Taller de Programación de Internet para Servidor

Cristian Luttgue[1]

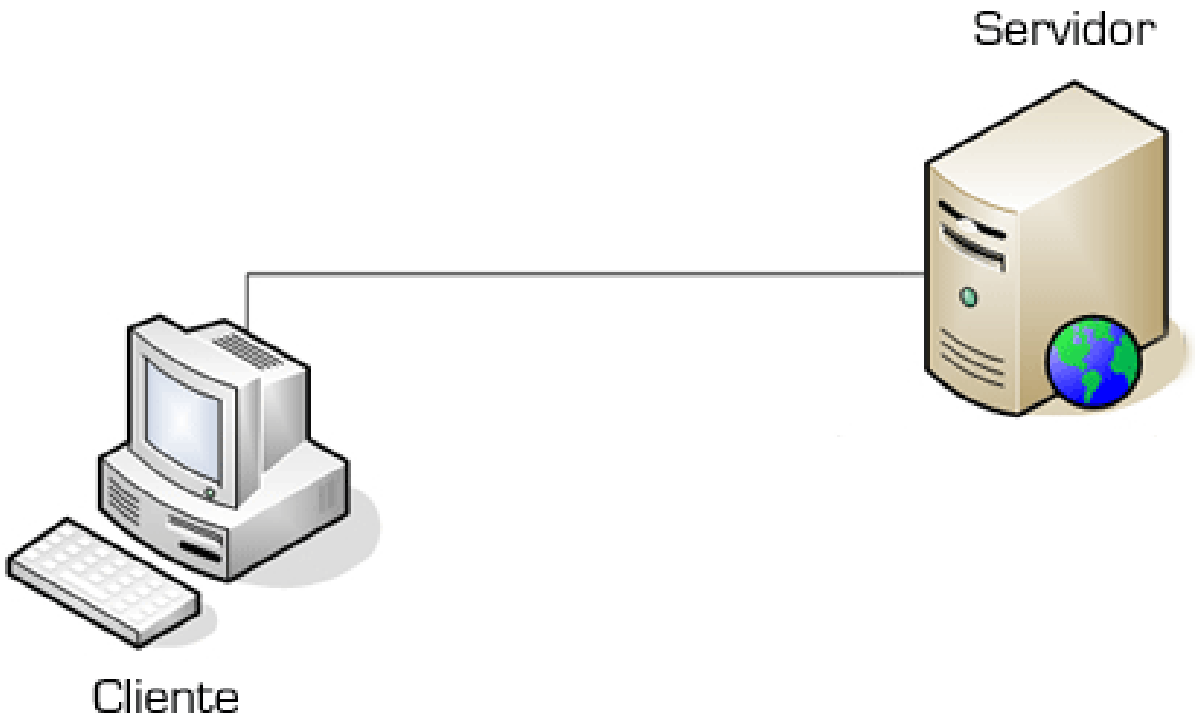
[1]Lic. Cs. de la Ingeniería

cristian.luttgue@gmail.com

Arquitectura Cliente Servidor

Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Arquitectura Cliente Servidor



Arquitectura Cliente Servidor

- Ventajas
- Desventajas

Arquitectura Cliente Servidor

- Ventajas
- Desventajas

Ventajas

- **Centralización del control:** los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).
- **Escalabilidad:** se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Ventajas

- **Centralización del control:** los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).
- **Escalabilidad:** se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Ventajas

- **Fácil mantenimiento:** al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación. Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo.

Desventajas

- Congestión de trafico
- Hardware determinante
- Centralización(¿ambiguo no?)

Desventajas

- Congestión de trafico
- Hardware determinante
- Centralización(¿ambiguo no?)

Desventajas

- Congestión de trafico
- Hardware determinante
- Centralización(¿ambiguo no?)

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

Tecnologías

Las tecnologías que trabajan con esta arquitectura:

- ASP.NET
- Java EE
- PHP
- Python
- Ruby
- etc...

HTTP

El Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) o HTTP es el protocolo para la transferencia de los recursos que componen una web.

- Protocolo orientado a transacciones que sigue un esquema de petición-respuesta entre un cliente y un servidor
- No guarda estado, es decir, no guarda información de la conexión (Recurso a que cliente)

HTTP

El Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) o HTTP es el protocolo para la transferencia de los recursos que componen una web.

- Protocolo orientado a transacciones que sigue un esquema de petición-respuesta entre un cliente y un servidor
- No guarda estado, es decir, no guarda información de la conexión (Recurso a que cliente)

HTTP

El Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) o HTTP es el protocolo para la transferencia de los recursos que componen una web.

- Protocolo orientado a transacciones que sigue un esquema de petición-respuesta entre un cliente y un servidor
- No guarda estado, es decir, no guarda información de la conexión (Recurso a que cliente)

URL

Indican como localizar algún recurso el cual es traducido por el DNS. Su formato es:

protocolo://maquina:puerto/camino/fichero

http://javaHispano.org:4040/ejemplo/inicio.html

URL

Indican como localizar algún recurso el cual es traducido por el DNS. Su formato es:
protocolo://maquina:puerto/camino/fichero
http://javaHispano.org:4040/ejemplo/inicio.html

URL

Indican como localizar algún recurso el cual es traducido por el DNS. Su formato es:
protocolo://maquina:puerto/camino/fichero
http://javaHispano.org:4040/ejemplo/inicio.html

Peticiones HTTP

Método SP URL SP Versión Http CRLF
(nombre-cabecera: valor-cabecera (, valor-cabecera)*CRLF)*
Cuerpo del mensaje

Peticiones HTTP

- **SP** significa espacio en blanco (se dejan por legibilidad)
- **CRLF** significa retorno de carro
- **Parentesis** Los paréntesis que están dentro de los paréntesis indican contenido opcional dentro del propio paréntesis
- Cuando después de un paréntesis hay un *, significa que el contenido del paréntesis puede aparecer repetido cero o más veces.

Peticiones HTTP

Método es el método con que se realiza la petición http (POST O GET normalmente)

Métodos

- **GET:** Devuelve el recurso identificado en la URL pedida.
- **HEAD:** Funciona como el GET, pero sin que el servidor devuelva el cuerpo del mensaje. Es decir, sólo se devuelve la información de cabecera.
- **POST:** Indica al servidor que se prepare para recibir información del cliente. Suele usarse para enviar información desde formularios.
- **PUT:** Envía el recurso identificado en la URL desde el cliente hacia el servidor.

Métodos

- **OPTIONS:** Pide información sobre las características de comunicación proporcionadas por el servidor. Le permite al cliente negociar los parámetros de comunicación.
- **TRACE:** Inicia un ciclo de mensajes de petición. Se usa para depuración y permite al cliente ver lo que el servidor recibe en el otro lado.
- **DELETE:** Solicita al servidor que borre el recurso identificado con el URL.
- **CONNECT:** Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

Método GET

GET
/en/html/dummy?name=MyName&married=not+single&male=yes
HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11)
Gecko/20070312 Firefox/1.5.0.11
Accept:
text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5

Método GET

Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml

Método POST

POST /en/html/dummy HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11)
Gecko/20070312 Firefox/1.5.0.11
Accept:
text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate Accept-Charset:
ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300

Método POST

Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
name=MyName&married=not+single&male=yes

Respuesta HTTP

Versión-http SP código-estado SP frase-explicación CRLF
(nombre-cabecera: valor-cabecera ("," valor-cabecera)* CRLF)*
Cuerpo del mensaje

Códigos de estado HTTP

- Códigos 1xx : Mensajes
  100-111 Conexión rechazada
- Códigos 2xx: Operación realizada con éxito
  - 200 OK
  - 201-203 Información no oficial
  - 204 Sin Contenido
  - 205 Contenido para recargar
  - 206 Contenido parcial

Códigos de estado HTTP

- Códigos 3xx: Redireción
  - 301 Mudado permanentemente
  - 302 Encontrado
  - 303 Vea otros
  - 304 No modificado
  - 305 Utilice un proxy
  - 307 Redirección temporal

Códigos de estado HTTP

- Códigos 4xx: Error por parte del cliente
  - 400 Solicitud incorrecta
  - 402 Pago requerido
  - 403 Prohibido
  - 404 No encontrado
  - 409 Conflicto
  - 410 Ya no disponible
  - 412 Falló precondición

Códigos de estado HTTP

- Códigos 5xx: Error del servidor
  - 500 Error interno
  - 501 No implementado
  - 502 Pasarela incorrecta
  - 503 Servicio no disponible
  - 504 Tiempo de espera de la pasarela agotado
  - 505 Versión de HTTP no soportada

Servidor de aplicaciones

Ejecuta y controla nuestras aplicaciones web.
Util para repartir la carga entre distintos servidores
Es quien recibe efectivamente las peticiones http

Servidor de aplicaciones

Servidores de aplicaciones mas comunes:

- Apache
- Internet Information Service(IIS)
- Glassfish

Declaración de variables

Toda variable en Java antes de ser utilizada **debe ser declarada** con su tipo de dato asociado, en caso contrario ocurre un error en tiempo de compilación. Antes de ver como se declaran datos debemos conocer que tipos de datos se pueden declarar, todo tipo de dato primitivo corresponde a una palabra reservada del lenguaje.

Tipos de datos numéricos

Enteros

- int

Example

```
int x;
x = 0;
```

- long

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Enteros

- **int**

Example

```
int x;
x = 0;
```

- long

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Enteros

- **int**

Example

```
int x;
x = 0;
```

- long

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Enteros

- **int**

Example

```
int x;
x = 0;
```

- **long**

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Enteros

- **int**

Example

```
int x;
x = 0;
```

- **long**

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Enteros

- **int**

Example

```
int x;
x = 0;
```

- **long**

Example

```
long x;
x = 0;
```

Tipos de datos numéricos

Flotante

- float

Example

```
float x;
x = 12.54f;
Tambien es valido x = (float) 12.54;
```

- double

Example

```
double x;
x = 0d;
```

Tipos de datos numéricos
Flotante

- **float**

Example

```
float x;
x = 12.54f;
Tambien es valido x = (float) 12.54;
```

- **double**

Example

```
double x;
x = 0d;
```

Tipos de datos numéricos
Flotante

- **float**

Example

float x;
x = 12.54f;
Tambien es valido x = (float) 12.54;

- double

Example

double x;
x = 0d;

Tipos de datos numéricos
Flotante

- **float**

Example

float x;
x = 12.54f;
Tambien es valido x = (float) 12.54;

- **double**

Example

double x;
x = 0d;

Tipos de datos numéricos
Flotante

- **float**

Example

float x;
x = 12.54f;
Tambien es valido x = (float) 12.54;

- **double**

Example

double x;
x = 0d;

Tipos de datos texto

- char

Example

```
char a;
a = 'c';
```

- string

Example

```
String a;
a = "Hola amiguitos!!";
```

Tipos de datos texto

- **char**

Example

```
char a;
a = 'c';
```

- string

Example

```
String a;
a = "Hola amiguitos!!";
```

Tipos de datos texto

- **char**

Example

```
char a;
a = 'c';
```

- string

Example

```
String a;
a = "Hola amiguitos!!";
```

Tipos de datos texto

- **char**

Example

```
char a;
a = 'c';
```

- **string**

Example

```
String a;
a = "Hola amiguitos!!";
```

Tipos de datos texto

- **char**

Example

```
char a;
a = 'c';
```

- **string**

Example

```
String a;
a = "Hola amiguitos!!";
```

Tipo de datos booleanos

True o False

Se usa para almacenar variables que presenten dos estados, que serán representados por los valores true y false. Representan valores bi-estado, provenientes del denominado álgebra de Boole.

Example

```
boolean reciboPagado;
reciboPagado= false; // ¡¿Aun no nos han pagado?!
```

Tipo de datos booleanos

True o False

Se usa para almacenar variables que presenten dos estados, que serán representados por los valores true y false. Representan valores bi-estado, provenientes del denominado álgebra de Boole.

Example

```
boolean reciboPagado;
reciboPagado= false; // ¡¿Aun no nos han pagado?!
```

Tipo de datos booleanos

True o False

Se usa para almacenar variables que presenten dos estados, que serán representados por los valores true y false. Representan valores bi-estado, provenientes del denominado álgebra de Boole.

Example

```
boolean reciboPagado;
reciboPagado= false; // ¡¿Aun no nos han pagado?!
```

Operadores Aritméticos

- La precedencia de los operadores en Java se presenta de igual manera que en la aritmética corriente.
- En los casos de (*, /,%) se evalúan primero. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.
- En los casos de (+,-) se evalúan después. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.

Operadores Aritméticos

- La precedencia de los operadores en Java se presenta de igual manera que en la aritmética corriente.
- En los casos de (*, /, %) se evalúan primero. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.
- En los casos de (+,-) se evalúan después. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.

Operadores Aritméticos

- La precedencia de los operadores en Java se presenta de igual manera que en la aritmética corriente.
- En los casos de (*, /, %) se evalúan primero. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.
- En los casos de (+,-) se evalúan después. Si hay varios operadores de este tipo se evalúan de izquierda a derecha.

Ejemplos de expresiones algebraicas y de Java

Example

Álgebra $m = \frac{a+b+c+d+e}{5}$

Java $m = (a+b+c+d+e)/5$

Example

¿Como se calculara la siguiente expresión?

$y = 2*5*5+3*5+7$

Ejemplos de expresiones algebraicas y de Java

Example

Álgebra $m = \frac{a+b+c+d+e}{5}$

Java $m = (a+b+c+d+e)/5$

Example

¿Como se calculara la siguiente expresión?

$y = 2*5*5+3*5+7$

Ejemplos de expresiones algebraicas y de Java

Example

Álgebra $m = \frac{a+b+c+d+e}{5}$

Java $m = (a+b+c+d+e)/5$

Example

¿Como se calculara la siguiente expresión?

$y = 2*5*5+3*5+7$

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Algunos ejemplos

- Sumar dos números
- Restar tres números
- Concatenar dos variables
- Operación matemática con variables

Operadores de incremento y decremento

Estos operadores son para simplificar las sumas y restas, específicamente para sumar y restar 1.

Operador	Nombre	Expresion ej.
++	Preincremento	++a
++	Postincremento	a++
–	Predecrecimiento	–b
–	Postdecrecimiento	b–

Operadores de incremento y decremento

Estos operadores son para simplificar las sumas y restas, específicamente para sumar y restar 1.

Operador	Nombre	Expresion ej.
++	Preincremento	++a
++	Postincremento	a++
–	Predecrecimiento	–b
–	Postdecrecimiento	b–

Toma de decisiones
Operadores de igualdad y relacionales

Definition

Los operadores de igualdad y relacionales son utilizados para verificar si una condición es true o false.

Toma de decisiones
Operadores de igualdad

Operador	Operador en Java	Ejemplo	Significado
$=$	$==$	$x == y$	x es igual a y
$\neq$	$!=$	$x != y$	x no es igual a y

Toma de decisiones

Operadores de relación

Operador	Operador en Java	Ejemplo	Significado
$<$	$<$	$x < y$	x es menor que y
$>$	$>$	$x > y$	x es mayor que y
$\geq$	$>=$	$x >= y$	x es mayor o igual que y
$\leq$	$<=$	$x <= y$	x es menor o igual que y

¿Que son las estructuras de selección?

La estructura de selección selectiva permite que la ejecución del programa se bifurque a una instrucción (o conjunto) u otra/s, según un criterio o condición lógica establecida, sólo uno de los caminos en la bifurcación será el tomado para ejecutarse.

if

Es una estructura de selección simple, realiza(selecciona) una acción si la condición es verdadera, o evita la acción si la condición es falsa.

La sintaxis es:

```
if(condición){

    //Aquí ingreso si la condición solo es verdadera!!
}
```

if

Es una estructura de selección simple, realiza(selecciona) una acción si la condición es verdadera, o evita la acción si la condición es falsa.

La sintaxis es:

```
if(condición){

    //Aquí ingreso si la condición solo es verdadera!!
}
```

if

Es una estructura de selección simple, realiza(selecciona) una acción si la condición es verdadera, o evita la acción si la condición es falsa.

La sintaxis es:

```
if(condición){

    //Aquí ingreso si la condición solo es verdadera!!
}
```

if
Ejemplo

Determinar si la calificación de un estudiante es mayor o igual que 40, en dicho caso muestre por pantalla Aprobado. (La nota es solicitada por teclado).

NOTA: ¿Verificar mayor o igual como cadena, en este caso se puede comprobar algo?

if
Ejemplo

Determinar si la calificación de un estudiante es mayor o igual que 40, en dicho caso muestre por pantalla Aprobado. (La nota es solicitada por teclado).

NOTA: ¿Verificar mayor o igual como cadena, en este caso se puede comprobar algo?

if ... else

Realiza una acción si la condición es verdadera, o realiza una acción si la condición es falsa. También es conocida como selección doble.

La sintaxis es:

```
if(condición){

    //Ingreso al ser verdadera la condición!!
}
else{
    //Ingreso al ser falsa la condición
}
```

if ... else

Realiza una acción si la condición es verdadera, o realiza una acción si la condición es falsa. También es conocida como selección doble.
La sintaxis es:

```
if(condición){

    //Ingreso al ser verdadera la condición!!
}
else{
    //Ingreso al ser falsa la condición
}
```

if ... else

Realiza una acción si la condición es verdadera, o realiza una acción si la condición es falsa. También es conocida como selección doble.
La sintaxis es:

```
if(condición){

    //Ingreso al ser verdadera la condición!!
}
else{
    //Ingreso al ser falsa la condición
}
```

if ... else
Ejemplo

Determinar si la calificación de un estudiante es mayor o igual que 40, en dicho caso muestre por pantalla Aprobado. en caso contrario muestre por pantalla el mensaje de reprobado(La nota es solicitada por teclado).

NOTA: ¿Puede existir el caso que no ingrese a ninguno de los dos(if o else)?

if ... else
Ejemplo

Determinar si la calificación de un estudiante es mayor o igual que 40, en dicho caso muestre por pantalla Aprobado. en caso contrario muestre por pantalla el mensaje de reprobado(La nota es solicitada por teclado).

NOTA: ¿Puede existir el caso que no ingrese a ninguno de los dos(if o else)?

if ... else
Ejemplo

- **Anidación de if ... else.**

Crear un algoritmo que verifique si dos números son igual o mayores que cinco en caso contrario escribir por pantalla que los números son menores que cinco.
NOTA: Verificar los posibles resultados.

if ... else
Ejemplo

- **Anidación de if ... else.**

Crear un algoritmo que verifique si dos números son igual o mayores que cinco en caso contrario escribir por pantalla que los números son menores que cinco.
NOTA: Verificar los posibles resultados.

if ... else
Ejemplo

- **Anidación de if ... else.**

Crear un algoritmo que verifique si dos números son igual o mayores que cinco en caso contrario escribir por pantalla que los números son menores que cinco.
NOTA: Verificar los posibles resultados.

switch

Realiza una de entre varias acciones distintas, dependiendo del valor de una expresión.

Su sintaxis es:

```
switch(int o char)
{
    case const1: instrucción(es);
    break;
    case const2: instrucción(es);
    break;
    case const3: instrucción(es);
    break;
    default: instrucción(es);
};
```

switch

Realiza una de entre varias acciones distintas, dependiendo del valor de una expresión.

Su sintaxis es:

```
switch(int o char)
{
    case const1: instrucción(es);
    break;
    case const2: instrucción(es);
    break;
    case const3: instrucción(es);
    break;
    default: instrucción(es);
};
```

switch

Realiza una de entre varias acciones distintas, dependiendo del valor de una expresión.

Su sintaxis es:

```
switch(int o char)
{
    case const1: instrucción(es);
    break;
    case const2: instrucción(es);
    break;
    case const3: instrucción(es);
    break;
    default: instrucción(es);
};
```

switch
Ejemplo

Solicite por pantalla el ingreso de un 1 o un 2. En caso que no se ingresen ninguno de estos números escribir por pantalla “Error en el ingreso”.

NOTA: Verificar con otro tipo primitivo de dato.

switch
Ejemplo

Solicite por pantalla el ingreso de un 1 o un 2. En caso que no se ingresen ninguno de estos números escribir por pantalla “Error en el ingreso”.

NOTA: Verificar con otro tipo primitivo de dato.

¿Que son las estructuras de repetición?

Las estructuras de repetición también llamadas instrucciones de ciclo permiten a los programas ejecutar instrucciones de forma repetida, siempre y cuando una condición(condición de continuación del ciclo) siga siendo verdadera.

while

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
while(condicion)
{
    //Ingreso mientras sea verdadero
}
```

while

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
while(condicion)
{
    //Ingreso mientras sea verdadero
}
```

while

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
while(condicion)
{
    //Ingreso mientras sea verdadero
}
```

do... while

La instrucción do...while realiza la el grupo de acciones uno o mas veces.

Su sintaxis es:

```
Do{
    //Dentro del código
}while(condición);
```

do... while

La instrucción do...while realiza la el grupo de acciones uno o mas veces.

Su sintaxis es:

```
Do{
    //Dentro del código
}while(condición);
```

do... while

La instrucción do...while realiza la el grupo de acciones uno o mas veces.

Su sintaxis es:

```
Do{
    //Dentro del código
}while(condición);
```

do... while

La instrucción do...while realiza la el grupo de acciones uno o mas veces.

Su sintaxis es:

```
Do{
    //Dentro del código
}while(condición);
```

for

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
for(valor inicial; variable de control;
incremento(o decremento)){
    //Instrucción
}
```

for

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
for(valor inicial; variable de control;
incremento(o decremento)){
    //Instrucción
}
```

for

Realiza la acción(o grupo de acciones) en sus cuerpos, cero o mas veces, según la condición de ciclo(verdadera o falsa).

Su sintaxis es:

```
for(valor inicial; variable de control;
incremento(o decremento)){
    //Instrucción
}
```

Ejemplos

Escribir por pantalla 10 veces hola mundo

- Con sentencia **for**
- Con sentencia **while**

Ejemplos

Escribir por pantalla 10 veces hola mundo

- Con sentencia **for**
- Con sentencia **while**

Ejemplos

Escribir por pantalla 10 veces hola mundo

- Con sentencia **for**
- Con sentencia **while**

Ejemplos

- Con sentencia **do while**

Definición de una clase Java

```
public class Prueba{
    //Declaraciones de atributos
    private int a;
    //Declaración de metodos
    public String metodoA(){
        //
    }
}
```

Modificadores de acceso

Los modificadores de acceso son elementos del lenguaje que se colocan **antes** de la definición de variables locales, datos miembros, métodos o clases, los cuales alteran o condicionan el significado del elemento.

modificador definición

Ejemplos:

```
[modificador] tipoVariable nombreVariable;
```

```
[modificador] tipoDevuelto nombreMetodo(listaArgumentos)
```

Modificadores de acceso

Los modificadores de acceso son elementos del lenguaje que se colocan **antes** de la definición de variables locales, datos miembros, métodos o clases, los cuales alteran o condicionan el significado del elemento.

modificador definición

Ejemplos:

```
[modificador] tipoVariable nombreVariable;
```

```
[modificador] tipoDevuelto nombreMetodo(listaArgumentos)
```

Modificadores de acceso

Los modificadores de acceso son elementos del lenguaje que se colocan **antes** de la definición de variables locales, datos miembros, métodos o clases, los cuales alteran o condicionan el significado del elemento.

modificador definición

Ejemplos:

```
[modificador] tipoVariable nombreVariable;
```

```
[modificador] tipoDevuelto nombreMetodo(listaArgumentos)
```

Modificadores de acceso

Los modificadores de acceso son elementos del lenguaje que se colocan **antes** de la definición de variables locales, datos miembros, métodos o clases, los cuales alteran o condicionan el significado del elemento.

modificador definición

Ejemplos:

```
[modificador] tipoVariable nombreVariable;
```

```
[modificador] tipoDevuelto nombreMetodo(listaArgumentos)
```

Modificadores de acceso

Los modificadores de acceso son elementos del lenguaje que se colocan **antes** de la definición de variables locales, datos miembros, métodos o clases, los cuales alteran o condicionan el significado del elemento.

modificador definición

Ejemplos:

```
[modificador] tipoVariable nombreVariable;
```

```
[modificador] tipoDevuelto nombreMetodo(listaArgumentos)
```

Modificadores de acceso

- **public**

Todo el mundo puede acceder al elemento. Si es un dato miembro, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo. Si es un método todo el mundo puede invocarlo

- **private**

Sólo se puede acceder al elemento desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase.

Modificadores de acceso

- **public**

Todo el mundo puede acceder al elemento. Si es un dato miembro, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo. Si es un método todo el mundo puede invocarlo

- **private**

Sólo se puede acceder al elemento desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase.

Modificadores de acceso

- **public**

Todo el mundo puede acceder al elemento. Si es un dato miembro, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo. Si es un método todo el mundo puede invocarlo

- **private**

Sólo se puede acceder al elemento desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase.

Modificadores de acceso

- **public**

Todo el mundo puede acceder al elemento. Si es un dato miembro, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo. Si es un método todo el mundo puede invocarlo

- **private**

Sólo se puede acceder al elemento desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase.

Modificadores de acceso

- **protected**

Es una combinación de los accesos que proporcionan los modificadores public y private. **protected** proporciona acceso público para las clases derivadas y acceso privado (prohibido) para el resto de clases.

- **sin modificador**

Se puede acceder al elemento desde cualquier clase del package donde se define la clase.

Modificadores de acceso

- **protected**

Es una combinación de los accesos que proporcionan los modificadores public y private. **protected** proporciona acceso público para las clases derivadas y acceso privado (prohibido) para el resto de clases.

- **sin modificador**

Se puede acceder al elemento desde cualquier clase del package donde se define la clase.

Modificadores de acceso

- **protected**

Es una combinación de los accesos que proporcionan los modificadores public y private. **protected** proporciona acceso público para las clases derivadas y acceso privado (prohibido) para el resto de clases.

- **sin modificador**

Se puede acceder al elemento desde cualquier clase del package donde se define la clase.

Modificadores de acceso

- **protected**

Es una combinación de los accesos que proporcionan los modificadores public y private. **protected** proporciona acceso público para las clases derivadas y acceso privado (prohibido) para el resto de clases.

- **sin modificador**

Se puede acceder al elemento desde cualquier clase del package donde se define la clase.

Modificadores de acceso

- **static**

Se usa para definir datos miembros o métodos como pertenecientes a una clase, en lugar de pertenecer a una instancia. Es decir, sirve para compartir el valor de una variable miembro entre objetos de una misma clase. Si declaramos una variable miembro de una clase, todos los objetos que declaremos basándonos en esa clase compartirán el valor de aquellas variables a las que se les haya aplicado el modificador static, y se podrá modificar el valor de este desde todas.

Modificadores de acceso

- **static**

Se usa para definir datos miembros o métodos como pertenecientes a una clase, en lugar de pertenecer a una instancia. Es decir, sirve para compartir el valor de una variable miembro entre objetos de una misma clase. Si declaramos una variable miembro de una clase, todos los objetos que declaremos basándonos en esa clase compartirán el valor de aquellas variables a las que se les haya aplicado el modificador static, y se podrá modificar el valor de este desde todas.

Modificadores de acceso

- **final**

Indica que una variable, método o clase no se va a modificar, lo cuál puede ser útil para añadir más semántica, por cuestiones de rendimiento, y para detectar errores.
Si una **variable** se marca como final, no se podrá asignar un nuevo valor a la variable. Si una **clase** se marca como final, no se podrá extender la clase. Si es un **método** el que se declara como final, no se podrá sobreescribir.

Modificadores de acceso

- **final**

Indica que una variable, método o clase no se va a modificar, lo cuál puede ser útil para añadir más semántica, por cuestiones de rendimiento, y para detectar errores.
Si una **variable** se marca como final, no se podrá asignar un nuevo valor a la variable. Si una **clase** se marca como final, no se podrá extender la clase. Si es un **método** el que se declara como final, no se podrá sobreescribir.

Modificadores de acceso

- **final**

Indica que una variable, método o clase no se va a modificar, lo cuál puede ser útil para añadir más semántica, por cuestiones de rendimiento, y para detectar errores.
Si una **variable** se marca como final, no se podrá asignar un nuevo valor a la variable. Si una **clase** se marca como final, no se podrá extender la clase. Si es un **método** el que se declara como final, no se podrá sobreescribir.

Modificadores de acceso

- **abstract**

Es una clase que puede tener herederas, pero no puede ser instanciada. Es decir, solo es declarado los métodos de la clase pero no su implementación.

Modificadores de acceso

- **synchronized**

La palabra reservada synchronized se usa para indicar que ciertas partes del código, (habitualmente, una función miembro) están sincronizadas, es decir, que solamente un subproceso puede acceder a dicho método a la vez.
Cada método sincronizado posee una especie de llave que puede cerrar o abrir la puerta de acceso. Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave entonces el subproceso puede acceder a dicho código sincronizado.

Modificadores de acceso

- **synchronized**

La palabra reservada synchronized se usa para indicar que ciertas partes del código, (habitualmente, una función miembro) están sincronizadas, es decir, que solamente un subproceso puede acceder a dicho método a la vez.
Cada método sincronizado posee una especie de llave que puede cerrar o abrir la puerta de acceso. Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave entonces el subproceso puede acceder a dicho código sincronizado.

Ejemplo

Cree una clase llamada Persona con los atributos nombre, rut y cuente cuantos objetos se crean de dicha clase, además cree una clase derivada llamada Alumno con dos constructores, uno por defecto y otro que posea el parámetro numMatricula, fechaIngreso, nombre.
Posteriormente cambie a final la clase base, ¿que ocurre?

¿Que es Java EE?

Java Platform, Enterprise Edition (Java EE), anteriormente conocido como J2EE, es una plataforma de aplicación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con **arquitecturas N capas distribuidas**

Servidor de aplicaciones

- Glassfish
  - Distribuido por Oracle
  - Código abierto, distribuido bajo licencia GNU/Linux
  - Soporte Java EE 6

Frameworks Java EE

- Spring
- JavaServer Faces
- JPA
- Struts

En su mayoría implementan el patrón arquitectónico MVC

Tipos de Contenedores

¿Que es un contenedor?

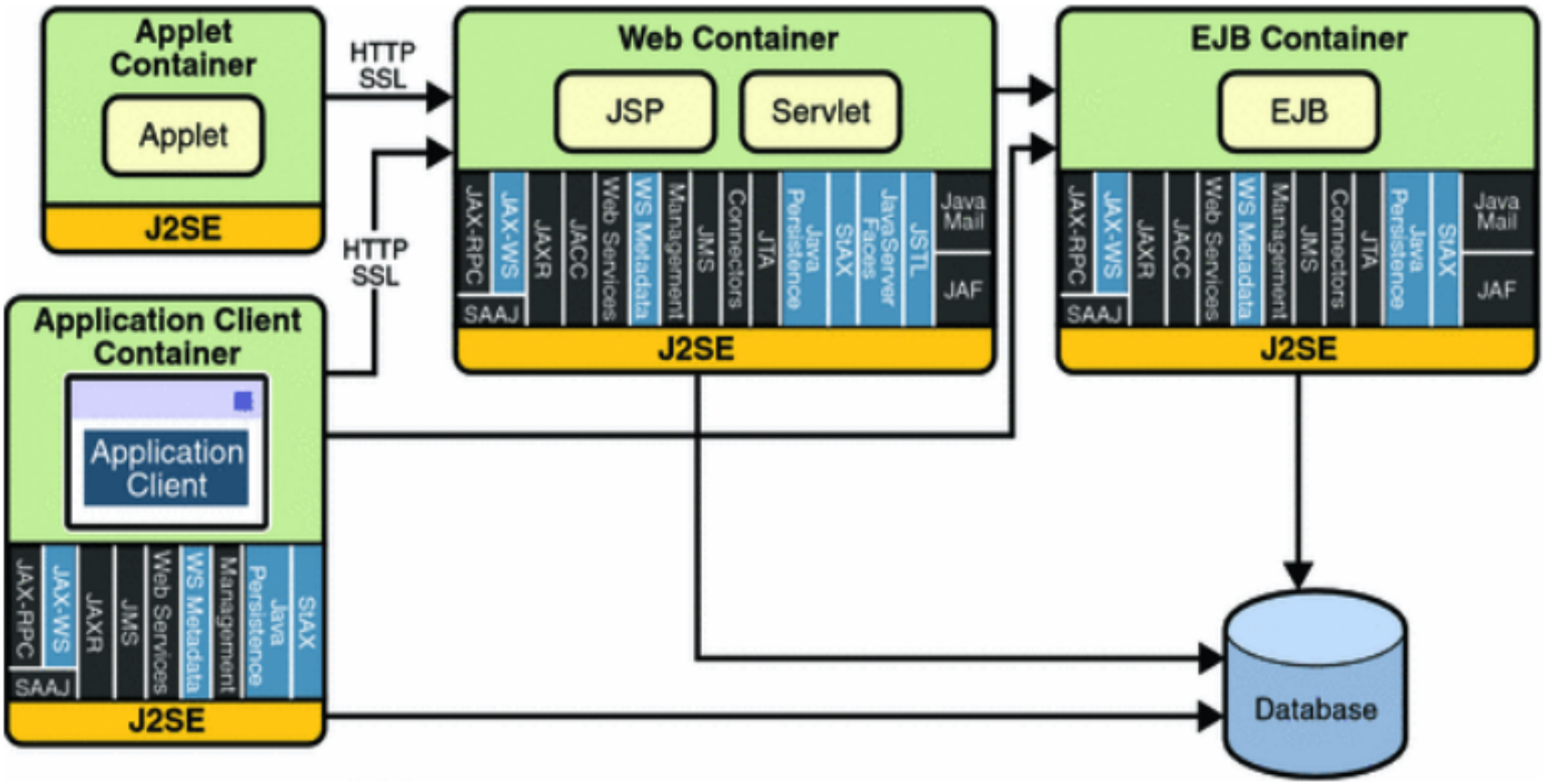
Un contenedor es un entorno de ejecución específico para un conjunto de objetos de un determinado tipo y con unos fines concretos

Tipos de Contenedores

¿Que es un contenedor?

Un contenedor es un entorno de ejecución específico para un conjunto de objetos de un determinado tipo y con unos fines concretos

Tipos de Contenedores



Servlets

Es un componente escrito puramente en Java, en términos Java esto equivale a una Clase y como cualquier otra Clase Java ésta se encuentra compuesta por diversos métodos|funciones.Los tres métodos que debe implementar un Servlet son:

- service (Obligatorio): Este método es la parte medular de todo Servlet ya que dentro de él se incluyen las tareas principales de ejecución.
- init (Opcional) : Es un método ejecutado antes del método service, su labor principal es adquirir/inicializar algún recurso que será empleado por service, estos recursos típicamente son conexiones hacia Bases de Datos.

Servlets

Es un componente escrito puramente en Java, en términos Java esto equivale a una Clase y como cualquier otra Clase Java ésta se encuentra compuesta por diversos métodos|funciones.Los tres métodos que debe implementar un Servlet son:

- service (Obligatorio): Este método es la parte medular de todo Servlet ya que dentro de él se incluyen las tareas principales de ejecución.
- init (Opcional) : Es un método ejecutado antes del método service, su labor principal es adquirir/inicializar algún recurso que será empleado por service, estos recursos típicamente son conexiones hacia Bases de Datos.

Servlets

Es un componente escrito puramente en Java, en términos Java esto equivale a una Clase y como cualquier otra Clase Java ésta se encuentra compuesta por diversos métodos|funciones.Los tres métodos que debe implementar un Servlet son:

- service (Obligatorio): Este método es la parte medular de todo Servlet ya que dentro de él se incluyen las tareas principales de ejecución.

- init (Opcional) : Es un método ejecutado antes del método service, su labor principal es adquirir/inicializar algún recurso que será empleado por service, estos recursos típicamente son conexiones hacia Bases de Datos.

Servlets

- destroy (Opcional) : Ejecutado una vez que ha terminado el método service, su labor es liberar los recursos utilizados/adquiridos en el proceso de ejecución los cuales generalmente son aquellos reservados por init.

Servlets

La clase HttpServlet define (entre otros) dos métodos: doGet y doPost
Si la petición que ha llegado para el Servlet era una petición tipo GET invocará al primer método y si era tipo POST invocará al segundo método.

Creemos un formulario

Creemos un formulario que contenga nombre, rut, teléfono y mostraremos dicha información respondiendo por el servlet.
Mostremos ahora el string de los parámetros en la pagina.

Creemos un formulario

Creemos un formulario que contenga nombre, rut, teléfono y mostraremos dicha información respondiendo por el servlet. Mostremos ahora el string de los parámetros en la pagina.

Creemos un formulario

Cree un formulario que contenga edad, teléfono y muestren la información en el servlet.

¿Si cambio al método POST ocurre algo?

Del formulario anterior se solicita que se muestre el resultado como una tabla.

Repita 5 veces la fila, es decir, que muestre tabulado los mismos datos.

Creemos un formulario

Cree un formulario que contenga edad, teléfono y muestren la información en el servlet.
¿Si cambio al método POST ocurre algo?
Del formulario anterior se solicita que se muestre el resultado como una tabla.
Repita 5 veces la fila, es decir, que muestre tabulado los mismos datos.

Creemos un formulario

Cree un formulario que contenga edad, teléfono y muestren la información en el servlet.
¿Si cambio al método POST ocurre algo?
Del formulario anterior se solicita que se muestre el resultado como una tabla.
Repita 5 veces la fila, es decir, que muestre tabulado los mismos datos.

Creemos un formulario

Cree un formulario que contenga edad, teléfono y muestren la información en el servlet.
¿Si cambio al método POST ocurre algo?
Del formulario anterior se solicita que se muestre el resultado como una tabla.
Repita 5 veces la fila, es decir, que muestre tabulado los mismos datos.

Creemos un formulario

Cree un método que valide que la edad es mayor que 18, en dicho caso muestre por el navegador “Ud. esta autorizado para ingresar a este sitio”, en caso contrario, muestre “Ud. NO esta autorizado para ingresar a este sitio”

Creemos un formulario

Cree un formulario el cual solicite dos números y luego cree en el servlet un método que realice la operación asociada.
Muestre el resultado 50 veces por el cliente.

Creemos un formulario

Cree un formulario el cual solicite dos números y luego cree en el servlet un método que realice la operación asociada.
Muestre el resultado 50 veces por el cliente.

Conexión a base de datos

- Debemos utilizar JDBC para la conexión a base de datos
- Se utiliza un driver de conexión
- Independiente de la plataforma
- Debemos anexar el driver al proyecto

Conexión a base de datos

- Debemos utilizar JDBC para la conexión a base de datos
- Se utiliza un driver de conexión
- Independiente de la plataforma
- Debemos anexar el driver al proyecto

Conexión a base de datos

- Debemos utilizar JDBC para la conexión a base de datos
- Se utiliza un driver de conexión
- Independiente de la plataforma
- Debemos anexar el driver al proyecto

Conexión a base de datos

- Debemos utilizar JDBC para la conexión a base de datos
- Se utiliza un driver de conexión
- Independiente de la plataforma
- Debemos anexar el driver al proyecto

Conexión a base de datos

1. Bajar driver para conexión
2. Cargar el driver en el proyecto
3. Cargar la clase controlador de la BD
4. Crear el objeto de conexión
5. Consultar (SQL)
6. Cerrar conexión

Conexión a base de datos

Cargamos el driver con

- Class.forname("com.mysql.jdbc.Driver")

Posteriormente debemos abrir conexión

- DriverManager.getConnection(url, user,pass);
- Donde la url = jdbc:subprotocolo://dirección:puerto/BD

Estamos en pie de comenzar a enviar consultas y recogerlas en un ResultSet

Conexión a base de datos

Cargamos el driver con

- Class.forname("com.mysql.jdbc.Driver")

Posteriormente debemos abrir conexión

- DriverManager.getConnection(url, user,pass);
- Donde la url = jdbc:subprotocolo://dirección:puerto/BD

Estamos en pie de comenzar a enviar consultas y recogerlas en un ResultSet

Conexión a base de datos

Cargamos el driver con

- Class.forname("com.mysql.jdbc.Driver")

Posteriormente debemos abrir conexión

- DriverManager.getConnection(url, user,pass);
- Donde la url = jdbc:subprotocolo://dirección:puerto/BD

Estamos en pie de comenzar a enviar consultas y recogerlas en un ResultSet

Conexión a base de datos

Cargamos el driver con

- Class.forname("com.mysql.jdbc.Driver")

Posteriormente debemos abrir conexión

- DriverManager.getConnection(url, user,pass);
- Donde la url = jdbc:subprotocolo://dirección:puerto/BD

Estamos en pie de comenzar a enviar consultas y recogerlas en un ResultSet

Conexión a base de datos

Cargamos el driver con

- Class.forname("com.mysql.jdbc.Driver")

Posteriormente debemos abrir conexión

- DriverManager.getConnection(url, user,pass);
- Donde la url = jdbc:subprotocolo://dirección:puerto/BD

Estamos en pie de comenzar a enviar consultas y recogerlas en un ResultSet

Conexión a base de datos

- Debemos sincronizar las consultas con synchronized
- Permite crear una "cola" para que no ocurran problemas de consistencia con los datos
- Debemos sincronizar el objeto del tipo Statement

Conexión a base de datos

- Para realizar consultas sencillas debemos utilizar executeQuery
- Para realizar update, delete e instert utilizamos executyUpdate

Conexión a base de datos

Cree un formulario que solicite un usuario y contraseña, valide si es que la información contenida en una tabla llamada personas (user, pass) es la misma, en dicho caso de permisos de ingreso reenvie a un jsp llamado inicio.jsp (en el cual se muestre un mensaje de bienvenida),

Nota para pasar de una pagina a otra utilizamos response.sendRedirect(nombreDondeQueremosEnviar);

Conexión a base de datos

Cree un formulario de registro que solicite el nombre, fecha de nacimiento, rut y los inserte en la tabla personas
Cree un formulario que modifique los datos de la tabla persona, es decir, genere una tabla en la cual aparezca un botón llamado modificar y reenvíe a otra pagina que contenga los datos y los modifique.

Sesiones y cookies

- El protocolo http contempla una forma de almacenar pequeñas piezas de información en el cliente, piezas de información que el navegador web del cliente nos va a devolver cada vez que realice una petición (cookies)
- Sesiones permiten guardar información cuando estamos en el sitio (Permite asociar al visitante a la pagina)
- Útiles para usar trazas con los sitios

Sesiones y cookies

- El protocolo http contempla una forma de almacenar pequeñas piezas de información en el cliente, piezas de información que el navegador web del cliente nos va a devolver cada vez que realice una petición (cookies)
- Sesiones permiten guardar información cuando estamos en el sitio (Permite asociar al visitante a la pagina)
- Útiles para usar trazas con los sitios

Sesiones y cookies

- El protocolo http contempla una forma de almacenar pequeñas piezas de información en el cliente, piezas de información que el navegador web del cliente nos va a devolver cada vez que realice una petición (cookies)
- Sesiones permiten guardar información cuando estamos en el sitio (Permite asociar al visitante a la pagina)
- Útiles para usar trazas con los sitios

Sesiones
HttpServlet

Sesiones
HttpServlet

Manejo de cookies

Procesar un formulario

Primera pagina Java EE

```
public class Persona {
    private String nombre;


    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

Bibliografía I

https://dl.dropbox.com/u/15868607/Como.Programar.en.Java.-.Deitel.5ta.Edicion.ED.PRENTICE.HALL.zip

S. Alguien.
Sobre esto y aquello.
Revista Esto y Aquello. 2(1):50–100, 2000.