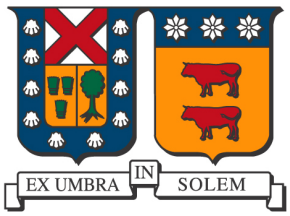


Redes de Computadores
Laboratorio 1
Sockets y HTTP

Profesor: Oscar Encina
Ayudante: Daniel Tapia

Requerimientos

Lenguaje de Programación: Java 1.7/1.8 Preferido o C
Sistemas Operativos: Windows / Linux / MacOSX
IDE: Intelij 14 (Opcional)

Introducción

En la ingeniería de software se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador (Fuente: Wikipedia).
Ustedes han sido contratados por la corporación Cloud Productions para diseñar una solución propietaria de sus servidores, con los cuales planean servir sus aplicaciones web.

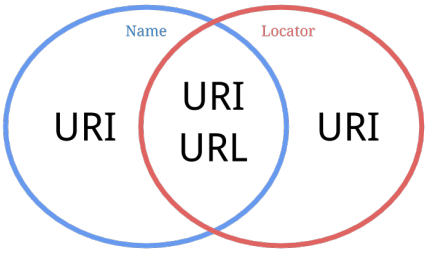
Como requerimiento para poder llevar a cabo este propósito, se realizará la implementación de un servidor, el cual escuchará por medio de un puerto a elección que debe ser configurable y responder con páginas de bienvenida y acerca de la empresa (estáticas). Pudiendo además brindar la posibilidad cualquier cliente de enlazar con el servidor para recibir las respuestas de éste.

Cada URI que el servidor deba responder debe ir asociado a un handler, esto es una clase que se encarga de procesar el HTTP Request, y devolver un HTTP Response.

Cabe destacar la diferencia entre una URL y una URI.
URL : estos son los strings que tienen el protocolo como parte de su estructura, ej:

- **http://www.google.com**
- **ftp://192.168.1.131**

URI: son una compacta secuencia de caracteres que identifica una recurso físico o abstracto.



La diferencia que uno debe tener en cuenta es que si hacemos un paralelo con una persona viviendo en su hogar, URL sería su dirección o como ubicarlo, mientras que la URI seria su nombre o identificador único.

Parte 1: Sockets y Server TCP

La primera parte del laboratorio consiste en programar un servidor que sea capaz de:

- Enlazarse a un socket y escuchar por conexiones entrantes en los puertos a especificar. Dicho servidor deberá retornar un HTML dependiendo del URI que se le otorgue al browser.

- Cuando este servidor inicie se debe crear un Thread Pool

- Por cada conexión entrante asignarle un thread para su ejecución.

  EJ: localhost:8080/about.html deberá retornar una página about.html

- Para lograr este objetivo se necesitará que por cada URI se programe un HTTP Handler para dicho recurso.

  Se deberán programar los siguientes HTTP Handlers:

  1. **GET http://localhost:8080**: Es la página de bienvenida y deberá retornar una página home.html , con un response header 200 (OK)
  2. **GET http://localhost:8080/home_old**: corresponde una página que ya no existe, deberá enviar un response header 301 (Moved Permanently) y redirigir a la página de bienvenida.
  3. **GET http://localhost:8080/secret**: corresponde a una página secreta, ingresos a esta página deberán retornar un response code 403 (Forbidden) si es que no esta autentificado y una página HTML que le advierta al usuario que no puede ingresar ahí. No se les pedirá que manipulen cookies ni sesiones para autentificar al úsuario,

Parte 2: Métodos HTTP

Ahora necesitamos generar un formulario para que nuestros usuarios se puedan autentificar y poder acceder a la página que se esconde detrás de **http://localhost:8080/secret**. Se usará un browser de su elección para visualizar las páginas HTML que retornará su servidor.

Para esta parte del laboratorio se deberá programar el siguiente HTTP Handler:

1. **GET http://localhost:8080/login**: que retorne un formulario y que tenga un usuario y password como campos, luego al hacer click en un botón "Log In", ésta debe hacer un request hacia:
   POST http://localhost:8080/secret y si el usuario y password coinciden con los que el servidor tiene, retornar una página de 200(OK) en vez de 403 (Forbidden).

2. **POST http://localhost:8080/secret:** handler que tiene como propósito la validación para el usuario y password proveniente del formulario **login**, los parámetros de autentificación son los siguientes:

   usuario: root
   password: laboratorio1

   Si las credenciales coinciden se dejará un archivo que le permitirá a la aplicación saber que ya hubo una autentificación aceptada, pueden usar cualquier método que se les ocurra para llevar a cabo esta operación (Ingénienselas!). Al ingresar a **GET http://localhost:8080/secret,** ahora deberá mostrar una página HTML con el contenido "AUTENTIFIACIÓN COMPLETADA"

Es decir, si bien no pueden acceder a **GET http://localhost:8080/secret** ya que siempre retornará un 403 si es que no están autentificados, la forma de autentificarse es mandando el formulario a **POST http://localhost:8080/secret** y éste dejará un archivo que le permitirá a la aplicación saber que ya hubo una autentificación aceptada.

Bonus

Se le debe agregar una funcionalidad extra al servidor, ésta es la de registrar que IP hizo el request, por cada conexión, se deberá escribir en un archivo log.txt una fila con la IP y la URL consultada.

Restricciones

- No se puede usar ninguna implementación de java que generan servidores.
  Ej: HttpServerImpl

- Pueden usar la interface HttpServer, e implementar sus métodos como guía, como también crear sus propias interfaces e implementaciones.

- No se pude usar ninguna implementación de java para la generación del ThreadPool.

  Ej: Executor, ExecutorService.

  Ahora sin embargo, pueden ver esas implementaciones e intentar entender cómo funcionan por debajo.

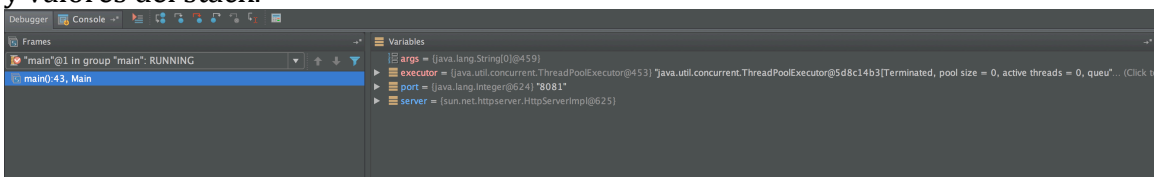
Consideraciones

- Se debe trabajar en grupos de a lo más dos personas.
- La entrega es por Moodle.
- Las dudas o aclaraciones pueden preguntarlas por Moodle.
- La entrega debe realizarse en un ZIP y debe llevar el nombre Tarea1RDC-RolIntegrante-1-RolIntegrante-2.
- Debe ir dentro del ZIP un archivo README.txt que debe contener nombre y rol de los integrantes del grupo y cualquier supuesto que se haya tomado durante la realización de la tarea, y un instructivo de cómo compilar el código adjunto.
- La entrega será usando un método que será informado a la brevedad y el plazo máximo de entrega es hasta el 02 de Abril de 2015 a las 23:59.
- Por cada día de atraso se descontarán 20 puntos.
- Las copias serán evaluadas con nota 0.

Herramientas

InteliJ14 (IDE)

En InteliJ14 existe la opción de compilar en modo debug, esto hace que la aplicación sea capaz de parar la ejecución del programa para poder ver los valores de variables y valores del stack.



Nmap (Tool)

Herramienta de network scanning, esto sirve para poder confirmar si efectivamente su aplicación está enlazándose correctamente los puertos.

```
~ $ nmap localhost -p 8080

Starting Nmap 6.47 ( http://nmap.org ) at 2015-03-14 19:06 CLST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000087s latency).
PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
~ $
```