Dynamic Web Application

Nada Nahle

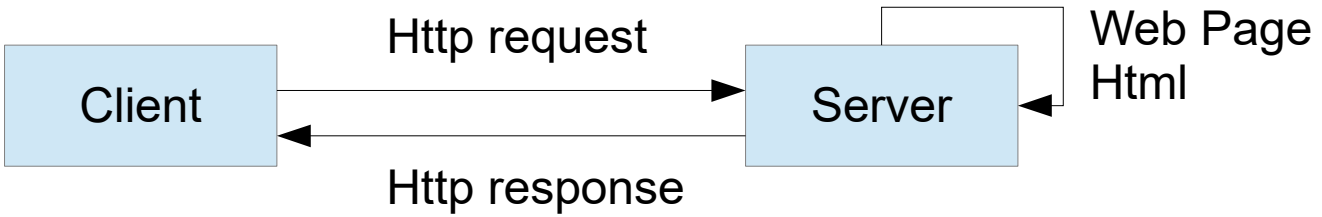
Dynamic Web Application : Overview

- A web site where the HTML pages are generated dynamically in response to user requests.

Dynamic Web Application : Overview

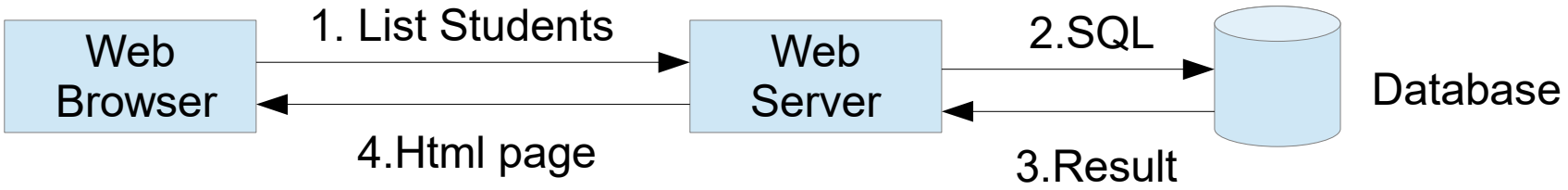
- A web site where the HTML pages are generated dynamically in response to user requests.

- Client-Server application : Communication via Http.
  - Client : Browser (Mozilla, chrome, ...)
  - Server : Web Server (Apache Tomcat, Glassfish,...)

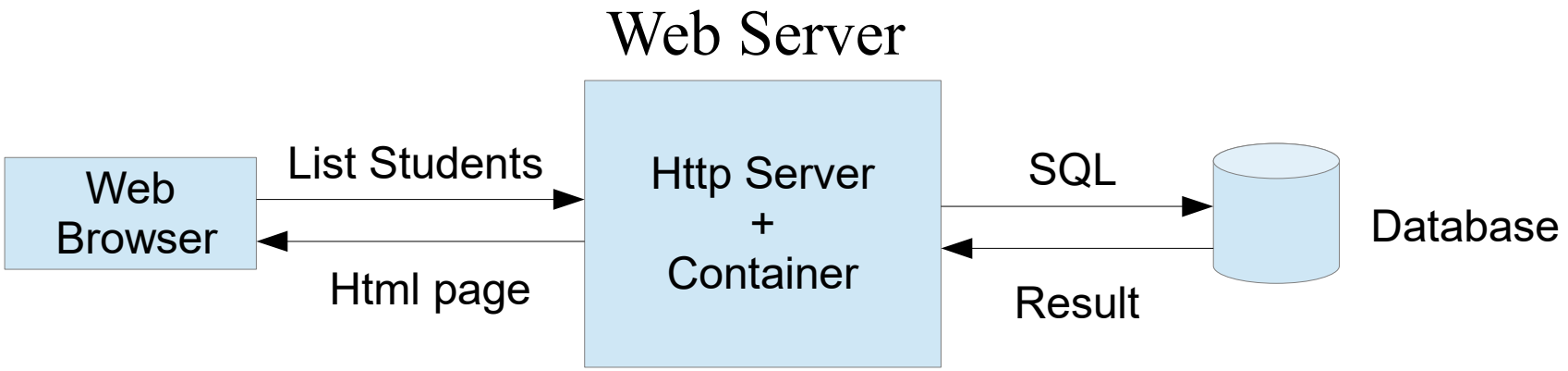


Dynamic Web Application : Example

- A client asks the server to display students.
- The Web Server gets the list of students from the database and returns it back as an html page to the browser.



Zoom to Web Server



Web Server = Http Server + Container

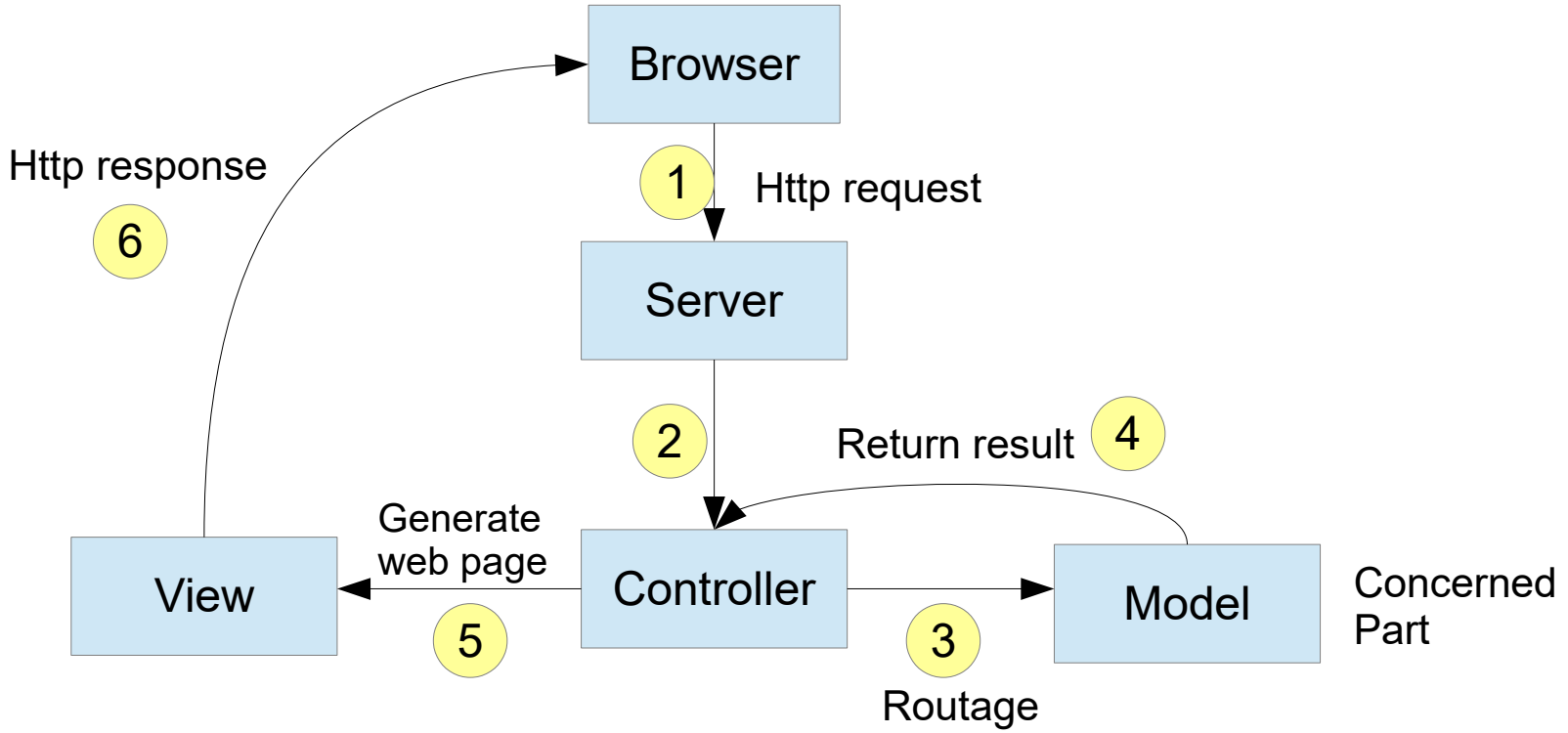
- The Http server handles the Client's requests on the corresponding http ports and send them to the container.
- The container executes the Java code.

Where is the Java Code ?

- Java EE does not impose any Java coding methodology
- Good practice : MVC Design Pattern

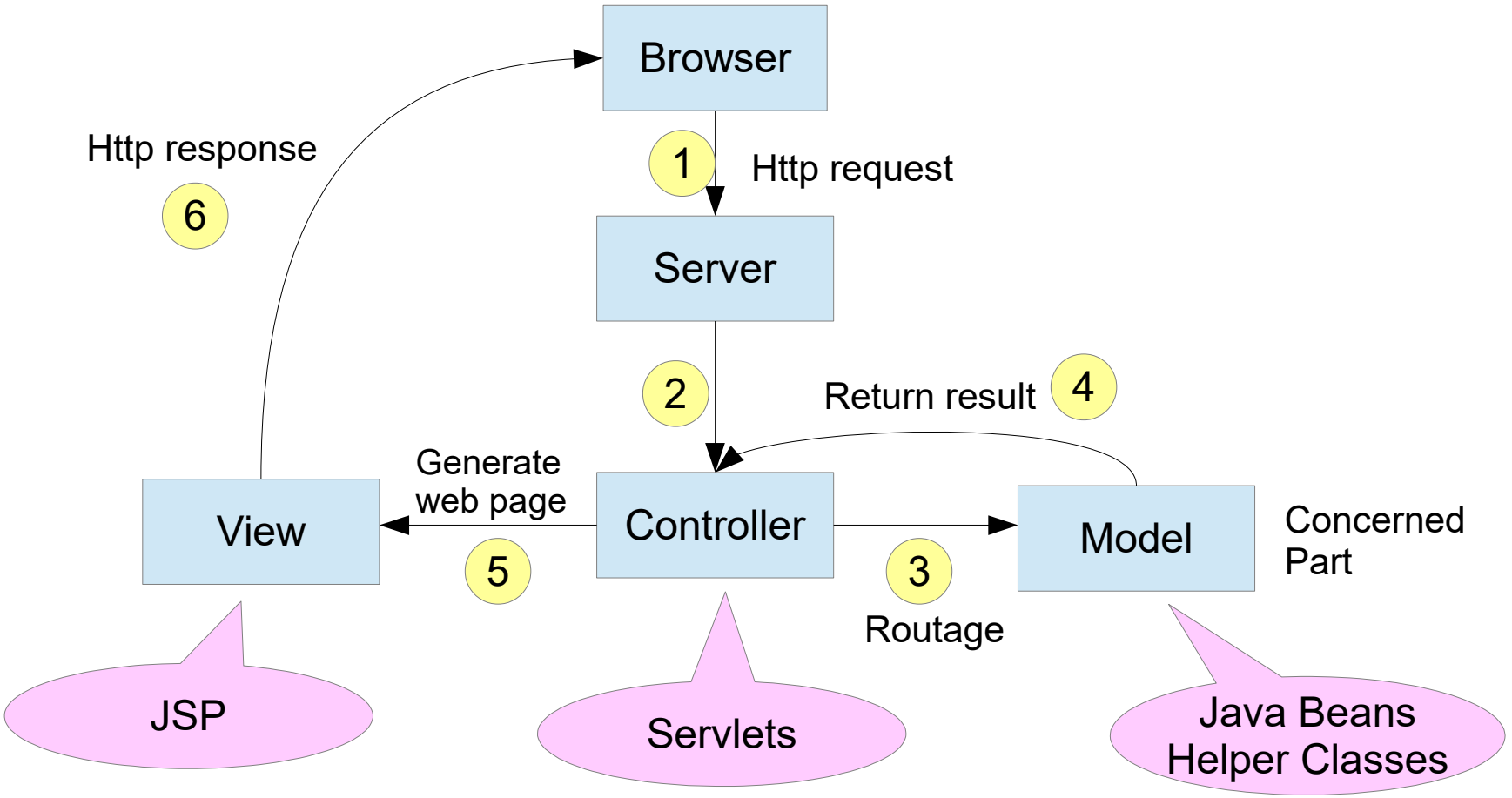
MVC Design Pattern

- Java EE does not impose any Java coding methodology
- Good practice : MVC Design Pattern



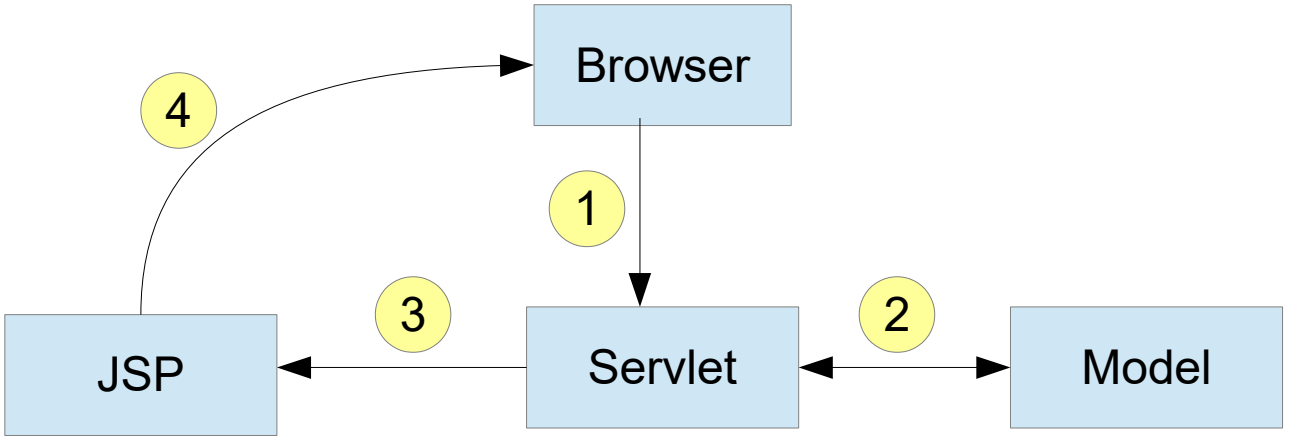
MVC Design Pattern

- Java EE does not impose any Java coding methodology
- Good practice : MVC Design Pattern



MVC Design Pattern

- Http request is handled by a Servlet.
- The Servlet communicates with the Model
- The servlet call the JSP to display the response on the browser



Where is the Java Code ?

- Servlets
- Java Server Pages (JSPs)
- JavaBeans – Helper Classes

Servlets and JSPs

- Servlets and JSPs are key components of Java EE.
- Popular MVC frameworks are built upon Servlets and JSPs.
  - Java Server Faces (JSF)
  - Spring MVC
  - Struts

Servlet Demo

```
//@WebServlet("/SampleServlet")
@WebServlet(urlPatterns= "/Bonjour")
public class SampleServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
// TODO Auto-generated method stub
String name = request.getParameter("name");

response.setContentType("text/html");
response.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body> Bonjour </body>");
out.println(name);
out.println("</html>");
}
}
```

Servlet Demo

```
//@WebServlet("/SampleServlet")
@WebServlet(urlPatterns= "/Bonjour")
public class SampleServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
// TODO Auto-generated method stub
String name = request.getParameter("name");

response.setContentType("text/html");
response.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body> Bonjour </body>");
out.println(name);
out.println("</html>");
}
}
```

Bad Practice

Servlet Demo

```
@WebServlet(urlPatterns = "/Bonjour")
public class SampleServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
// TODO Auto-generated method stub
String name = request.getParameter("name");

/*response.setContentType("text/html");
response.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body> Bonjour </body>");
out.println(name);
out.println("</html>");*/
request.getRequestDispatcher("/bonjour.jsp").forward(request,
response);
}
}
```

Servlet Demo

```
@WebServlet(urlPatterns = "/Bonjour")
public class SampleServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
// TODO Auto-generated method stub
String name = request.getParameter("name");
request.setAttribute("name", name);
request.getRequestDispatcher("/bonjour.jsp").forward(request,
response);
}
}
```

Servlet Demo

- A servlet is a class Java that extends the HttpServlet .
- Contains doGet and doPost Methods and others..
- DoGet Method is called when wen run the Servlet on server
- DoGet(HttpServletRequest , HttpServletResponse)
  - Request : request of the user + parameters
  - Response : we build it – html pages
- Can call a JSP file with RequestDispatcher
- A request can carry some attributes with request.setAttribute

JSP Demo

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Bonjour</title>
</head>
<body>
Bonjour
<% String name = (String) request.getAttribute("name");%>
<%= name %>
</body>
</html>
```

JSP Demo

- A JSP contains both HTML and Java code.
- Scriplets : <% Java Code %>
- Gets attributes from the Servlet using request.getAttribute
- Returned back to the browser as HTML page after scriplets execution.

Bad Practice

- **Mix HTML and Java Code is a bad practice**

  → Expression Language

  ${AttributeName}

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<title>Bonjour</title>
</head>
<body>
Bonjour ${name }
</body>
</html>
```

Expression Language (EL)

- With EL, we can do also:
  - Calculations : ${6*5}
  - Test on parameters : ${empty name}
  - If Condition : ${empty name ? ' ' : name}
  - And others..

Expression Language (EL)

- Good reference for EL

https://www.tutorialspoint.com/jsp/jsp_expression_language.htm

JavaBeans

- In the Model, we may use JavaBeans
  - Java classes with attributes and getters and setters
- In the controller, we can have instances of Model classes
- We can set an instance as request attribute and send it to the view.

JavaBeans

```
package com.nada.sample;

public class Student {

private String fisrtname;
private String lastName;
public String getFisrtname() {
return fisrtname;
}
public void setFisrtname(String fisrtname) {
this.fisrtname = fisrtname;
}
public String getLastName() {
return lastName;
}
public void setLastName(String lastName) {
this.lastName = lastName;
}
public Student(String fisrtname, String lastName) {
super();
this.fisrtname = fisrtname;
this.lastName = lastName;
}
@Override
public String toString() {
return "Student [fisrtname=" + fisrtname + ", lastName=" + lastName + "]";
}
}
```

We create the Student Class

JavaBeans

```
@WebServlet(urlPatterns = "/Bonjour")
public class SampleServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

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

Student student = new Student("nada","nahle");
request.setAttribute("student", student);
request.getRequestDispatcher("/bonjour.jsp").forward(request,
response);
}
}
```

We create an instance of
Student Class in the servlet
and set it as request attribute

JavaBeans

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<title>Bonjour</title>
</head>
<body>
Bonjour ${student }
</body>
</html>
```

We get the Student by EL → ToString() method in student class

JSTL (Java Standard Tag Library)

- To avoid writing Java Code in JSP.

JSTL (Java Standard Tag Library)

- To avoid writing Java Code in JSP
  - XML tags : Loops, conditions, affectation, functions,...
- 5 libraries : Core – Format – XML – SQL – Function
  - Core : Variables, loops, conditions, …

- We have to add JSLT jars to the « lib » folder in WEB-INF
- We have to add the taglib at the top of our JSP

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

JSTL (Syntax) - Examples

```
<c:forEach var="Student" items="${STUDENT_LIST }" >
<c : out value ="${Student.firstName}"/>
</c:forEach>
```

```
<c:set target="${student} property="firstName" value="Ponpon"/>
```

JSTL (Syntax) - Examples

Good reference for JSTL syntax :

https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

Servlets and JSPs

- We can develop Web Applications only with Servlets
  - Servlets can produce html output with Java Code

- We can develop Web Applications only with JSPs
  - JSP = HTML + Java

- Both of them run on the server and can return Http response.
- Both of them can contain both Business and presentation View.

Benefits of MVC

- Integrate Servlets and JSPs together.
  - Servlet does business logic
    - Minimizes Html code in Servlet
  - JSP takes care of the presentation view
    - Less Java code in JSP

doPost Method

- In a servlet, we can find « doPost » method as well as « doGet »
- Can be called from a Web Form (in a JSP page)

```
<form action="WelcomeServlet" method="post">
First Name: <input type="text" name="firstName" />
Last Name:<input type="text" name="lastName" />
<input type="submit" value="OK" />
</form>
```

- Once we press « OK», the « firstName » and « lastName » are sent as parameters to the « doPost » method of « WelcomeServlet ».

doPost Method

- The parameters do not figure in the URL but are in the body of the request.
- Can be sent as attributes to a JSP page.

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
String fistName= request.getParameter("firstName");
String lastName = request.getParameter("lastName");

request.setAttribute("fname", firstName);
request.setAttribute("lname", lastName );
request.getRequestDispatcher("/Welcome.jsp").forward(request, response);
}
```

doPost Method

- In the « welcome.jsp », we can access request attributes via EL or JSTL.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Welcome</title>
</head>
<body>
Welcome ${fname } ${lname}
</body>
</html>
```

Sessions

- A session's variable is an information that the servers should maintain as long as the visitor is on the site.
- In case of e-banking, e-commerce, tracking apps, etc...
- We can access a session variable in all the pages of the site.

Sessions-Example

- If we have a welcome page that asks for a visitor's first name and last name and that we want that the site keep this information all along the visitor's navigation, we can put the firstName and lastName in session variables.

```
<form action="WelcomeServlet" method="post">
First Name: <input type="text" name="firstName" />
Last Name:<input type="text" name="lastName" />
<input type="submit" value="OK" />
</form>
```

Sessions-Example

- In the servlet

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
String fistName= request.getParameter("firstName");
String lastName = request.getParameter("lastName");

HttpSession session = request.getSession();
Session.setAttribute("prenom", firstName);
Session.setAttribute("nom", lastName);
request.getRequestDispatcher("/Welcome.jsp").forward(request, response);
}
```

- The server deletes this information after logging out or closing the navigator or a certain timeout.
- We can free the session variable from the servlet by using :

  session.invalidate() ;

Sessions-Example

- In the JSP

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome</title>
</head>

<body>

<c:if test="${!empty sessionScope.prenom && !empty sessionScope.nom}">
<p> Welcome ${sessionScope.prenom} ${sessionScope.nom} </p>
</c:if>

</body>
</html>
```

Cookies

- A cookie is an information stored on the browser of the client for a certain period (one week, one month, one year, ...)
- In case of login information for example (helps in entering form's data)

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
String fistName= request.getParameter("firstName");
String lastName = request.getParameter("lastName");

Cookie cookie = new Cookie("prenom", firstName);
cookie.setMaxAge(60*60*24) ; // in seconds, here for 24 hours
response.addCookie(cookie) ;
request.getRequestDispatcher("/Welcome.jsp").forward(request, response);
}
```

Cookies

- When we access the site again, the cookies are retrieved by the doGet method.

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

Cookie [] cookies = request.getCookies();
if(cookies!= null){
    for(Cookie cookie:cookies){
        if(cookie.getName().equals("prenom"))
            request.setAttribute("prenom", cookie.getValue() ;

request.getRequestDispatcher("/Welcome.jsp").forward(request, response);
}
```

Cookies

- And in JSPs, we can access the cookies through EL

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome</title>
</head>

<body>
 Welcome ${prenom}
</body>

</html>
```

- We can also fill the firstName input field of the form with the « prenom » cookie to simplify data entry.