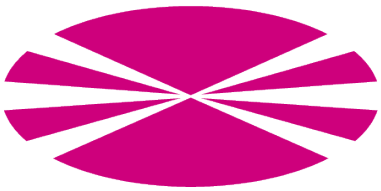

UNIVERSIDADE DA CORUÑA

LABORATORIO DE REDES

Práctica 1: Implementación de un Servidor Web

Implementación de un Servidor Web en Java

El objetivo final será obtener un servidor web capaz de interactuar con un cliente para navegar a través de un sitio web.

Requisitos

Los requisitos para el servidor web son:

- Usar el protocolo HTTP versión 1.0, en donde se generan solicitudes HTTP independientes para cada recurso (por ejemplo, ficheros HTML, imágenes, etc.).
- Hacer que el servidor sea multihilo, para poder procesar múltiples peticiones en paralelo.
- Procesar los métodos GET y HEAD de una petición HTTP.
- Procesar correctamente la cabecera If-Modified-Since de la petición, en caso de ser recibida.
- Generar un fichero de log.
- Responder correctamente a las peticiones a directorios.

El servidor web recibirá y procesará las peticiones HTTP, preparará y enviará un respuesta HTTP. Con respecto a la línea de estado de las respuestas HTTP, el servidor web implementará los códigos siguientes:

- 200 OK
- 304 Not Modified: se enviará como respuesta a una petición que incluya la cabecera If-Modified-Since, siempre que sea necesario.
- 400 Bad Request: la petición no ha sido comprendida por el servidor.
- 403 Forbidden: el servidor entiende la petición, pero rechaza autorizarla.
- 404 Not Found: el recurso solicitado no existe en el servidor.

Respecto a las líneas de cabecera de la respuesta HTTP, el servidor deberá implementar, al menos, las siguientes:

- **Date:** fecha y hora en la que se creó y envió la respuesta HTTP.
- **Server:** especifica el tipo de servidor web que ha atendido la petición. La cabecera Server tendrá como contenido un nombre para el servidor web a elección del estudiante.
- **Content-Length:** indica el número de bytes del recurso enviado.

- **Content-Type:** indica el tipo de recurso incluido en el cuerpo de entidad. Este campo es necesario, ya que la extensión del archivo no especifica (formalmente) el tipo de recurso asociado. Los tipos más comúnmente utilizados son:
  - text/html: indica que la respuesta está en formato HTML.
  - text/plain: indica que la respuesta está en texto plano.
  - image/gif: indica que se trata de una imagen en formato gif.
  - image/png: indica que se trata de una imagen en formato png.
  - application/octet-stream: utilizado cuando no se identifica el formato del archivo.
- **Last-Modified:** indica la fecha y hora en que el recurso fue creado o modificado por última vez.

Para procesar correctamente una petición GET con la cabecera If-Modified-Since, si la fecha de modificación del recurso solicitado es posterior a la especificada en la cabecera, el recurso deberá ser enviado normalmente (código de estado 200 OK). En otro caso, el servidor responderá con un código 304 Not Modified en la línea de estado y no se enviará el recurso solicitado.

Para manejar las fechas en Java es aconsejable leer el siguiente apartado: https://docs.oracle.com/javase/tutorial/datetime/iso/format.html. Se recomienda emplear RFC_1123_DATE_TIME como formato de las fechas (https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html#patterns).

Para poder revisar el comportamiento del servidor, se definirá un fichero de log con todas las peticiones recibidas por el servidor. El fichero se llamará **server_log.txt**. Su contenido no será necesario vaciarlo. Las nuevas entradas se añadirán al final del fichero, y cada entrada tendrá el siguiente formato:

- Línea de petición del mensaje HTTP enviado por el cliente (`Request`).
- Dirección IP del cliente que realiza la solicitud (`Client`).
- Fecha y hora en la que se recibió la petición: [día/mes/año hora:minuto:segundo zona_horaria] (`Date`).
- Código y mensaje de estado que el servidor envía como respuesta al cliente (`Response`).
- Tamaño (en bytes) del recurso enviado al cliente (`Content-Length`).

A continuación, se muestra un ejemplo de posible contenido del fichero:

```
Request: GET /index.html HTTP/1.0 | Client: 127.0.0.1 | Date: 24/01/2025 19:12:33 CET | Response: 200 OK | Content-Length: 216
```

```
Request: GET /notExist.html HTTP/1.0 | Client: 127.0.0.1 |
Date: 24/01/2025 19:38:02 CET | Response: 404 Not Found |
Content-Length: 129
```

El proyecto en github incluye el directorio **p1-files**, que contiene ficheros útiles para el desarrollo de la práctica, como ficheros de ejemplo en los formatos requeridos: index.html, LICENSE.txt, fic.png, udc.gif, etc. Este directorio debe ser el **directorio base** para recuperar los recursos solicitados por los clientes, es decir, se considerará que las rutas de los recursos solicitados son relativas a este directorio. Por su parte, el **working directory** configurado para el servidor web en IntelliJ debe mantener su valor por defecto (directorio java-labs-<team-name>*).*

El servidor web deberá ser capaz de responder ante **peticiones a directorios**. Para ello, definiremos un **fichero por defecto**: index.html. En caso de recibir una petición a un directorio (por ejemplo, petición al directorio / por medio de http://localhost:1111/), se buscará el fichero por defecto bajo el directorio o subdirectorio del que se ha recibido la petición (en el ejemplo, bajo el / del servidor o, lo que es lo mismo, el directorio base). La lógica será la siguiente:

- Si en dicho directorio existe el fichero por defecto (**index.html)**, se devolverá el fichero por defecto.
- Si en el directorio no existe el fichero por defecto, se comprobará el segundo parámetro del programa, que indica si se permiten (valor 1) o no (valor 0) los listados de los directorios:
  - Si el parámetro es 1, se devolverá contenido HTML generado de manera dinámica (sin que exista un fichero con dicho contenido), que mostrará el listado de todos los archivos/directorios existentes dentro del directorio sobre el que se hace la petición, con un enlace que permitirá abrir cada fichero y acceder a cada uno de los subdirectorios, aplicando, en este último caso, la lógica de forma recursiva.
  - Si el parámetro es 0, se mostrará el error 403 (Forbidden).

Bajo el directorio base se proporciona una jerarquía de directorios para poner a prueba la lógica descrita.

Implementación

Para la realización de esta práctica se empleará como base el proyecto facilitado en **https://github.com/GEI-Red-614G010172425/java-labs-<team-name>**. En concreto, se modificarán clases disponibles en el paquete **es.udc.redes.webserver**. Además, el proyecto incorpora un fichero **README.md** con los distintos pasos para la configuración del entorno , en caso de que algún estudiante no lo tenga ya creado.

El servidor web (**es.udc.redes.webserver.WebServer**) debe ser ejecutado desde IntelliJ, recibiendo dos parámetros: el número de puerto en el que

escuchará y un valor 1 o 0 en función de si se permiten o no los listados de directorios.

Para la implementación de la práctica se recomienda una aproximación en dos etapas. En una primera etapa, el servidor multihilo únicamente mostrará por pantalla el contenido de las solicitudes HTTP recibidas. Esto puede ser probado iniciando el servidor en un puerto no reservado (por encima del puerto 1024 (como el 5000, por ejemplo), y usando el navegador para enviar una petición al servidor http://localhost:5000/index.html.

En el segundo paso el servidor debería ser capaz de generar la respuesta HTTP apropiada a la consulta recibida, y enviar el recurso solicitado en caso de ser necesario. Si se produce algún error (por ejemplo, el fichero solicitado no se encuentra), el servidor deberá responder con el código HTTP adecuado para cada situación errónea y una página HTML para el error (en el directorio p1-files/ se proporcionan algunos ficheros HTML de error).

Comprobaciones

En los primeros pasos del desarrollo del servidor web, recomendamos el empleo del **comando nc** para comprobar la correcta ejecución del servidor.

Siguiendo los siguientes puntos se pueden evaluar diferentes aspectos del servidor web:

1. Arrancar un nc al puerto del servidor y dejarlo conectado (para probar que es multithread).
2. Abrir un navegador y cargar una página HTML que incluya texto plano, imágenes gif y png.
3. Si se muestra la página => Multithread OK, GET OK, formatos básicos OK
4. Volver al nc y enviar una petición HEAD (p.e. HEAD /index.html HTTP/1.0).
5. Comprobar que devuelve la línea de estado con las líneas de cabecera => HEAD OK.
6. Comprobar que se envían correctamente las cabeceras Date, Server, Content-Type, Content-Length y Last-Modified => Parámetros cabecera HTTP OK.
7. Arrancar otro nc al puerto del servidor y enviar una petición incorrecta (p.e. GETO /index.html HTTP/1.0): Comprobar que devuelve un error 400 Bad Request + una página de error (con las líneas de cabecera correctas) => Bad Request OK.
8. Enviar una petición a un recurso que no exista (p.e. GET /indeeex.html HTTP/1.0): Comprobar que devuelve un error 404 Not Found + una página de error (con las líneas de cabecera correctas) => Not Found OK.

9. Usando el navegador web, recargar la petición a un recurso (por ejemplo, http://localhost:5000/index.html) y comprobar que el código de respuesta es 304 Not Modified, usando las herramientas de desarrollador disponibles en el navegador.
10. Realizar varias peticiones a distintos directorios del servidor y revisar la respuesta en función de si el directorio contiene o no el fichero por defecto, y del segundo parámetro recibido por el servidor.
11. Comprobar que el fichero de logs contiene todas las peticiones enviadas previamente.

Se proporciona una aplicación Java (**p1-files/httptester.jar**) que permite evaluar parte del funcionamiento básico del servidor web. Esta aplicación se puede ejecutar también desde el IDE, teniendo en cuenta que sus parámetros son <host> <puerto> [<0-9>], en donde <host> es la dirección o el nombre del servidor, <puerto> es el número de puerto en el que corre el servidor, y [<0-9>] es un parámetro opcional que indica el número de test a ejecutar (por defecto se ejecutan los 10 tests).

Evaluación

La programación de sockets en Java supondrá hasta 1,25 puntos en la nota final de la asignatura. Los contenidos de p0 y p1 serán evaluados **en un examen escrito en el grupo de teoría** que corresponda. Cualquier cambio de grupo de teoría deberá solicitarse y justificarse con anterioridad a la fecha del examen. La nota (sobre 10) será ponderada hasta un máximo de 1,25. **La fecha del examen será el martes 18 de marzo de 2025.**