Guía #1: Trabajando con el protocolo HTTP

Introducción

Como hemos discutido en la sesión anterior, el protocolo HTTP es muy utilizado por una gran cantidad de aplicaciones en la Internet. En esta guía continuamos explorando los diferentes elementos que conforman este protocolo.

Requerimientos

Para poder llevar a cabo necesitamos:

1. Instalar Python de acuerdo con el sistema operativo que tengamos.

Una vez que hayamos finalizado con la instalación, podemos abrir una consola y ejecutar el siguiente comando para verificar la versión de Python instalada:

```
:~$ python --version
```

Procedimientos

A continuación se detallan los procedimientos que se deben llevar a cabo en este ejercicio:

Los métodos HTTP

Recordamos que HTTP consta de dos elementos principales en cuanto a la comunicación cliente-servidor, que son la petición y respuesta. Sin embargo, HTTP proporciona más acciones que podemos realizar, a los que llamamos métodos. El método más común (el que va incluido en una petición) es el GET. Existen otros métodos como son:

- HEAD
- POST
- PUT
- DELETE
- OPTION
- TRACE
- CONNECT
- PATCH

Vamos a ver cómo funcionan uno de los métodos anteriores. Para el resto, deben buscar qué significa cada uno y escribirlo en una línea.

El método HEAD

Este método es bastante similar al GET, solo que con HEAD el servidor no incluye el cuerpo en la respuesta, aún cuando haya un recurso válido en la URL solicitada. Para poder ver cómo funciona el método entonces vamos a trabajar con una librería:

```
~:$ python
```

Con lo anterior podemos entrar a la consola interactiva, entonces podemos proceder a utilizar un módulo de la librería e importar funciones del módulo:

```
>>> from urllib.request import urlopen, Request
>>>  peticion= Request('http://www.google.com', method= 'HEAD')
>>> respuesta= urlopen(peticion)
>>> respuesta.status
200
>>> respuesta.read()
b''
>>>
```

Ahora vamos a repetir el paso, pero lo haremos con el recurso mostrado en la sesión de clase, la página de nuestra universidad:

```
>>>  peticion2= Request('http://www.unan.edu.ni', method= 'HEAD')
>>> respuesta= urlopen(peticion2)
>>> respuesta.status
200
>>> respuesta.read()
```

¿Qué observa de diferente en este resultado con respecto al primero? Intente explicar por qué el método HEAD no funciona en el segundo ejercicio.

Manejo de cookies

HTTP se le conoce como un protocolo sin estado. Esto es porque no tiene algún mecanismo para que un servidor pueda determinar si dos peticiones vienen del mismo cliente. Aquí es donde entran en juego las cookies. Una cookie es un pequeño elemento de información que el servidor envía como parte de la respuesta a la petición del cliente. El cliente guarda las cookies localmente y son incluidas entonces en peticiones posteriores. De esta forma es que el servidor puede identificar que se trata del mismo cliente.

Lo primero que vamos a hacer es crear un espacio donde almacenar las cookies que serán enviadas desde el servidor. Así, podemos inspeccionar el contenido de las cookies, entre otras acciones. En la misma sesión de consola abierta en el procedimiento anterior:

```
>>> from http.cookiejar import CookieJar
>>> cookies_guardar= CookieJar()
```

Luego de haber definido donde se guardarán las cookies entonces procedemos a crear un extractor, es decir, tomará las cookies y las almacenará donde hemos definido.

```
>>> from urllib.request import build_opener, HTTPCookieProcessor
>>> extractor= build_opener(HTTPCookieProcessor(cookies_guardar))
```

Con esto podemos hacer la petición HTTP para que el servidor nos envíe las cookies. Vamos a hacer una petición al sitio donde se han subido los recursos didácticos de la clase:

```
>>> extractor.open('http://www.github.com')
```

Entonces podemos consultar las cookies que nos han enviado el servidor:

```
>>> len(cookies_guardar)
```

Escriba el resultado de este paso.

Para finalizar este ejercicio vamos a examinar algunas de las propiedades de las cookies que nos ha enviado el servidor. Para ello, vamos a crear una lista a partir de ellas y luego 'leemos' el contenido:

```
>>> lista_cookies= list(cookies_guardar)
>>> lista_cookies
```

El resultado de este paso nos muestra el contenido de las cookies, y tambien podemos observar que estas contienen dos partes (apertura y cierre de corchetes nos indica el inicio y fin de cada elemento). Como las cookies fueron guardadas en una lista, podemos acceder cada elemento utilizando el subíndice:

```
>>> lista_cookies[0].name

>>> lista_cookies[0].domain

>>> lista_cookies[0].expires
```

En el caso de la última propiedad, su valor es una fecha ¿En qué formato esta representado este?