

CC68J APLICACIONES EMPRESARIALES CON JEE

SERVLETS

Entendiendo como se realiza la comunicación

Profesores:
■ Andrés Farías

fcfm

v1.0

Objetivos: aprender a...

1. A

v1.1

¿Qué son los Servlets?

- Servlets son módulos que extienden las funcionalidades de un servidor "java-enabled".
- Piedra angular del desarrollo de aplicaciones web en Java.
- Pensados para reemplazar a los CGI's
 - ✓ Cada petición de un cliente hacia al servidor lanza un nuevo proceso con el programa CGI
- Normalmente generan código HTML dinámicamente, el cual se manda al browser, responsable de su render.
- Por ejemplo, pueden mandar una query a una base de datos, la que puede estar basada en parámetros que mandó el browser al servidor. El resultado se manda en formato HTML.

```

graph LR
    Client[Order-Entry Client] --> Servlet[Order-Entry Servlet]
    Servlet --- Server[HTTP Server]
    Servlet <--> DB[(Inventory Database)]
  
```

v1.1

Ventajas

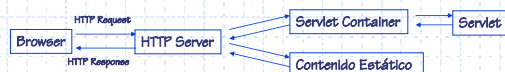
- **Rendimiento.** cada petición es procesada por un único proceso en el contenedor de servlets.
- **Portabilidad.** heredado de Java
- **Rápido desarrollo.** acceso a las riquísimas librerías de Java.
- **Robustez.** Son gestionados por la máquina virtual de Java (garbage collection)
- **Amplio soporte.** muchos desarrolladores y compañías utilizan esta tecnología

La representación en Java

LA CLASE HTTPServlet

Arquitectura

- Servlet Container = servidor web capaz de ejecutar servlets.
- Cuando una petición es enviada desde el cliente (browser) el servidor la transfiere al **Servlet Container**, transfiriéndoselo a su vez al Servlet que debe atender la petición.
- El Servlet atiende la petición y genera la respuesta, creando y configurando el objeto que representa esta respuesta (response).



La clase `HttpServlet`

Anatomía

- Un Servlet se escribe extendiendo la clase `HttpServlet` la que tiene los siguientes métodos declarados pero vacíos
- Método `init()`.
 - ✓ Es invocado por el servidor cuando el servlet se usa por primera vez (cuando sucede esto depende del servidor)
- Método `doGet(HttpServletRequest req, HttpServletResponse res)` throws `ServletException`, `IOException`
 - ✓ Es invocado cada vez que un servlet es contactado por un requerimiento GET (por default)
- Método `doPost(HttpServletRequest req, HttpServletResponse res)` throws `ServletException`, `IOException`
 - ✓ Es invocado cada vez que un servlet es contactado por un requerimiento POST

La clase `HttpServlet`

Ejemplo

```
public class MyServlet extends HttpServlet {

    public void init() {
        // Sobre-escribir para que haga lo que queramos
    }

    public void doGet ( HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //Sobre-escribir para que haga lo que queramos
    }

    public void doPost( HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //Sobre-escribir para que haga lo que queramos
    }
}
```

La clase `HttpServlet`

El parámetro `HttpServletRequest`

- `HttpServletRequest` es la clase del primer parámetro que reciben los métodos `doGet` y `doPost`. Provee acceso a:
 - ✓ Información acerca del cliente (por ejemplo, los parámetros que envió, la versión del protocolo que está usando, la IP del cliente, etc.).
 - ✓ Además da acceso a un `ServletInputStream` que puede ser usado por el servidor para recibir información adicional (por ejemplo un archivo que el cliente quiere "uploadear" cuando se ha usado el método `POST` o `PUT`).

La clase `HttpServlet`

El parámetro `HttpServletResponse`

- `HttpServletResponse` es la clase del segundo argumento que reciben `doGet` y `doPost`.
- Provee métodos para :
 - ✓ Decirle al browser cuál es el tipo MIME de la respuesta que se le va a dar al cliente
 - ✓ Obtener un objeto `ServletOutputStream` y un `PrintWriter` a través del cual podemos mandar código HTML dinámicamente al cliente.
 - ✓ Mandar otras informaciones importantes (cookies, redirección, etc...)

Ejemplo de Servlet (1)

Enviando la fecha del sistema

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

        //send data
        out.println("<HTML>");
        out.println("<H1> Mi Primer Servlet </H1>");
        out.println("<BR> <H2>Fecha y hora: " + (new Date()) + "</H2>");
        out.println("</HTML>");
        out.close();
    }
}
```

Se indica el tipo de contenido

Se abre un canal para la respuesta.

Se incluye la fecha del sistema.

Ejemplo de Servlet (2)

Implementando un contador

- Implementar un contador web
- Cuenta cuantas veces se contacta al servlet (con GET).
- También le mostrará al cliente su dirección IP.

Ejemplo de Servlet (2)

Implementando un contador

```
public class Count extends HttpServlet {
    int count = 0;

    public void doGet ( HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        count++;
        res.setContentType("text/html"); PrintWriter out = res.getWriter();

        out.println("<H1> A web page counter: </H1>");
        out.println("<BR>");
        out.println("This servlet was accessed "+count+" time(s)");
        out.println("<BR>");
        out.println("Your computer is "+req.getRemoteHost());
        out.println("<BR>"); out.close();
    }
}
```

Se imprime el contenido al response

Contador iniciado en cero

Incrementa el contador cada vez que se recibe una petición con el método get

Ejemplo de Servlet (2)

Preguntas...

- ¿Qué pasa si el servidor se cae y se reinicia?
 - ✓ Contador volverá a 0
 - ✓ Para recordar el valor que tenía esta variable incluso si se cae el sistema la iremos escribiendo en un archivo cada vez que cambia su valor.
- Si una instancia de Count está atendiendo peticiones, y el valor de su contador es 10 ¿Qué valor desplegará si otro cliente hace una petición a la misma URL?
 - ✓ Desplegará el valor 0.
 - ✓ El servidor establece una sesión (lógica) con cada cliente.
 - ✓ Se instancia un servlet en un nuevo thread por cada sesión.

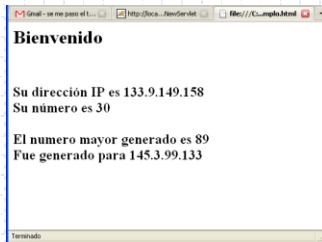
Ejemplo de Servlet (3)

Variables estáticas

- Escriba un servlet que genere un número aleatorio entre 1 y 100 cada vez que es contactado y lo muestra al cliente junto con la IP del computador del cliente
- Además muestra el número más grande generado hasta ahora y la IP del computador para el cual lo generó
- La dirección del computador del cliente puede ser obtenida
 - `String s = request.getRemoteHost();`
 - Retorna el número IP (133.8.109.158)
 - `String s = request.getRemoteAddress();`
 - Retorna el nombre si puede ser recuperado (www.waaseda.jp)
- Un número aleatorio entre 1 y 100 se genera con la siguiente instrucción en Java
 - `int numero = (int)(1 + Math.random()*100);`

Ejemplo de Servlet (3)

Como se vería...



Ejemplo de Servlet (3)

La solución

```
public class NeoServlet extends HttpServlet {
    static int maxNumber = 0;
    static String maxIP = "";

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

        int aleatorio = (int)(1 + Math.random()*100);
        String ip = request.getRemoteAddr();
        if (aleatorio > maxNumber) {
            maxNumber = aleatorio;
            maxIP = ip;
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<p>Su dirección IP es " + ip + "<br>"); out.println("Su número es " + aleatorio + "<br>");
        out.println("El número mayor generado es " + maxNumber); out.println("<br>Fue generado para " + maxIP);
        out.close();
    }
}
```

Variables estáticas para almacenar los campeones!

Se genera el número random y se obtiene la IP del cliente.

Actualización del campeón.

Se envía la información al cliente.

Ejemplo de Servlet (4)

Memoria de las IP visitadas

- Modificar el servlet contador de modo de generar una respuesta que muestre la IP de todos los computadores que lo han contactado hasta ahora
- ✓ Use un vector o un arreglo para ir guardándolas

Ejemplo de Servlet (4)

Memoria de las IP visitadas

```
public class AddressesServlet extends HttpServlet {
    int nIP = 0;
    String[] IPs = new String[1000];

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

        String ip = request.getRemoteHost();
        IPs[nIP++] = ip;

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

        out.println("<h1> Han visitado este sitio </h1><br>");

        for (int i = 0; i < nIP; i++)
            out.println("<br> " + IPs[i]);
        out.close();
    }
}
```

Se obtiene la IP del cliente y se almacena en el arreglo.

Imprime cada IP separado por una línea.

Ejemplo de Servlet (5)

Memoria de las IP visitadas

- Modificar el ejemplo anterior para que en vez de repetir la ip muestre cuantas veces ha visitado ese cliente al servlet
- ✓ Use un vector que contenga objetos de la clase Node, el cual contiene un String (para guardar el número IP) y un entero (para contar las visitas del cliente)

```
public class Node {
    int count; String ip;
    Node(String x, int y) {
        ip = x; count = y;
    }
}
```

Ejemplo de Servlet (5)

Solución

```
public class ShowCountsServlet extends
    HttpServlet {
    Vector v = new Vector();

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String ip = request.getRemoteHost();

        Node aux; int p = 0;
        while(p < v.size()) {
            aux = (Node)v.elementAt(p);
            if (ip.equals(aux.ip)) break;
            p++;
        }

        if (p == v.size()) {
            aux = new Node(ip, 0);
            v.add(aux);
            out.println("agregado");
        }

        aux = (Node)v.elementAt(p);
        aux.count++;

        out.println("<h1> Following clients
        visited this site </h1><br>");
        for (int i = 0; i < v.size(); i++) {
            aux = (Node)v.elementAt(i);
            out.println("<br> " + aux.ip + "
            has been " + aux.count + " times here");
        }
        out.close();
    }
}
```

Si la IP no estaba, se agrega al final del vector.

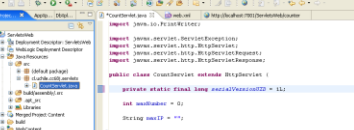
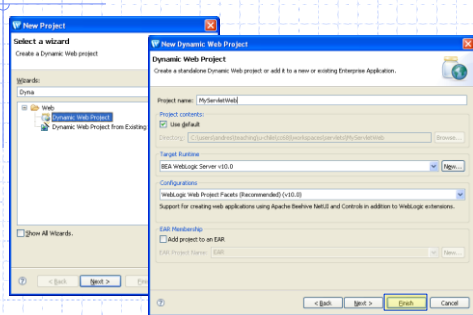
Se incrementa el contador.

Se obtiene la IP.

Se busca la IP en el vector para incrementar el contador.

Se imprimen los clientes y sus contadores de visita.

UNA IMPLEMENTACIÓN CONCRETA



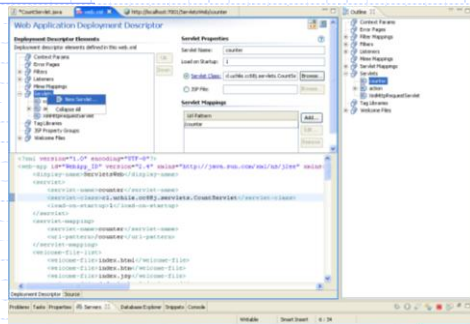
```

1  using System;
2  using System.Net.Http;
3  using System.Text;
4  using System.Threading.Tasks;
5
6  namespace Client
7  {
8      class Program
9      {
10         static async Task Main(string[] args)
11         {
12             await RunAsync().ConfigureAwait(false);
13         }
14
15         static async Task RunAsync()
16         {
17             var client = new HttpClient();
18             var url = "http://localhost:5000/api/values";
19             var data = new { id = 1, name = "John" };
20             var response = await client.PostAsync(url, new StringContent(JsonConvert.SerializeObject(data), Encoding.UTF8, "application/json")).ConfigureAwait(false);
21             if (response.IsSuccessStatusCode)
22             {
23                 RunAsyncResponse(response);
24             }
25             else
26             {
27                 RunAsyncError(response);
28             }
29         }
30
31         static void RunAsyncResponse(HttpResponseMessage response)
32         {
33             Console.WriteLine(response.Content.ReadAsStringAsync().Result);
34         }
35
36         static void RunAsyncError(HttpResponseMessage response)
37         {
38             Console.WriteLine(response.Content.ReadAsStringAsync().Result);
39         }
40     }
41 }

```

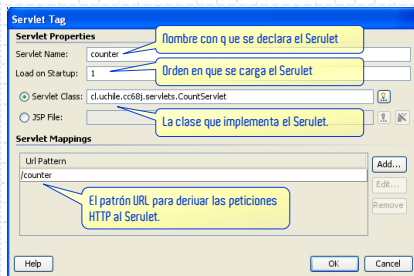

Configuración

Modificando el web.xml



Configuración

Ingresando los detalles del Servlet



Configuración

Resultado final

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>ServletesWeb</display-name>
  <servlet>
    <servlet-name>counter</servlet-name>
    <servlet-class>cl.uchile.cc68j.servlets.CountServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>counter</servlet-name>
    <url-pattern>/counter</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  ...
</web-app>
```

Enviándole datos al Servlet

ENVÍO DE PARÁMETROS

Envío de parámetros

Uía URL (uía método get)

- Una de las funcionalidades que hacen que la web sea interactiva es el paso de parámetros (información) del cliente al servidor
- El cliente puede traspasar parámetros al servidores con el requerimiento (URL) de la siguiente manera:
 - ✓ `http://host:port/servlet?param1=valor1¶m2=valor2`
 - ✓ Esto implica que el servidor recibirá 2 parámetros: uno con el nombre `param1` y valor `valor1` y el otro con el nombre `param2` y valor `valor2`
- Un servlet puede consultar por estos parámetros de la siguiente manera:
 - ✓ `String valueOfParam1 = request.getParameter("param1");`
 - ✓ `String valueOfParam2 = request.getParameter("param2");`
- Los nombres y valores de los parámetros son strings.
- Los nombres difieren en el caso de contener mayúsculas y minúsculas (`Param1 != param1`)

Ejemplo

`http://host:port/MyServlet?firstname=Nelson&lastname=Baloian`

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

public class ServletParameter1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, java.io.IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");

        // obtaining parameter value for parameter named "firstname"
        String fname = request.getParameter("firstname");

        // obtaining parameter value for parameter named "lastname"
        String lname = request.getParameter("lastname");
        out.println("<h1> Hello "+fname+" "+lname+"</h1>");
        out.close();
    }
}
```

Envío de parámetros

Úa un formulario web

- Un Form es una página HTML que recolectará la información y la traspasarán automáticamente al servidor con la URL.

```
<HTML>
<H1>
Collecting parameters
</H1>
```

```
<FORM ACTION="/MyServlet">
```

```
Nombre: <INPUT TYPE=TEXT NAME="firstname"><BR>
```

```
Apellido: <INPUT TYPE=TEXT NAME="lastname"><BR>
```

```
<INPUT TYPE=SUBMIT VALUE="MANDAR">
```

```
</FORM ACTION="/MyServlet">
```

```
</HTML>
```

Inicio del formulario y declaración del action

Type Submit es para enviar la petición con los parámetros a la acción indicada.

Input elemento de entrada de datos. Type define el tipo (texto), y name es el nombre del parámetro.

Envío de parámetros

Envío vía método get

- Al oprimir el botón se obtiene el Resultado que muestra la figura de abajo.
- Fijarse en la URL que se generó.
- Automáticamente con los parámetros

Ejercicio 1

¿Qué página generan estos recursos?

```
<HTML>
```

```
<FORM ACTION="/Jalisco">
```

Ingresar un número cualquiera y luego oprime el botón:

```
<INPUT TYPE=TEXT NAME=numero><BR>
```

```
<INPUT TYPE=SUBMIT VALUE="jugar">
```

```
</FORM>
```

```
</HTML>
```

Ejercicio 1

¿Qué página generan este Servlet?

```
public class Jalisco extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, java.io.IOException {
        PrintWriter out;
        response.setContentType("text/html");

        String snum = request.getParameter("numero");

        int num = Integer.parseInt(snum);

        out = response.getWriter();
        out.println("<h1> Te gano con el "+(num+1)+"</h1>");
        out.close();
    }
}
```

<http://host:port/Jalisco?numero=47>

Ejercicio 2

¿Qué página generan este Servlet?

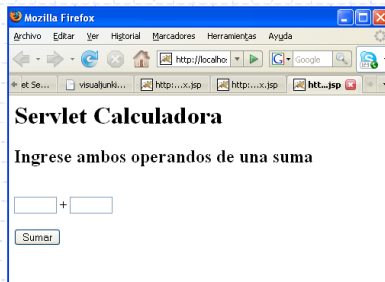
```
<HTML>
<H1> Servlet Calculadora </H1>
<H2>Ingrese ambos operandos de una suma
</H2> <BR>
<FORM ACTION="ServletCalculadora">
    <INPUT TYPE=TEXT SIZE=5 NAME="op1"> +
    <INPUT TYPE=TEXT SIZE=5 NAME="op2"><BR> <BR>
    <INPUT TYPE=SUBMIT VALUE="Sumar">
</FORM>
</HTML>
```

Ejercicio 2

¿Qué página generan este Servlet?

1. Escriba el servlet ServletCalculadora que al ser contactado responda con la suma de ambos números

2. Modifique el HTML de modo que la operación también sea ingresada por el usuario y el servlet haga la operación adecuada (solo se permite +, -, *, /)



ELEMENTOS HTML DE INPUT

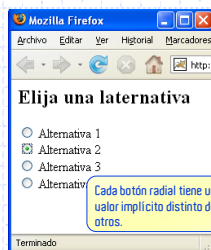
Data inputs

Otros elementos HTML para input

- **Radio:**
 - ✓ se puede seleccionar una alternativa entre varias
- **Select:**
 - ✓ como el radio pero con un menú pulldown
- **TextArea:**
 - ✓ como el texto pero contiene varias líneas
- **Password:**
 - ✓ como el texto pero lo que se ingrese se verá como *****
 - ✓ en vez de lo que realmente se ingresará
 - ✓ Ojo que el dato viaja como texto plano de todos modos.

Radio Button

Sintaxis HTML



<h2> Elija una Alternativa </h2>

<HTML>

El tipo del elemento es "radio"

<input type="radio" name="radio1" value="valor1"> Alternativa 1

<input type="radio" name="radio1" value="valor2"> Alternativa 2

<input type="radio" name="radio1" value="valor3"> Alternativa 3

<input type="radio" name="radio1" value="valor4"> Alternativa 4

</HTML>

Todos los botones asociados al mismo grupo deben tener el mismo nombre.

Radio Button

Recuperación del parámetro

<h2> Elija una Alternativa </h2>

<HTML>

<input type="radio" name="radio1" value="valor1"> Uvas

<input type="radio" name="radio1" value="valor2"> Peras

<input type="radio" name="radio1" value="valor3"> Higos

<input type="radio" name="radio1" value="valor4"> Mangos

</HTML>

```
String alt =
request.getParameter("radio1");

if (alt.equals("valor1"))
out.println("Ud. Eligió Uvas");

else if (alt.equals("valor2"))
out.println("Ud. Eligió Peras");

else if (alt.equals("valor3"))
out.println("Ud. Eligió Higos");

else
out.println("Ud. Eligió Mangos");
```

Select

Elección entre varias alternativas con pull-down menu

```
<form action="MyServlet"
method="POST">
<select name="colors" size="1">
<option
value="r">Red</option>
<option
value="g">Green</option>
<option
value="b">Blue</option>
</form>
</HTML>
```

Preview

What color do you like?

```
String option =
request.getParameter("colors");

if (option.equals("r"))
option = "red";

if (option.equals("g"))
option = "green";

if (option.equals("b"))
option = "blue";

out.println("Tu elejiste: " +
option);
```

Text Area

Ingreso de texto libre

<h2>Ingrese aquí su opinión</h2>

<TEXTAREA NAME="Ta1" ROWS=10 COLS=40>

Este texto aparece en el campo, pero es editable

</TEXTAREA>

Ingrese aquí su opinion

lo que se escriba no se saldrá en el área
pero se puede editar

Check Box

Parámetros con múltiples valores

¿Qué IDE usas? (BR)

```
<input type="checkbox" name="ide" value="NB"MetBeans (BR) /> <input />
<input type="checkbox" name="ide" value="E" Eclipse /> <input /> (BR)
<input type="checkbox" name="ide" value="JWS"JawaWorkShop /> <input /> (BR)
<input type="checkbox" name="ide" value="J++" /> <input /> (BR)
<input type="checkbox" name="ide" value="Cafe" /> <input />
```

What IDEs do you use?

☒ NetBeans
☒ Eclipse
☐ JawaWorkShop
☒ J++
☐ Cafe'

```
String[] values =
    request.getParameterValues("ide");
for (int i = 0; i < values.length; i++)
    out.println(i+" "+values[i]+"<br>");
```

1- NB
 2- EX
 3- JP

Nombres de parámetros

Recuperar los nombres desde el request

- A veces el programador no puede conocer los nombres de los parámetros
- Por ejemplo cuando la página ha sido generada como resultado de una consulta a una base de datos
- En este caso podemos usar `getParameterNames()` que retornará un objeto de la tipo **Enumeration** conteniendo todos los nombres de los parámetros.
- Enumeration en = request.getParameterNames()

Nombres de parámetros

Iterando una enumeración

- Para recuperar los valores de los parámetros desconocidos podemos usar los métodos `nextElement()` y `hasMoreElement()`

```
Enumeration<String> en = request.getParameterNames();
while(en.hasMoreElements()) {
    String parameterName = en.nextElement();
    String parameterValue =
        request.getParameter(parameterName);
    out.println("parametro "+parameterName+
        "tiene valor "+parameterValue);
}
```

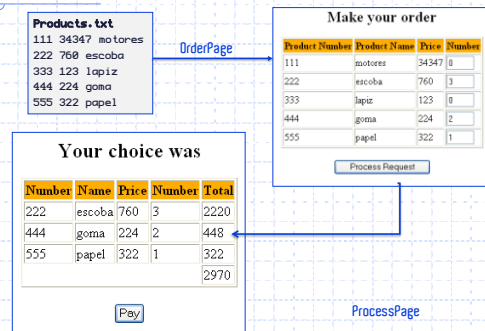
Ejercicio

Productos de un archivo

- La información sobre productos está guardada en un archivo. En cada línea hay un código, precio y descripción separados por un espacio.
- Un primer servlet debe mostrar esto y permitir a los usuarios especificar la cantidad a comprar.
- Después de especificar esto (ingresando números) se aprieta un botón que va a un servlet que muestra el total de la compra

Ejercicio

La idea del workflow



Ejercicio

Una clase para Producto

- Una clase para guardar la información de cada producto.

```
public class Product {
    public String code;
    public int price;
    public String desc;

    Product(String x, String y, int z){
        code = x;
        desc = y;
        price = z;
    }
}
```

Ejercicio

Una clase para los items

```
import java.util.Hashtable;
public class Item {

    static public Hashtable.getItems() {
        BufferedReader in = new BufferedReader(
            new FileReader("Productos.txt");
        Hashtable v = new Hashtable();
        String l;
        while( (l= in.readLine()) != null) {
            int i = l.indexOf(" ");
            int j = l.indexOf(" ", i+1);
            String c = l.substring(0,i);
            String ps = l.substring(i+1,j);
            int pi = Integer.parseInt(ps);
            String d = l.substring(j+1);
            Product p = new Product(c,d,pi);
            v.put(code,p);
        }
        return v;
    }
}
```

Ejercicio

Página con la opción de compra

```
public void doGet(
    out.print("<H2 ALIGN=CENTER> Make your order</H2>\n" +
    "<TABLE BORDER=1 ALIGN=CENTER><TR BGCOLOR=#FFA080> " +
    "<TH>Product Number</TH><TH>Product Name</TH><TH>Price</TH><TH> Number</TH>");
    Enumeration enum = h.getKeys();
    out.print("<form action=ProcessPage method='POST'>");
    while(enum.hasMoreElements()) {
        Product e = (Product)enum.nextElement();
        out.print("<TR>");
        out.print("<TD>" + e.number);
        out.print("<TD>" + e.name );
        out.print("<TD>" + e.price+"<TD>");
        out.print("<input type=text area SIZE=3 "+
        out.print(" name="+e.number+" value=0 >");
    }
    out.println("</TABLE>");
    out.println("<INPUT TYPE='SUBMIT' VALUE='Process'>");
}
```

Se itera sobre los productos

Se agrega el campo de texto para ingresar la cantidad. El nombre del campo será el nombre del producto.

Ejercicio

Página con el resumen de la compra

```
public void doGet( ... request, ... response) throws ... {
    Hashtable items = Item.getItems();
    response.setContentType("text/html");    PrintWriter out = response.getWriter();

    out.print("<H2 ALIGN=CENTER> Your choice was</H2>\n" +
    "<TABLE BORDER=1 ALIGN=CENTER><TR BGCOLOR=#FFA080> " +
    "<TH>Product Number</TH><TH>Product Name</TH><TH>Price</TH><TH> Total</TH>");
    Enumeration en = request.getParameterNames(); int total = 0;
    out.print("<form action=ProcessPayment method='POST'>");

    while(en.hasMoreElements()) {
        String number = (String)en.nextElement();
        String qty = request.getParameter(number);
        int qtyt = Integer.parseInt(qty);
        Product e = (Product)Item.get(number);
        out.print("<TR><TD>" + e.number+"<TD>" + e.name );
        out.print("<TD>" + e.price+"<TD>" + e.price*qtyt);
        total = total + e.price*qtyt;
    }
    out.println("<TR><TD><TD><TD><TD>" + total);
    out.println("</TABLE>");
    out.println("<INPUT TYPE='SUBMIT' VALUE='Process'>");
}
```

Se despliega la cantidad total de \$\$\$ de la compra.

Ejercicio

Variación: se elige 1 producto (1 unidad)

```
<H2 ALIGN=CENTER> Make your order</H2>
<TABLE BORDER=1 ALIGN=CENTER><TR BGCOLOR=#FFAD00>
<TH>Product Number <TH>Product Name<TH>Price <TH> Number
<TR><TD> 1111 <TD> motores <TD> 34347 <TD>
<input type=checkbox name=selection value=1111 >
<TR><TD> 2222 <TD> escoba <TD> 760 <TD>
<input type=checkbox name=selection value=2222 >
<TR><TD> 3333 <TD> lapiz <TD> 1237 <TD>
<input type=checkbox name=selection value=3333 >
<TR><TD> 4444 <TD> goma <TD> 224 <TD>
<input type=checkbox name=selection value=4444 >
<TR><TD> 5555 <TD> papel <TD> 322 <TD>
<input type=checkbox name=selection value=5555 >
```

Product Number	Product Name	Price	Number
1111	motores	34347	<input checked="" type="checkbox"/>
2222	escoba	760	<input type="checkbox"/>
3333	lapiz	1237	<input checked="" type="checkbox"/>
4444	goma	224	<input type="checkbox"/>
5555	papel	322	<input checked="" type="checkbox"/>

Ejercicio

Variación: se elige 1 producto (1 unidad)

```
public void doGet(.. request, ... response) throws ... {
    Hashtable items = Item.getItems();
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print("<H2 ALIGN=CENTER> Your choice was</H2>\n" +
    "<TABLE BORDER=1 ALIGN=CENTER><TR BGCOLOR=#FFAD00>" +
    "<TH>Product Number <TH>PENTER>Product Name<TH>Price <TH> Total");
    String[] values = request.getParameterValues("ide");
    out.print("<form action=ProcessPayment method='POST'>");
    for (int i = 0; i < values.length; i++){
        String number = values[i];
        Product e = (Product)items.get(number);
        out.print("<TR> <TD>" + e.number+"<TD>" + e.name );
        out.print("<TD>" + e.price+"<TD>"+e.price*nqty);
        total = total+e.price*nqty;
    }
    out.println("<TR> <TD> <TD> <TD>"+total);
    out.println("</TABLE>");
    out.println("<INPUT TYPE='SUBMIT' VALUE='Process'>");
}
```

SESSION TRACKING

Problemática

Session Tracking

```
public class CalculaSession extends HttpServlet {
    int op1, op2;

    protected void doGet(request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<h1> Calcula </h1>");
        op1 = (int)(Math.random()*100)+1;
        op2 = (int)(Math.random()*100)+1;
        out.println("<form method=post>");
        out.println("<h2>"+op1+" + "+op2+" = ");
        out.println("<input type=text size=4 name=resultado>");
        out.println("<input type=submit value=corregir>");
        out.close();
    }
}
```

Problemática

¿Porqué esto no funciona?

```
protected void doPost( request, response) throws Exception {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    String res = request.getParameter("resultado");
    int nres = Integer.parseInt(res);
    out.println("<h2>");
    if (nres == op1+op2)
        out.println("Excelente ");
    else
        out.println(" El resultado era "+(op1+op2));
}
}
```

Problemática

Concurrencia de Servlets

- `op1` y `op2` son variables de instancia del servlet.
- Cada vez que un usuario visita el servlet con un `GET` el servlet genera dos números distintos y reemplaza los antiguos
- Durante el tiempo que un usuario recibe respuesta de `doGet` y va a `doPost` otro usuario puede estar llamado a `doGet` y cambiar los valores de `op1` y `op2`
- El `doPost` va a chequear la respuesta con los últimos valores generados por `doGet` (que pueden no ser los originales)
- La respuesta buena podría ser considerada como mala en este caso porque entre medio se cambiaron los valores
- Para evitar esto el servlet debe recordar cuales valores generó para que usuario y usar ellos cuando el usuario esté de vuelta
- Esto se hace con la técnica llamada *Session Tracking*.

Session Tracking

El concepto

- *Session tracking* es un mecanismo que los servlets pueden usar para mantener información acerca del estado de la conversación con el cliente
- Una *sesión* es un dialogo entre una instancia de un browser y el servidor por un cierto período de tiempo.
- Es posible asociar información a una sesión para poder llevar a cabo esto.
- La sesión no es administrada por el servlet sino por el servidor, particularmente por el *Web Container*.

La clase HttpSession

Sus métodos

- `HttpSession session = request.getSession(true)` crea un objeto sesión si no existía antes, si existía lo recupera.
- `session.isNew()` retorna verdadero si el objeto sesión es nuevo
- `session.putAttribute(String nombre, Object valor)` asocia al parámetro nombre el valor valor (value se usa hasta v2.2)
- `Object o = session.getAttribute("nombre")` retorna el objeto asociado al nombre
- `session.removeAttribute("nombre")` borra el par nombre,valor asociado a esa sesión
- `Enumeration[] valores = session.getAttributeNames()`
- `String[] valores = session.ValueNames()` retorna un arreglo/enumeration de los atributos/valores que se han guardado en la sesión
- `long l = session.getCreationTime()` retorna el tiempo (en milisegundos a partir de 1.1.70 0:0:0) de cuando el objeto sesión fue creado
- `Long l = session.lastAccessedTime()` retorna el tiempo en milisegundos de cuando fue accesado por última vez el objeto
- `session.setMaxInactiveInterval(int seconds)` pone (modifica) el timeout de la sesión.

El mismo problema

Servlet con sesiones

```
public class CalculaSession extends HttpServlet {

    int op1, op2;

    protected void doGet(request, response) throws Exception {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(false);
        out.println("<h1> Calcula </h1>");
        op1 = (int)(Math.random()*100)+1;
        op2 = (int)(Math.random()*100)+1;
        session.setAttribute("op1", ""+op1);
        session.setAttribute("op2", ""+op2);
        out.println("<form method=post>");
        out.println("<h2>+op1+ " + "+op2+ " = ");
        out.println("<input type=text size=4 name=resultado>");
        out.println("<input type=submit value=corregir>");
        out.close();
    }
}
```

El mismo problema

Serulet con sesiones

```
protected void doPost( request, response) throws Exception {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession(true);
    String res = request.getParameter("resultado");
    int nres = Integer.parseInt(res);
    op1 = Integer.parseInt((String)session.getAttribute("op1"));
    op2 = Integer.parseInt((String)session.getAttribute("op2"));
    out.println("<h2>");
    if (nres == op1+op2)
        out.println("Excelente ");
    else
        out.println(" El resultado era "+(op1+op2));
}
}
```

¿ Como acumular productos en varias visitas ?

- La idea es que en cada visita se vayan seleccionando productos
- Estos se van acumulando para cada usuario
- Se puede usar un objeto Session para guardad pares
 - ✓ <codigo de producto><cantidad acumulada>
- Despues con cada visita se actualiza el estado de estos pares según lo que se seleccionó

Compra de productos

Serulet original

```
public void doGet( .. request, ... response) throws ... {
    Hashtable items = Item.getItems();
    . . . . .
    . . . . .
    out.print("<form action=ProcessPage method='POST'>");
    while(enum.hasMoreElements()) {
        Product e = (Product)enum.nextElement();
        out.print("<TR>");
        out.print("<TD>" + e.number);
        out.print("<TD>" + e.name );
        out.print("<TD>" + e.price+"<TD>");
        out.print("<input type=textarea SIZE=3 "+
            out.print(" name="+e.number+" value=0 >");
    }
    out.println("</TABLE>");
    out.println("<INPUT TYPE='SUBMIT' VALUE='Process'>");
}
```

Compra de productos

Servlet con session

```
public void doGet(.. request, ... response) throws ... {
    Hashtable items = Item.getItems();
    HttpSession s = request.getSession(true);
    Enumeration en = request.getParameterNames(); int total = 0;
    out.print("<form action=ProcessPayment method='POST'>");
    while(en.hasMoreElements()) {
        String number = (String)en.nextElement();
        String qty = request.getParameter(number);
        int nqty = Integer.parseInt(qty); if (nqty == 0) continue;
        String qtyAntigua = (String)s.getAttribute(number);
        if (qtyAntigua == null)
            s.setAttribute(codigo,cantidad+"");
        else {
            int qtyNueva = Integer.parseInt(qtyAntigua)+nqty;
            s.setAttribute(codigo,nuevaCantidad+"");
        }
        Product e = (Product)item.get(number);
        out.print("<TR><TD>" + e.number+"<TD>" + e.name );
        out.print("<TD>" + e.price+"<TD>"*e.price*nqty);
    }
    out.println("</TABLE>");
}
```

Compra de productos

Servlet con session (2)

```
out.println("</TABLE>\n</BODY></HTML>");
out.println("<br><br><h2> So far you have chosen</h2>"+
    "<TABLE BORDER=1 ALIGN=CENTER>\n" +
    "<TR BGCOLOR=\"#FFDAB9\">\n" +
    "<TH>codigo<TH>cantidad<TH> subtotal");
int total = 0;
Enumeration e = s.getAttributeNames();
while(e.hasMoreElements()) {
    String codigo = (String)e.nextElement();
    out.print("<TR><TD>" + codigo + "<TD>");
    String cantidad = s.getAttribute(codigo);
    int ncantidad = Integer.parseInt(cantidad);
    out.println(ncantidad);
    Product e = (Product)item.get(number);
    out.println("<TD>"*ncantidad*e.price);
    total = total + ncantidad*e.price;
}
out.println("</TABLE><br>");
out.println("<a href='OrderPage'> return to order </a> <br>");
out.println("<a href='CuentaPage'> make order </a>");
}
```

Cookies

Introducción

- Cookies son otra manera de hacer "sesion tracking"
- Por medio de una cookie el servlet puede enviar información al cliente que la guarda y la devuelve cada vez que el browser contacta al mismo servidor de nuevo.
- Servlets mandan cookies al cliente por medio de la información que va en el header, así mismo devuelven los clientes esta información al servidor.
- Los clientes devuelven automáticamente las cookies cada vez que contactan al servidor que se las envió (las use o no) en el header.
- Cookies tienen un nombre y un valor (ambos strings) Adicionalmente pueden guardar un comentario (otro string)
- Un servidor puede mandar más de una cookie.

Cookies

¿Cómo funciona?

- Para mandar cookies
 1. Instanciar (crear) un objeto Cookie
 Cookie c = new Cookie(string, string);
 2. Mandar la cookie
 response.addCookie(c);
- Recuperando las cookies,
 1. Recuperar toda las cookies
 Cookie[] c = request.getCookies();
 2. Recuperar el nombre y el valor
 String name = c[i].getName();
 String value = c[i].getValue();

Ejemplo

Con Cookies

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

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    out.println("<h1> Agregue una cookie </h1> <h2>");
    out.println("<form method=POST>");
    out.println("Nombre : <input type=text name=cnombre>");
    out.println("Valor : <input type=text name=cvalor>");
    out.println("<br> <input type=submit value=enviar>");
    out.println("</html>");
    out.close();
}
```

Ejemplo

Con Cookies

```
protected void doPost( ... response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    Cookie[] c = request.getCookies();
    String cn = request.getParameter("cnombre");
    String cv = request.getParameter("cvalor");
    if (cn != null && cv != null) {
        Cookie nuevacookie = new Cookie(cn,cv);
        response.addCookie(nuevacookie);
        out.println("<h1>La cookie con nombre "+cn+
            " y valor "+cv+" sera mandada</h1>");
    }
    if (c != null) {
        out.println("<h2> ademas las siguientes cookies fueron recibidas </h2>");
        for (int i = 0; i < c.length; i++)
            out.println("<br>Nombre : "+c[i].getName()+" valor : "+c[i].getValue());
    }
    out.print("<form><input type=submit value=retornar>");
    out.close();
}
```

Clase Cookie

Métodos de la clase

- **int getMaxAge()**
 - ✓ Retorna la edad máxima de la cookie, especificada en segundos, por defecto, -1 indica que la cookie persiste hasta que el browser se cierre.
- **String getName()**
 - ✓ Retorna el nombre de la cookie.
- **String getValue()**
 - ✓ Retorna el valor de la Cookie.
- **void setComment(String comment)**
 - ✓ Especifica un comentario que describe el propósito de la cookie.
- **void setMaxAge(int expiry)**
 - ✓ Establece la máxima edad de la cookie en segundos.
- **void setValue(String newValue)**
 - ✓ Le asigna un nuevo valor a la cookie después que la cookie ha sido creada.

Clase Cookie

Cookies o sesiones?

- Con sesiones la información es almacenada en el servidor, esto significa que hay un estado que debe ser administrado cuidadosamente.
- Con cookies es el cliente el que tiene la información, lo que significa que la información viaja y vuelve cada vez que el cliente contacta al servidor.
- El cliente puede prohibir el uso de cookies.
- Las sesiones pueden almacenar mucha más (y mejor) información.
- Las sesiones están implementadas con cookies!!!

Headers (encabezados)

De Request y Response

- Provee información de alto nivel desde el client y hacia el cliente
 - ✓ El request permite al servidor obtener características interesantes del cliente.
 - ✓ El response le permite al Servlet definir cómo la información será entregada al browser.
- En general, pueden ayudar a hacer el diálogo con el cliente más efectivo.
- Para el request, hay métodos llamados `getXXX` o `getHeader(XXX)` para obtener la información.
- Para el response, hay métodos llamados `setHeader(XXX)` o `setXXX` para definir la forma de la información de la respuesta.
- Frecuentemente ambos requieren ser usados en combinación para generar una adecuada respuesta.

La clase Request

Algunos métodos

- **getCookies()**: received the cookies which the client browser may have sent
- **getAuthType()**: is used for clients trying to access a page for which a password is required
- **getRemoteHost()**: to obtain the hostname of the client
- **getMethod()**: to get the name of the method with which the browser contacted the servlet (GET, POST, etc..)
- **getProtocol()**: version of the HTTP protocol the client is using
- **getHeaderNames()**: the name of all the headers the client has sent (is variable depending on the HTTP and browser version)

La clase Request

Algunos getXXX de getHeader("XXX")

- **"Accept"**: qué tipos MIME entiende el cliente.
- **"Accept-Charset"**: Cuál set de caracteres está usando el cliente.
- **"Accept-Encoding"**: Los algoritmos de encoding que está usando el cliente.
- **"Accept-Language"**: Idiomas (en-us, sp, ge, ..)
- **"Authorization"**: Para identificar los clientes con una página protegida.
- **"Host"**: El nombre del computador del cliente.
- **"Referer"**: La URL de la página que generó el contacto.
- **"Cookie"**: Para obtener las Cookies.

La clase Response

Algunos métodos

- **setContentType(XXX)**: para informar el tipo MIME de la respuesta
- **setContentLength(XXX)**: para informar el largo de la respuesta (usado al transmitir archivos)
- **addCookie(c)**: para agregar cookies
- **sendRedirect(XXX)**: para redirigir el request a otra URL.

La clase Request

Algunos getXXX de getHeader("XXX")

- **Content-Type:** Un tipo MIME como "image/gif"
- **Content-Length:** largo (para bytes)
- **Content-Encoding:** codificación
- **Content-Language:** idioma
- **Cache:** como se debe manejar el cache en el cliente (ej, no-cache, no-store, must-revalidate, max-age=xxx,
- **Refresh:** Informa al browser cuán seguido la página debe refrescarse.
- **www-Authenticate:** para administrar páginas protegidas con password.
