



LẬP TRÌNH JAVA 4

BÀI 2: SERVLET, REQUEST, RESPONSE, SESSION TRACKING

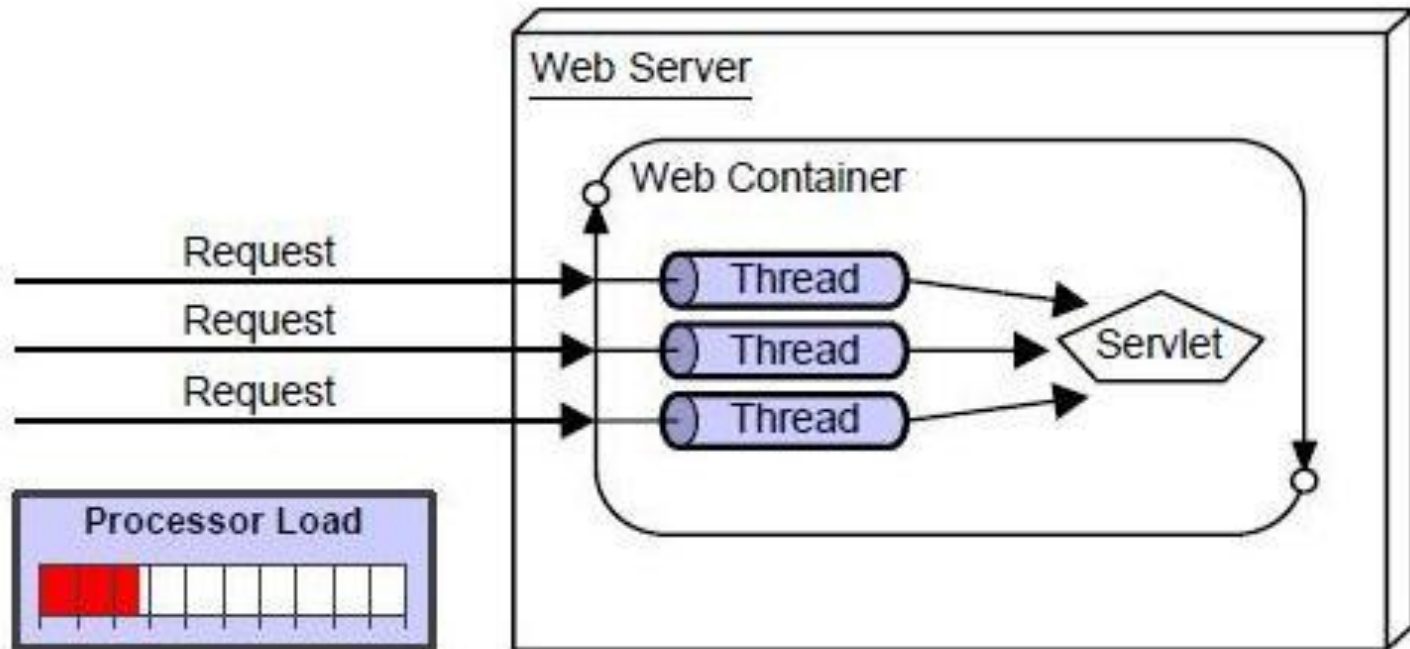
PHẦN 1

🎯 Kết thúc bài học này bạn có khả năng

- ❖ Servlet là gì?
- ❖ Servlet Scope Object
- ❖ Servlet Request
- ❖ Servlet Response
- ❖ Section Tracking

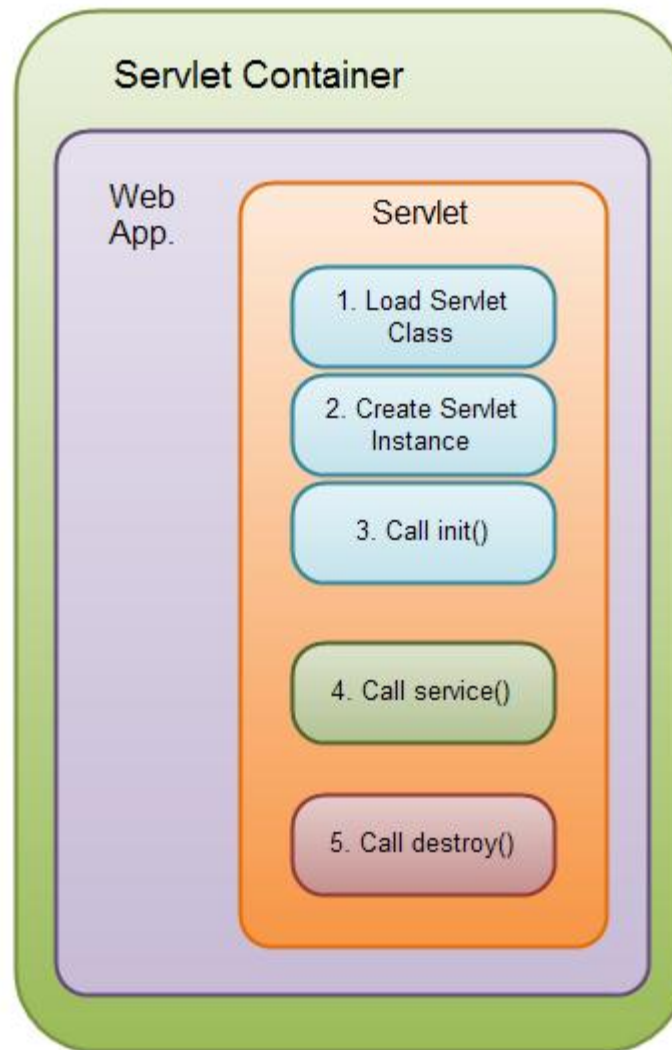


- Servlet là các đối tượng Java, mở rộng chức năng của một HTTP server, do đó được viết bằng ngôn ngữ Java
- Là những chương trình độc lập platform và chạy phía server
- Cơ chế hoạt động theo mô hình CGI mở rộng
- Chương trình servlet:
 - Thường extends class HttpServlet. Không có method main
 - Phải được dịch ra ở dạng byte-code và khai báo với web server



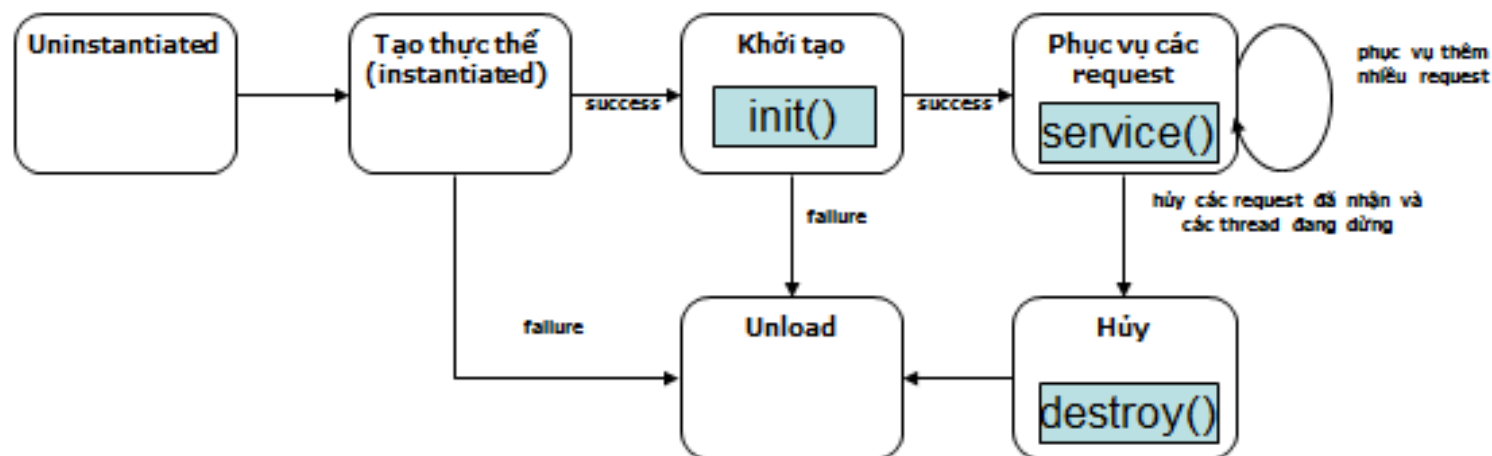
❑ Servlet có nhiều ưu điểm so với CGI.

- ❖ **Hiệu suất xử lý** tốt hơn(better performance): vì nó tạo ra một thread cho một yêu cầu chứ không phải tiến trình.
- ❖ **Khả chuyển**: bởi vì servlet được phát triển từ ngôn ngữ Java
- ❖ **Mạnh**(Robust): Servlet được quản lý bởi JVM, JVM chủ động quản lý bộ nhớ và thu thập rác.
- ❖ **An toàn**(Secure): bởi vì Java là ngôn ngữ an toàn

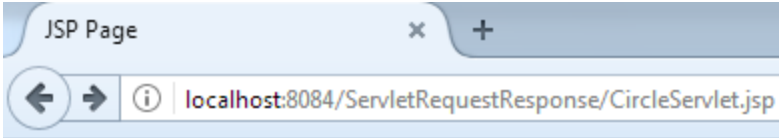


□ Có 5 bước:

- ❖ Tải Servlet Class vào bộ nhớ.
- ❖ Tạo đối tượng Servlet.
- ❖ Gọi method servlets `init()`
- ❖ Gọi method servlets `service()`.
- ❖ Gọi method servlets `destroy()`

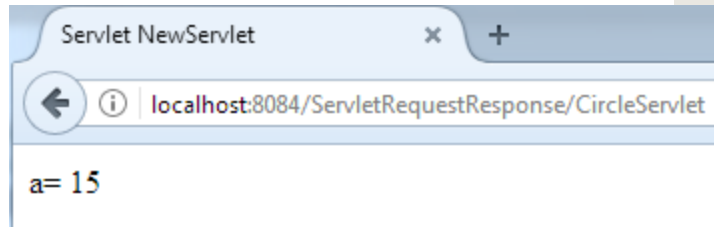


- ❑ **Bước 1, 2 và 3** được thực thi một lần duy nhất, khi mà servlet được nạp lần đầu. Mặc định các servlet không được tải lên cho tới khi nó nhận một đòi hỏi đầu tiên từ người dùng. Bạn có thể buộc ServletContainer (Bộ chứa các servlet) tải các servlet khi nó khởi động.
- ❑ **Bước 4** được thực thi nhiều lần, mỗi khi có đòi hỏi từ phía người dùng tới servlet.
- ❑ **Bước 5** nó được thực thi khi bộ chứa servlet (Servlet Container) trút bỏ (unloaded) servlet



Circle Servlet

Circle Servlet



Output - Apache Tomcat 8.0.27.0

```

29-May-2017 22:40:52.2
init
a= 5
29-May-2017 22:45:01.4
29-May-2017 22:45:01.4
29-May-2017 22:45:01.5
29-May-2017 22:45:01.6
29-May-2017 22:45:01.7
29-May-2017 22:45:01.7
29-May-2017 22:45:01.7
29-May-2017 22:45:01.7
Destroy is invoked
    
```

```

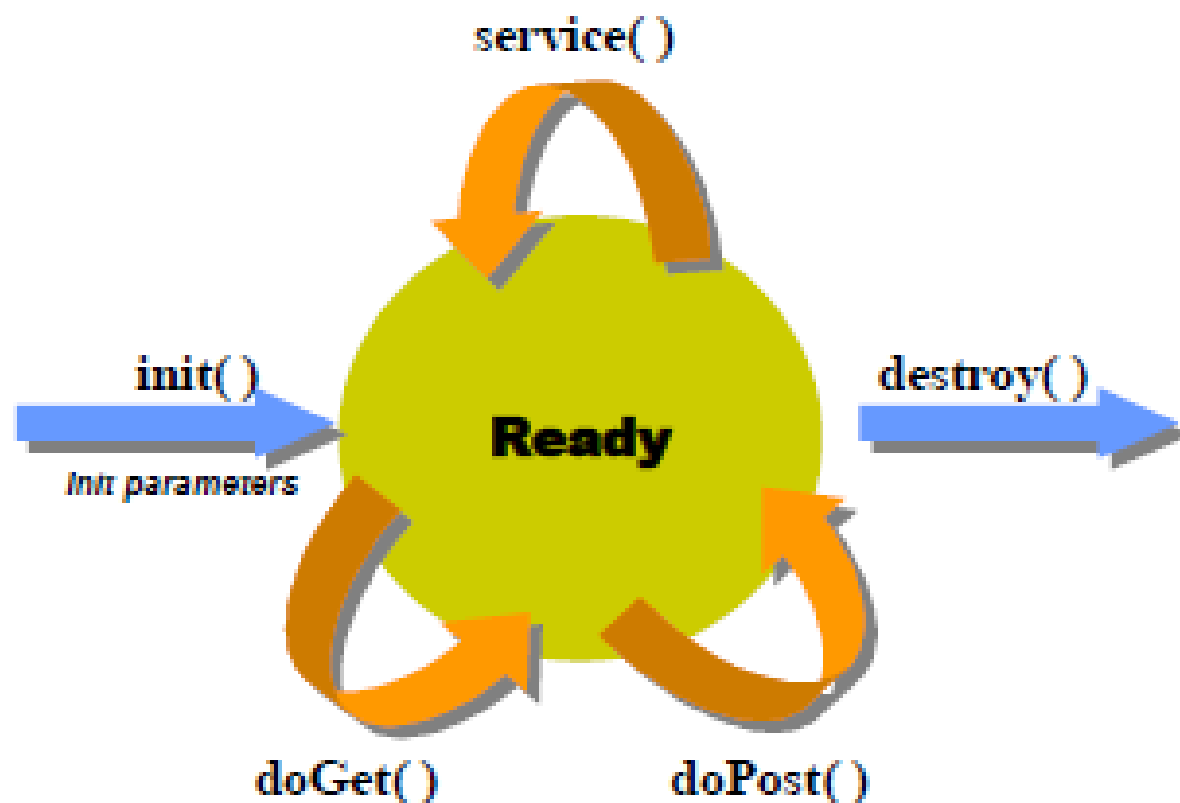
NewServlet.java  web.xml  index.jsp  CircleServlet.java  CircleServlet.jsp
Source  History
19  public class CircleServlet extends HttpServlet {
20      int a=0;
21
22      public void init() throws ServletException {
23          System.out.println("init");
24          a += 5;
25          System.out.println("a= "+a);
26      }
27
28      protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
29          protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
30              response.setContentType("text/html;charset=UTF-8");
31              PrintWriter out = response.getWriter();
32              try {
33                  out.println("<html>");
34                  out.println("<head>");
35                  out.println("<title>Servlet NewServlet</title>");
36                  out.println("</head>");
37                  out.println("<body>");
38                  a+=10;
39                  out.println("a= " +a);
40                  out.println("</body>");
41                  out.println("</html>");
42              } finally {
43                  out.close();
44              }
45          }
46      }
47
48      public void destroy() {
49          System.out.println("Destroy is invoked");
50      }
51
52      HttpServlet methods. Click on the + sign on the left to edit the code.
    
```



DEMO

Chạy và giải thích





■ init()

- Được gọi **MỘT** lần khi servlet được tạo thể hiện hóa lần đầu tiên
- Thực hiện các **khởi tạo** trong phương thức này
 - Ví dụ: tạo 1 kết nối CSDL

■ destroy()

- Được gọi trước khi hủy 1 servlet instance
- Thực hiện thao tác dọn dẹp
 - Ví dụ: đóng kết nối CSDL đã mở

```
public class CatalogServlet extends HttpServlet {
    private BookDB bookDB;

    // Perform any one-time operation for the servlet,
    // like getting database connection object.

    // Note: In this example, database connection object is assumed
    // to be created via other means (via life cycle event mechanism)
    // and saved in ServletContext object. This is to share a same
    // database connection object among multiple servlets.
    public void init() throws ServletException {
        bookDB = (BookDB) getServletContext().
            getAttribute("bookDB");
        if (bookDB == null) throw new
            UnavailableException("Couldn't get database.");
    }
    ...
}
```

```
public void init(ServletConfig config) throws
    ServletException {
    super.init(config);
    String driver = getInitParameter("driver");
    String fURL = getInitParameter("url");
    try {
        openDBConnection(driver, fURL);
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

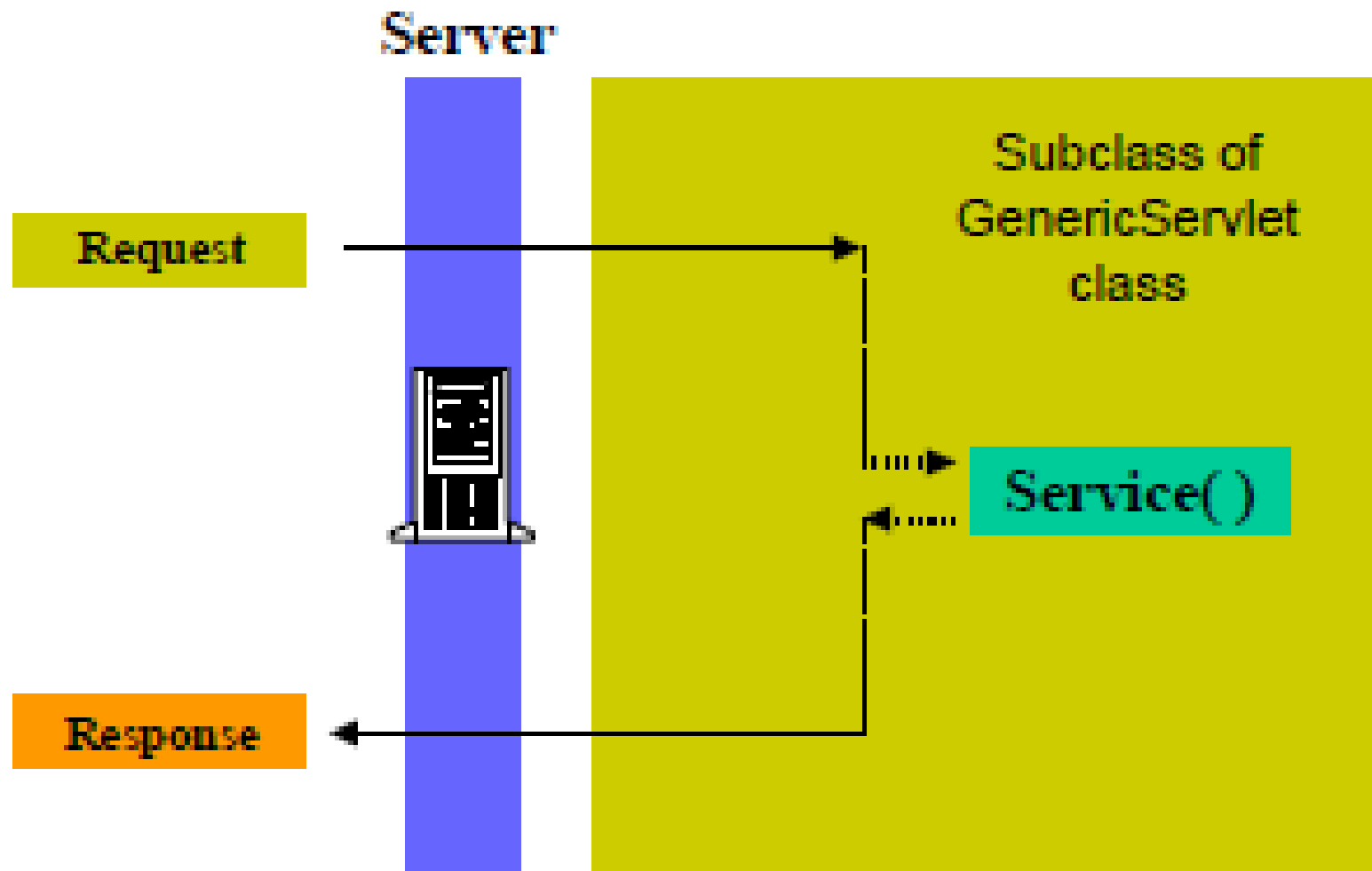
```
<web-app>
  <servlet>
    <servlet-name>chart</servlet-name>
    <servlet-class>ChartServlet</servlet-class>
    <init-param>
      <param-name>driver</param-name>
      <param-value>
        COM.cloudscape.core.RmiJdbcDriver
      </param-value>
    </init-param>

    <init-param>
      <param-name>url</param-name>
      <param-value>
        jdbc:cloudscape:rmi:CloudscapeDB
      </param-value>
    </init-param>
  </servlet>
</web-app>
```

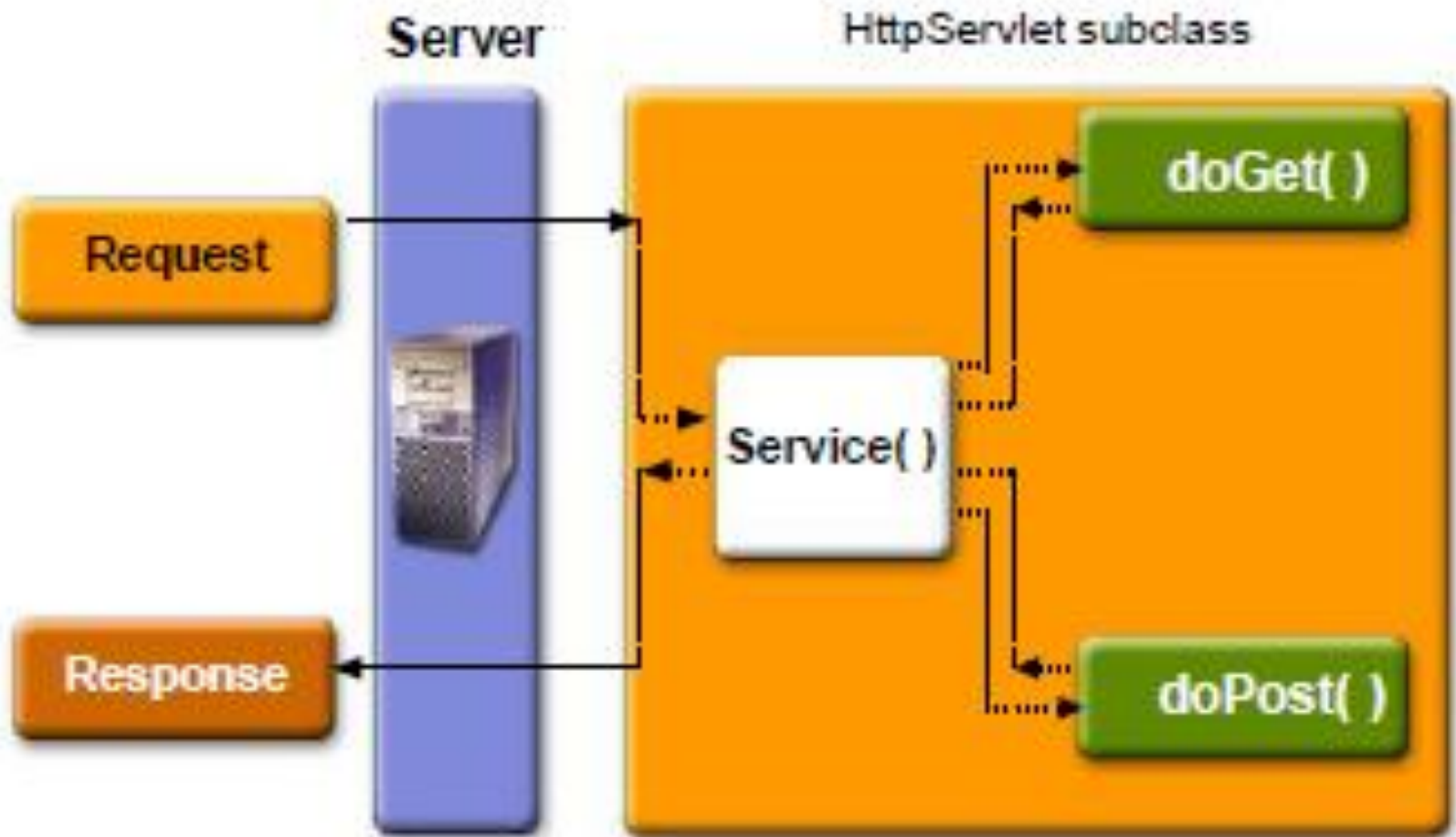
```
public class CatalogServlet extends HttpServlet {
    private BookDB bookDB;

    public void init() throws ServletException {
        bookDB = (BookDB) getServletContext().
            getAttribute("bookDB");
        if (bookDB == null) throw new
            UnavailableException("Couldn't get database.");
    }
    public void destroy() {
        bookDB = null;
    }
    ...
}
```


- ❑ Method `service()` của servlet được gọi mỗi khi có request từ client đến.
- ❑ Tùy vào loại http request cụ thể nó gọi đến một trong các method `doGet(..)`, `doPost(...)`. Tại các servlet của bạn, bạn phải ghi đè và xử lý tại các method này
- ❑ Các phương thức thường dùng:
 - ❖ `public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`
 - ❖ `public void doXXX(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`



DOGET() VÀ DOPOST()



- Trích xuất các thông tin gửi từ client (HTTP parameter) từ HTTP request
- Thiết lập/truy cập các thuộc tính của các **Scope objects**
- Thực hiện các xử lý nghiệp vụ (**business logic**) hoặc truy cập CSDL
- Tùy chọn forward request tới các Web components khác (Servlet hoặc JSP)
- Sinh HTTP response và trả về cho client

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

Public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        // Just send back a simple HTTP response
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<title>First Servlet</title>");
        out.println("<big>Hello J2EE Programmers! </big>");
    }
}
```

□ Nhiệm vụ của Servlet:

- ❖ Nhận client request (Hầu hết ở dạng HTTP request)
- ❖ Trích xuất 1 số thông tin từ request
- ❖ Xử lý nghiệp vụ (truy vấn DB, gọi EJBs,...) hoặc sinh nội dung động
- ❖ Tạo và gửi response cho client (hầu hết ở dạng HTTP response) hoặc forward request cho servlet khác

■ Request là gì?

- Thông tin được gửi từ client tới 1 server
 - Ai tạo ra request
 - Dữ liệu gì được user nhập vào và gửi đi
 - HTTP headers

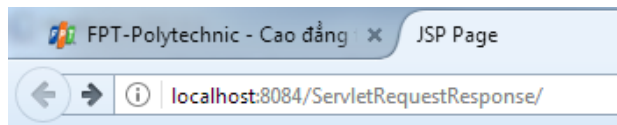
■ Response là gì?

- Thông tin được gửi đến client từ 1 server
 - Dữ liệu Text (html, thuần text) hoặc dữ liệu binary (image)
 - HTTP headers, cookies, ...

- HTTP request bao gồm
 - header
 - Phương thức
 - Get: Thông tin nhập vào trong form được truyền như 1 phần của URL
 - Post: Thông tin nhập vào trong form được truyền trong nội dung thông điệp (**message body**)
 - Put
 - Header
 - Dữ liệu trong request (**request data**)

- Các client requests thông dụng nhất
 - HTTP GET & HTTP POST
- GET requests:
 - Thông tin người dùng nhập vào **đính kèm** trong URL dưới dạng 1 query string
 - Chỉ gửi được lượng dữ liệu giới hạn
 - `.../servlet/ViewCourse?FirstName=Sang&LastName=Shin`
- POST requests:
 - Thông tin người dùng nhập vào được gửi dưới dạng dữ liệu (không đính kèm vào URL)
 - Gửi được lượng dữ liệu bất kỳ

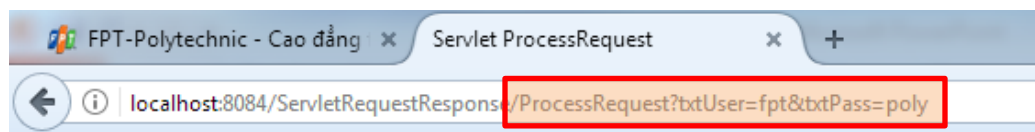
- Tất cả các servlet requests đều thực thi giao diện `ServletRequest` định nghĩa các phương thức truy cập tới:
 - Các tham số (parameters) gửi từ clients
 - Object-valued attributes
 - Client và server
 - Input stream
 - Thông tin về giao thức (Protocol information)
 - Content type
 - request có được tạo trên 1 kênh truyền secure không



Login Form

Username:

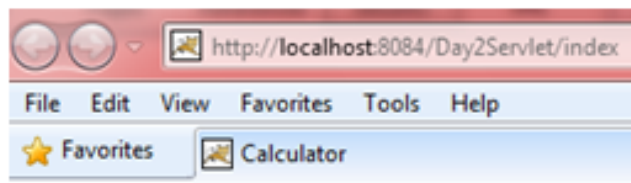
Password:



HttpServlet Request Demos

parName1is txtUser and value is fpt
parName2is txtPass and value is poly
Server name:localhost
Lengths in bytes: -1
Request method: GET
Path Info: null
Authentication type: null
Query string: txtUser=fpt&txtPass=poly

header host is localhost:8084
header user-agent is Mozilla/5.0 (Windows NT 6.1; rv:53.0) Gecko/20100101 Firefox/53.0
header accept is text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
header accept-language is en-US,en;q=0.5
header accept-encoding is gzip, deflate
header referer is http://localhost:8084/ServletRequestResponse/
header cookie is JSESSIONID=CC8216CD7DB56C2333456DA953EE21A9
header connection is keep-alive
header upgrade-insecure-requests is 1
Headers Accept text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8



Servlet Demo

Num1

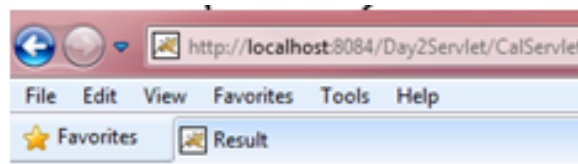
Num2

```
<form action="CalServlet">
  Num1 <input type="text" name="txtNum1" value="" /><br/>
  Num2 <input type="text" name="txtNum2" value="" /><br/>
  <input type="submit" value="+" name="btAction" />
  <input type="submit" value="-" name="btAction" />
</form>
```

```
String action = request.getParameter("btAction");
```

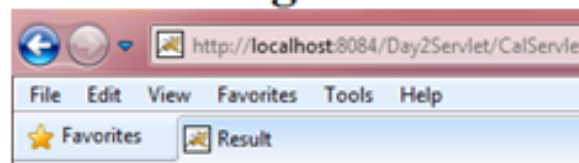
```
...
```

```
if(action.equals("+")){
    out.println("<h2> " + num1 + " + " + num2 + " = " + (a+b) + "</h2>");
}else {
    out.println("<h2> " + num1 + " - " + num2 + " = " + (a-b) + "</h2>");
}
```



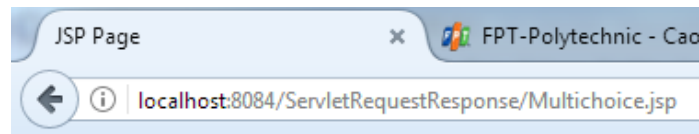
Result of Calculating

5 + 6 = 11



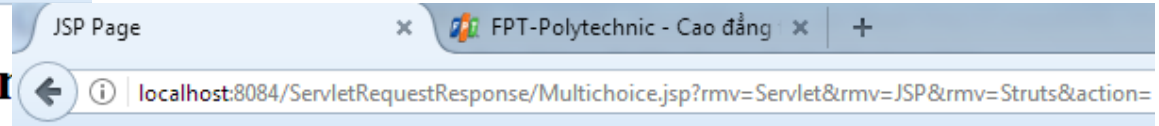
Result of Calculating

5 - 6 = -1



HttpRequest servlet demo

- ☐ Servlet
 - ☐ JSP
 - ☐ Struts
 - ☐ JSF
 - ☐ EJB
- Choose



HttpRequest servlet demo

- ☐ Servlet
 - ☐ JSP
 - ☐ Struts
 - ☐ JSF
 - ☐ EJB
- Choose

Selected item name: Servlet
Selected item name: JSP
Selected item name: Struts

```
String[] strSelect = request.getParameterValues("rmv");  
if(strSelect !=null){  
    for(int i=0; i<strSelect.length;i++){  
        out.println("Selected item name: " + strSelect[i] + "<br/>");  
    }  
}
```



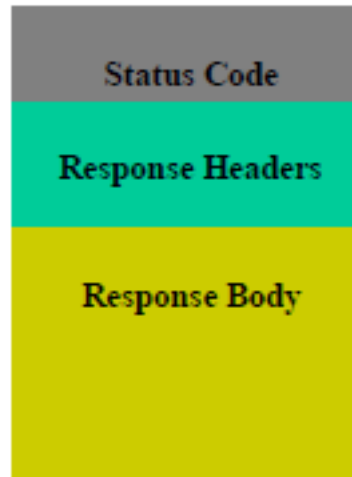
DEMO

Chạy và giải thích



- Tất cả các các servlet responses thực thi giao diện ServletResponse
 - Lấy 1 output stream
 - Chỉ định content type
 - Có thiết lập buffer đầu ra không
 - Thiết lập localization information
- HttpServletResponse kế thừa giao diện ServletResponse
 - Mã trạng thái HTTP trả về (HTTP response status code)
 - Cookies

□ Cấu trúc Response



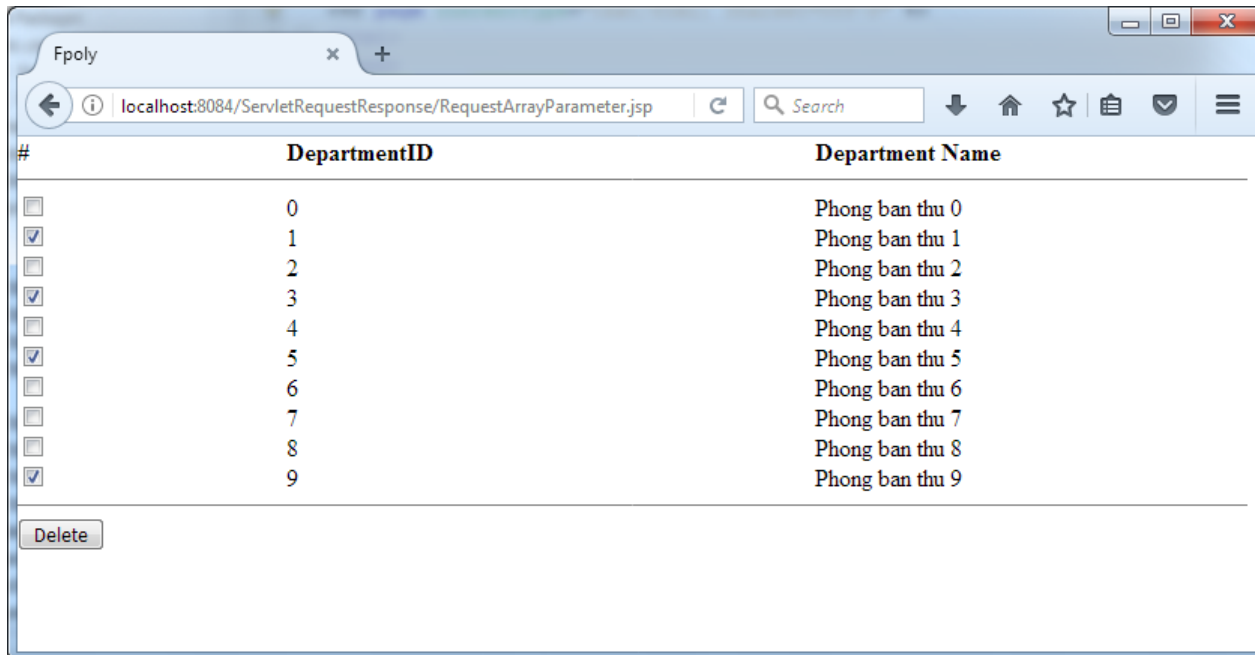
- Tại sao cần **HTTP response status code**?
 - Giúp trình duyệt forward đến 1 trang khác
 - Chỉ ra được có resource bị thiếu
 - Hướng dẫn browser sử dụng bản sao được cache của dữ liệu

- **public void setStatus(int statusCode)**
 - Mã trạng thái được định nghĩa trong `HttpServletResponse`
 - Các mã trạng thái chia làm 5 nhóm:
 - 100-199 Informational
 - 200-299 Successful
 - 300-399 Redirection
 - 400-499 Incomplete
 - 500-599 Server Error
 - Mã trạng thái mặc định là 200 (OK)

- Forward đến địa chỉ mới nào
- Sửa cookies
- Cung cấp thông tin thời gian chỉnh sửa page.
- Hướng dẫn trình duyệt load lại trang sau 1 khoảng thời gian nhất định
- Đưa ra kích thước file được sử dụng trong HTTP connections loại persistent
- Chỉ định loại document sinh ra & trả về client
- ...

- `setContentType`
 - Thiết lập **Content-Type** header. Servlets gần như luôn sử dụng phương thức này.
- `setContentLength`
 - Thiết lập **Content-Length** header. Được sử dụng cho HTTP connections loại persistent .
- `addCookie`
 - Thêm 1 giá trị trong **Set-Cookie** header.
- `sendRedirect`
 - Thiết lập **Location** header và thay đổi mã trạng thái

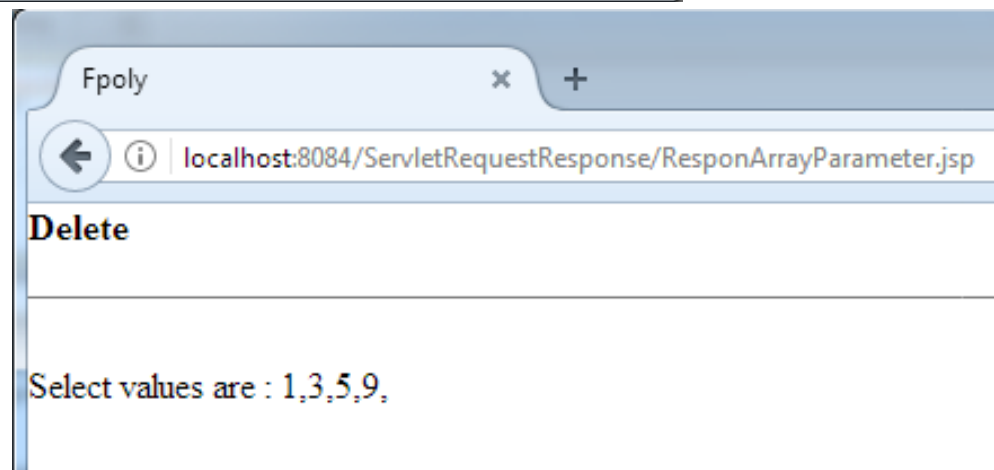
- Một servlet gần như luôn trả về 1 response body
- Response body có thể là một `PrintWriter` hoặc một `ServletOutputStream`
- `PrintWriter`
 - Sử dụng phương thức `response.getWriter()`
 - Cho output loại ký tự (character-based)
- `ServletOutputStream`
 - Sử dụng phương thức `response.getOutputStream()`
 - Cho dữ liệu dạng binary (ví dụ: image)



A screenshot of a web browser window titled 'Fpoly'. The address bar shows 'localhost:8084/ServletRequestResponse/RequestArrayParameter.jsp'. The page displays a table with three columns: '#', 'DepartmentID', and 'Department Name'. The table contains 10 rows, each with a checkbox, a DepartmentID (0-9), and a 'Phong ban thu' name. Rows 1, 3, 5, and 9 are selected. A 'Delete' button is at the bottom left.

#	DepartmentID	Department Name
<input type="checkbox"/>	0	Phong ban thu 0
<input checked="" type="checkbox"/>	1	Phong ban thu 1
<input type="checkbox"/>	2	Phong ban thu 2
<input checked="" type="checkbox"/>	3	Phong ban thu 3
<input type="checkbox"/>	4	Phong ban thu 4
<input checked="" type="checkbox"/>	5	Phong ban thu 5
<input type="checkbox"/>	6	Phong ban thu 6
<input type="checkbox"/>	7	Phong ban thu 7
<input type="checkbox"/>	8	Phong ban thu 8
<input checked="" type="checkbox"/>	9	Phong ban thu 9

Delete



A screenshot of a web browser window titled 'Fpoly'. The address bar shows 'localhost:8084/ServletRequestResponse/ResponArrayParameter.jsp'. The page displays the word 'Delete' in bold. Below it, the text 'Select values are : 1,3,5,9,' is shown.

Delete

Select values are : 1,3,5,9,



DEMO

Chạy và giải thích





LẬP TRÌNH JAVA 4

BÀI 2: SERVLET, REQUEST, RESPONSE, SESSION TRACKING

PHẦN 2

Session Tracking

- HTTP là protocol không trạng thái (stateless protocol), nghĩa là server khi tiếp nhận 1 yêu cầu sẽ trả về 1 đáp ứng rồi cắt kết nối, không lưu trữ thông tin trạng thái về yêu cầu vừa qua. Server xử lý độc lập theo từng yêu cầu từ client. Làm sao biết được một user đã vào web site của ta hay chưa? Thông qua Session, người phát triển web làm được điều này.
- Có 4 kỹ thuật để lưu dấu session:
 - URL rewriting – Thêm các tham số vào cuối URL
 - Hidden fields – Sử dụng các trường ẩn
 - Cookies – Sử dụng cookie để trao đổi dữ liệu giữa client và server
 - Session *objects* – Sử dụng các đối tượng có phạm vi (scope) là session để truyền thông tin.

Session Tracking

- **URL Rewriting:**
- VD: Http://localhost:8080/Demo/test?x=abc&y=xyz
- Phần sau ?x=abc&y=xyz là phần được thêm vào để truyền 2 parameter x và y
- Để lấy giá trị này ta dùng lệnh **request.getParameter("x")**, tương tự với y.

Session Tracking

- **Hidden fields:**
- `<INPUT TYPE=HIDDEN Name=hField VALUE="abc">`
- Để lấy giá trị này ta dùng lệnh **`request.getParameter("hField")`**.

Session Tracking

- **Cookies:** được lưu bởi server và gửi về client cùng response. Request được gửi tới server cùng với cookie nhưng ko thay đổi giá trị của cookie. Giá trị của cookie được lưu trong bộ nhớ (ổ cứng) của client.



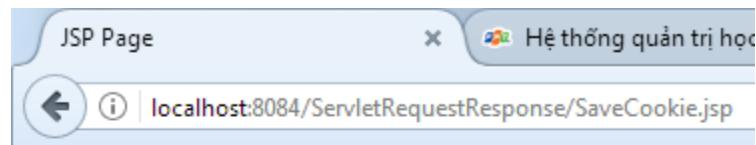
Session Tracking

- VD1: lưu cookie (sử dụng response)

```
Cookie c1 = new Cookie("userName", "Helen");  
Cookie c2 = new Cookie("password", "Keppler");  
c2.setMaxAge(300);  
response.addCookie(c1);  
response.addCookie(c2);
```

- VD2: đọc cookie

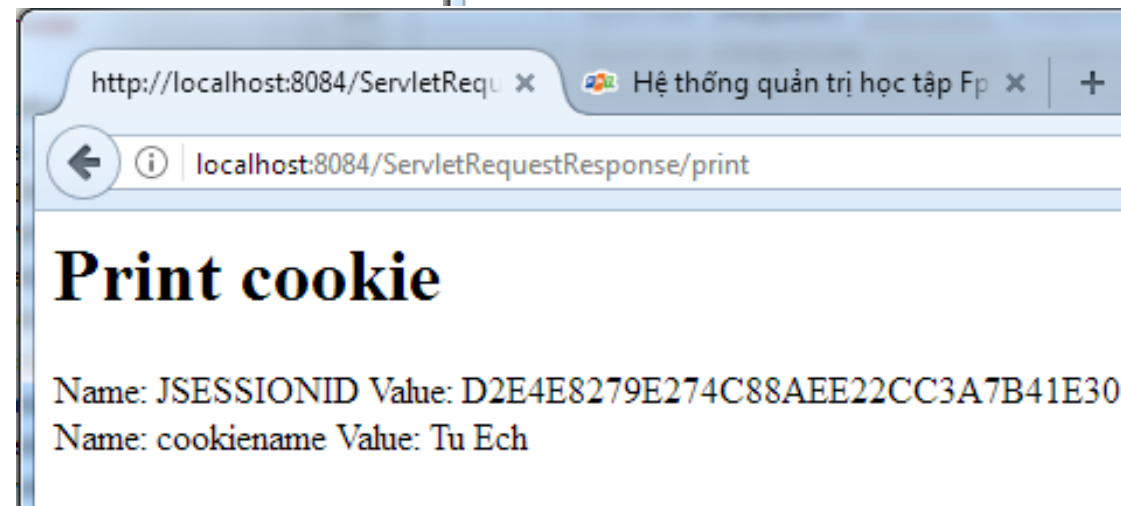
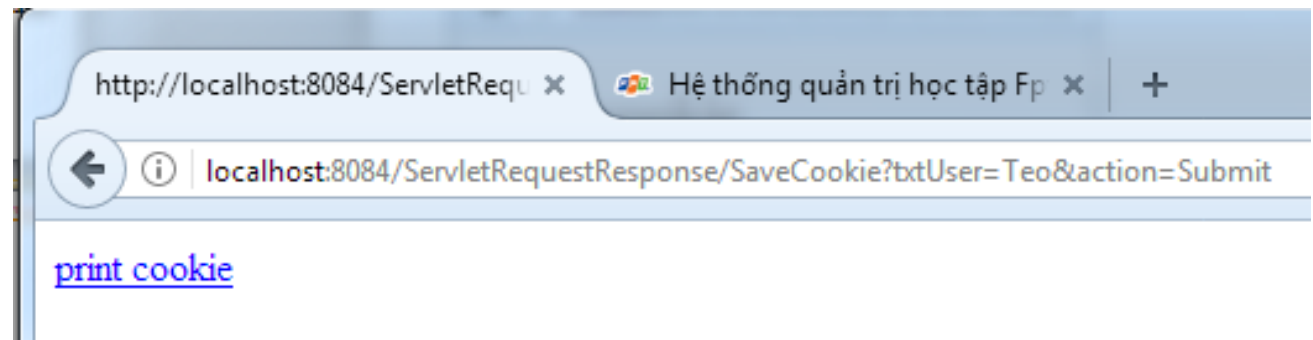
```
Cookie[] cookies = request.getCookies();  
for (int i = 0; i < cookies.length; i++) {  
    Cookie cookie = cookies[i];  
    out.println("Name->" + cookie.getName() + " Value->" + cookie.getValue());  
}
```

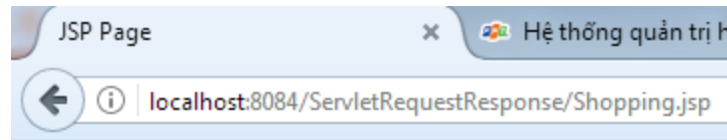


Save cookie

Teo

Submit





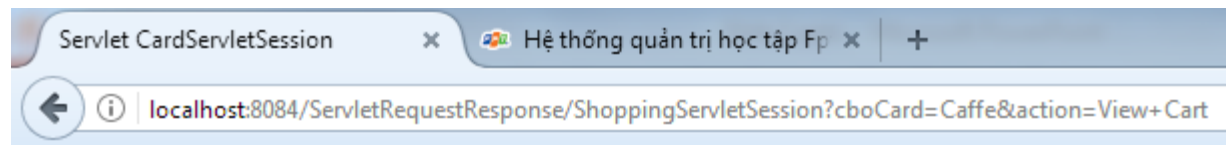
Shopping card

Please choose your favourish:

Caffe ▼

Add To Cart

View Cart



Shopping Cart

Cac mon hang da chon:

TT	Items	Check
1	Kem	<input type="checkbox"/>
2	Duong	<input type="checkbox"/>
3	Ca	<input type="checkbox"/>
4	Banh Mi	<input type="checkbox"/>
	Back	<input type="button" value="Delete"/>



DEMO

Chạy và giải thích



- ❖ Servlet là gì?
- ❖ Servlet Scope Object
- ❖ Servlet Request
- ❖ Servlet Response
- ❖ Section Tracking





Cảm ơn