Tema 1 (Parte 4)
Servidor dinámico CGI

J. Gutiérrez

Departament d'Informàtica
Universitat de València

DAW-TS (ISAW).
Curso 14-15

Índice

Servidor dinámico CGI

CGI (Common Gateway Interface) especifica cómo pasar la información desde el servidor Web al proceso que genera la respuesta.

Para crear el servidor dinámico hay que

1. Obtener los parámetros de la petición (ya se pasen en una petición GET o POST).
2. Proporcionar un mecanismo para realizar el *mapping* que asocie una URL con el proceso a ejecutar.
3. Proporcionar un mecanismo para pasar la información desde el servidor web (un proceso) a otro proceso.
4. Lanzar el proceso
5. Obtener la información que genera y darla al cliente como respuesta

Parámetros de la petición

Parámetros en la petición GET

```
GET /ruta/search?p1=v1&p2=v2 HTTP/1.1\r\n
Header1: Value1\r\n
...
\r\n
```

Parámetros en la petición POST

```
POST /ruta/search HTTP/1.1\r\n
Header1: Value1\r\n
...
\r\n
p1=v1&p2=v2
```

Habrá que añadir métodos en la clase `UtilsHTTP` que permitan obtener los parámetros.

Mapping URL - proceso

Este *mapping* se podría realizar en un fichero que contenga la URL y a continuación el proceso a ejecutar.

Además podríamos indicar qué parámetros acepta el proceso y el orden en el que deben aparecer.

```
/ruta1/recurso1;proceso1 arg1 arg2
/ruta2/recurso2;proceso2 arg1 arg2 arg3... argN
```

Con esta información se puede crear un `HashMap`:

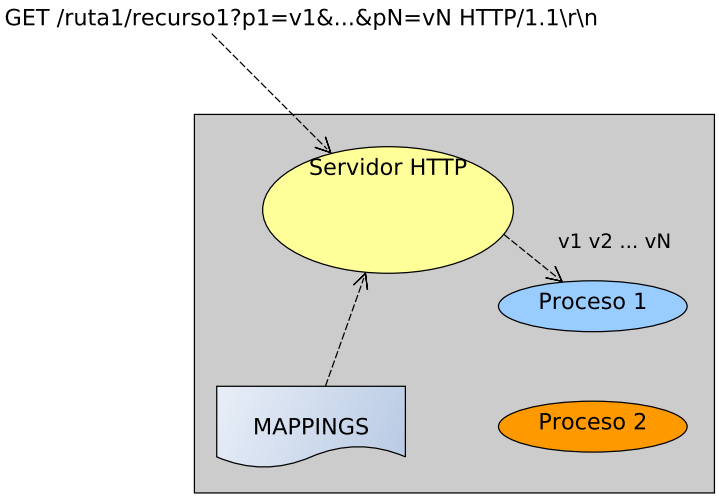
```
HashMap<String,ProcessArgsAndOrder> cgis = ...;
```


Titulo

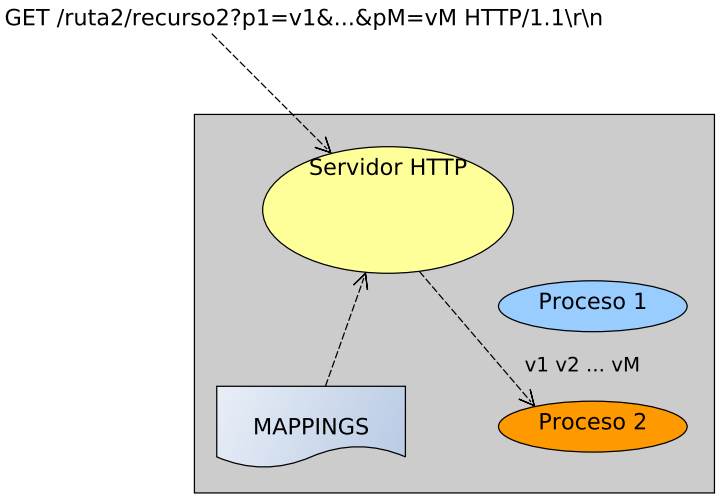
En el código del método `main` del servidor se debe comprobar si la petición va a un CGI o es un recurso estático.

```
String recurso = // Obtención del recurso que se solicita

boolean isCGI = (cgis.get(recurso)!=null);

if (isCGI){
   // Lanzar la tarea que cree el proceso
}
else{
   // Lanzar la tarea que acceda al recurso estático
}
```


Servidor dinámico CGI



Servidor dinámico CGI



Lanzar el proceso

Hay varias clases en el API de Java que permiten lanzar un proceso pasando los argumentos que necesita.

Una de ellas es la clase `ProcessBuilder`

```
class ProcessBuilder{
   // Crea un objeto de este tipo pasando el nombre del proceso y
   // los argumentos en una lista
   ProcessBuilder(List<String> proc){...}

   // Ejecuta el proceso
   public void start(){...}

   // Obtiene un flujo que permite leer de la salida estándar
   // generada por el proceso
   public InputStream getInputStream(){...}
}
```


Obtener la salida que genera el proceso

Suponemos que en el siguiente código el `OutputStream` se corresponde con el flujo de salida del `Socket` que comunica con el cliente.

```
public static void callProcess(OutputStream out, List<String> command)
throws Exception {
// Allows to execute a command with arguments
ProcessBuilder pb = new ProcessBuilder(command);
Process p = pb.start();

// This will allow to read from the output of the process
InputStream in = p.getInputStream();

int dato = 0;
while ((dato = in.read()) != -1)
out.write(dato);
in.close();
out.close();
}
```

Entonces estamos: ejecutando el proceso (pasando los argumentos que hemos obtenido de la petición GET o POST) y estamos leyendo la salida que genera para escribirla hacia el cliente.