REDES DE COMUNICACIONES II

Práctica 1 - Servidor Web

Volver a: Prácticas

2 Funcionalidad

2.3 Ejecución de scripts

El servidor Web soportará, también, la ejecución de scripts en Python y PHP, con el fin de ofrecer la posibilidad de utilizar contenido dinámico. Para ello, el recurso solicitado debe acabar necesariamente en las extensiones .py o .php, respectivamente.

Contenido dinámico

Tradicionalmente, los servidores Web han generado contenido dinámico, en el que la respuesta HTML se genera en tiempo real a través de un script o programa que ejecuta el servidor al recibir la petición, a través de dos métodos básicos, CGI e intérpretes embebidos.

El primero de ellos es un estándar de finales de los 90, ya obsoleto y que no debería utilizarse en nuevos desarrollos, pero que aún sigue en uso en algunas aplicaciones antiguas. Es relativamente sencillo, y lanza un proceso para cada petición, que ejecuta el script, recoge su salida y la devuelve en forma de respuesta al cliente.

Por otro lado, hoy en día cualquier servidor Web serio, soporta la ejecución de lenguajes interpretados, como Python, PHP o Perl, de forma nativa, al tener embebido un intérprete en el propio código del servidor. De esta forma, la ejecución es mucho más rápida y segura que con el estándar CGI.

CGI

De estos dos enfoques, vamos a utilizar el de CGI en la práctica, para simplificarla y hacerla abordable, con alguna modificación relativa a cómo se pasan los parámetros al script. Por ejemplo, el estándar CGI los pasa en una variable de entorno en caso de una petición GET y por la entrada estándar en caso de una petición POST.

Para unificar, en nuestro servidor, tanto en peticiones GET como POST el procedimiento será el siguiente:

1. El servidor leerá los argumentos de la URL o del cuerpo de la petición, en función de si es una petición GET o POST, respectivamente.
2. Ejecutar localmente el script con el intérprete adecuado (python o php), pasándole los argumentos recibidos en su entrada estándar.
3. Leer la salida estándar del script y devolver estos datos en forma de respuesta HTTP. A priori, el servidor no sabe el tamaño de la respuesta del script, que puede ser de cualquier longitud. El script señalizará el final de sus datos con una línea que solo contenga los caracteres '\r\n' (igual que se indica la separación entre cabeceras y cuerpo en una petición HTTP).

Véamos un ejemplo. Imaginemos que el servidor recibe la siguiente petición:

```
GET /scripts/backend.py?var1=abcd&var2=efgh
```

Una vez hechas todas las comprobaciones necesarias, el servidor sabe que debe ejecutar el script 'backend.py' que se encuentra en la ruta relativa 'scripts/', para lo que deberá lanzar un nuevo proceso que se encargue de la ejecución del mismo.

Scripts de prueba

Una manera útil de probar la práctica y entender mejor su funcionalidad será desarrollar, además del servidor, una serie de programas de prueba (en Python o C) que:

- Reciba un petición GET con un argumento que guarde el nombre de un usuario, y responda con la cadena "Hola <<nombre>>!"

- Reciba un petición POST con un argumento que guarde una temperatura en grados Celsius, y devuelva la temperatura correspondiente en Farenheit.

Si se escriben, se debe entregar el código de estos scrips junto con el resto de la práctica, e información sobre cómo ejecutarlos.

Volver a: Prácticas