hands on Session

Servlet CRUD Example

(contd..)

Delete Servlet:

```
import java.io.IOException;

import javax.servlet.*;

@WebServlet("/DeleteServlet")

public class DeleteServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request,
HttpServletResponse response)

         throws ServletException, IOException {

      String sid=request.getParameter("id");

      int id=Integer.parseInt(sid);

      EmpDao.delete(id);

      response.sendRedirect("ViewServlet");

   }

}
```

Save Servlet:

```
 import java.io.*;
import javax.servlet.*;
@WebServlet("/SaveServlet")
public class SaveServlet extends HttpServlet {
 protected void doPost(HttpServletRequest request,
HttpServletResponse response)
       throws ServletException, IOException {
      response.setContentType("text/html");
      String name=request.getParameter("name");
      String password=request.getParameter("password");
      String email=request.getParameter("email");
      String country=request.getParameter("country");
      Emp e=new Emp();
      e.setName(name);
      e.setPassword(password);
      e.setEmail(email);
      e.setCountry(country);
      int status=EmpDao.save(e);
   }
}
```

Servlets – Hands on Session

Key Take Away

Servlet Registration

There are 2 ways to register a Servlet in a web application:

- Via web.xml
- Via Annotation

Servlet implementation

2 types of classes can be extended while creating new Servlet class:

- HttpServlet
- GenericServlet

Servlet implementation

- HttpServlet class is preferred over GenericServlet because of its flexibility to dispatch different request types to respective methods.

Servlet implementation

Servlet has 3 methods:

- init() method
- service() method
- destroy() method

Knowledge Check

Q1. Web.xml is located at which folder in the project?

- WebContent
- WebContent/META-INF
- WebContent/WEB-INF
- None of the above.

Q2. Servlet-name is inside which tag in web.xml

- Servlet tag
- Servlet-mapping tag
- Servlet-properties tag
- Servlet-attributes

Q3. Which annotation is used to register servlet in java?

- @Web
- @Servlet
- @WebApplication
- @WebServlet

Q4. DAO layer is used to:

- Validate the request
- Perform database operations
- Create a servlet
- None of these

Break – 15 min

Session Tracking

Session Tracking

Objectives

01 **What is Session?**

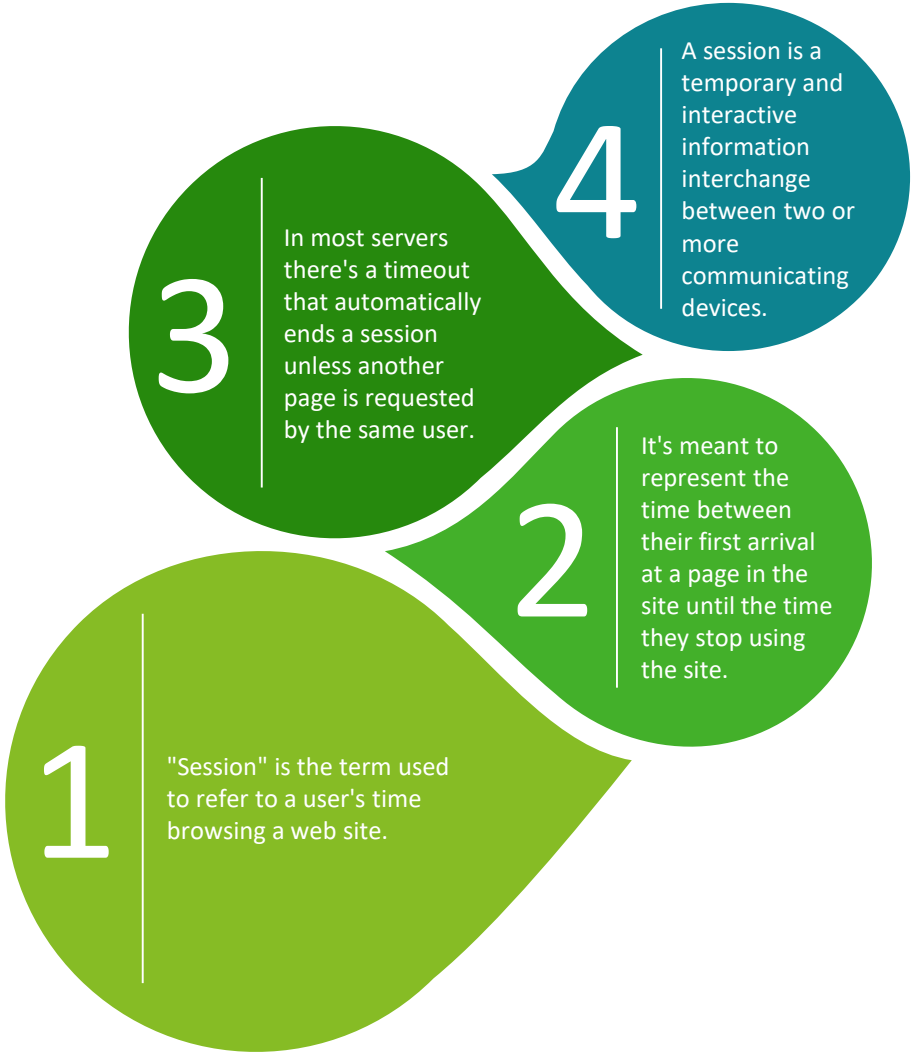
02 **Session Tracking – What & Why?**

03 **Session Techniques**

04 **Examples**

Session Tracking

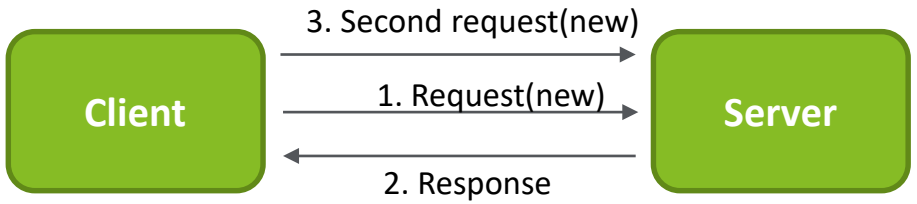
What is Session?



Session Tracking

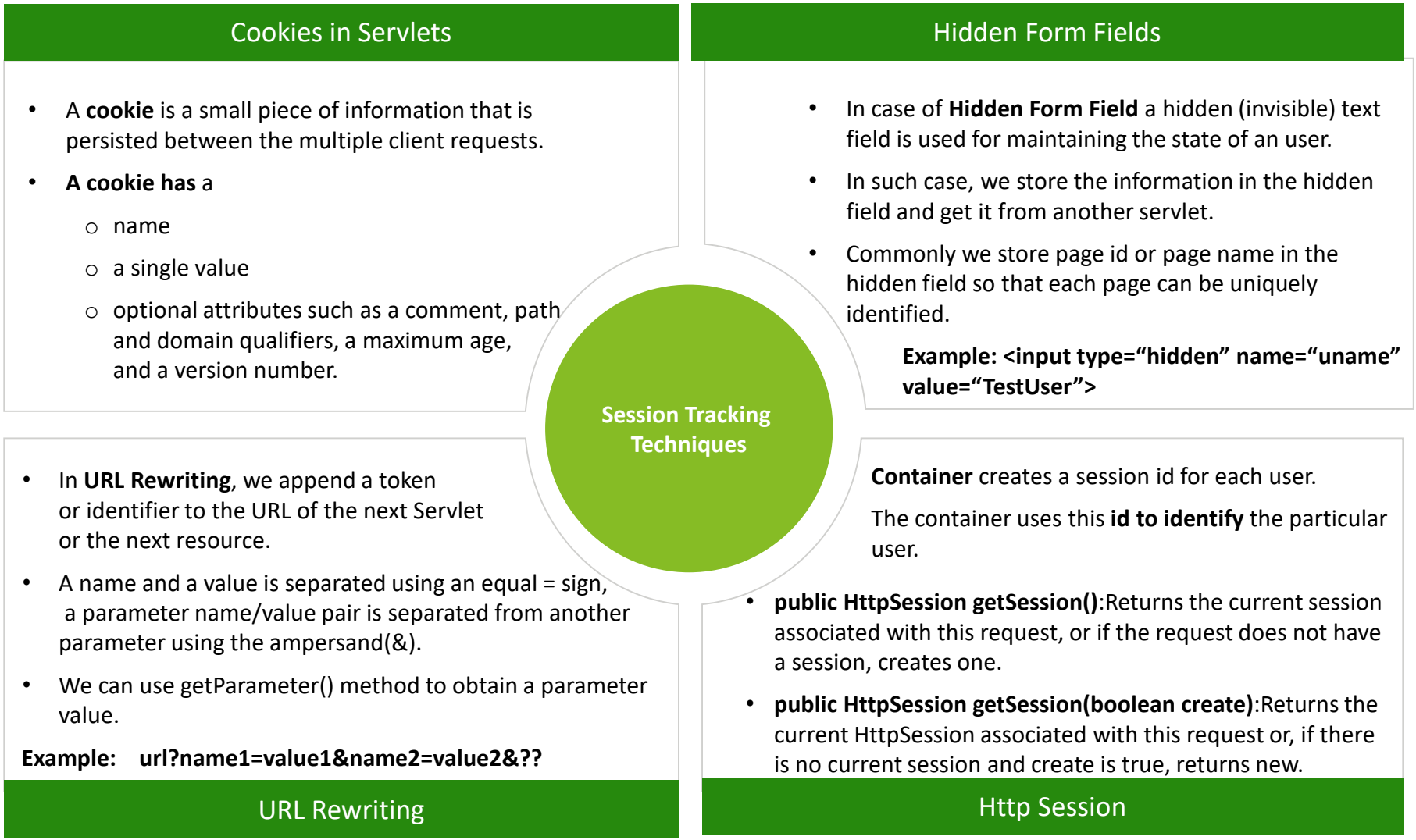
Session Tracking - What & Why?

- **Session Tracking**
  - Is a way to maintain state (data) of an user. It is also known as session management in servlet.
- **HTTP**
  - Is a "**stateless**" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
- **Why use Session Tracking?**
  - To recognize the user It is used to recognize the particular user & to show the related user specific data to the viewer.



Session Tracking

Session Tracking - Techniques

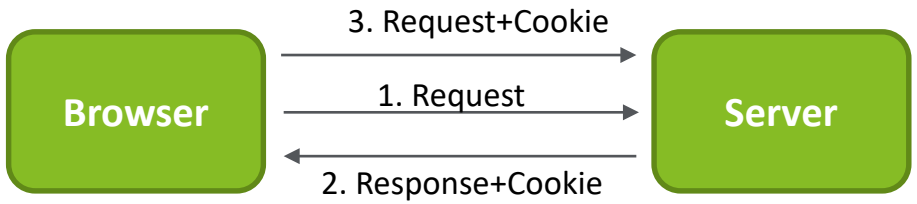


Session Tracking

Session Tracking – Techniques (contd..)

How to create a cookie

```
Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object
response.addCookie(ck);//adding cookie in the response
```




```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
}
```

```
Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

Session Tracking

Example

This example describes how to use the HttpSession object to find out the creation time and the last-accessed time for a session. We would associate a new session with the request if one does not already exist.

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import java.util.*;

import java.text.DateFormat;

public class MyServlet extends HttpServlet {

 public void doGet(HttpServletRequest request, HttpServletResponse response)

   throws ServletException, java.io.IOException {

      response.setContentType("text/html");

      java.io.PrintWriter out = response.getWriter();

      HttpSession session = request.getSession(true);

      Date creationTime = new Date(session.getCreationTime() );
```

Session Tracking

Example (contd..)

```
    Date lastAccessed = new Date(session.getLastAccessedTime());
    Date now = new Date();
    DateFormat formatter = DateFormat.getDateTimeInstance(DateFormat.MEDIUM,DateFormat.MEDIUM);


    out.println("<html>");
    out.println("<head>");
    out.println("<title>Displaying the Session Creation and Last-Accessed Time</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h2>Session Creation and Last-Accessed Time</h2>");
    out.println("The time and date now is: " + formatter.format(now) + "<br><br>");
    out.println("The session creation time: HttpSession.getCreationTime( ): " + formatter.format(creationTime) + "<br><br>");
    out.println("The last time the session was accessed: HttpSession.getLastAccessedTime( ): " + formatter.format(lastAccessed) );
    out.println("</body>");
    out.println("</html>");
  } }
```

Knowledge Check

Q1. Which of the below is not a session tracking method?

- URL rewriting
- History
- Cookies
- SSL sessions

Q2. Which of the following is stored at client side?

- URL rewriting
- Hidden form fields
- SSL sessions
- Cookies

Q3. SessionIDs are stored in cookies.

- True
- False

Q4. How can we invalidate a session?

- session.discontinue()
- session.invalidate()
- session.disconnect()
- session.falsify()

Session Tracking

Hands on Session

Register a Servlet and write an example to print session Id, session creation time and last accessed time.

Hint: HttpSession API provides a method *getId()* which returns the associated session Id, *getCreationTime()* to get the session creation time and *getLastAccessedTime()* to get session last accessed time.

Assignment of the day

Problem Statement

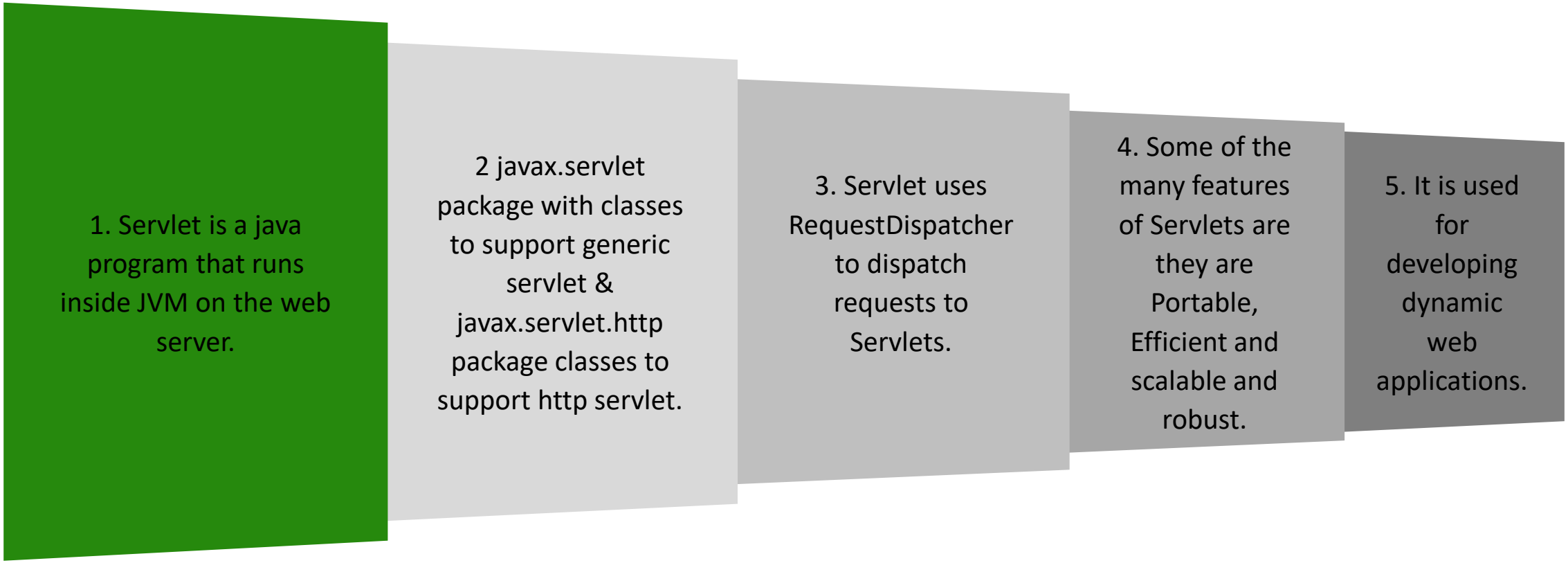
Create a web application with a user sign up/sign in/change password/deactivate user module, also handling user session.

Create the login module including jsp/html, Servlet, POJO class and DAO.

Servlets

Recap

Glimpse of Important points



Any Questions?

Thank you