université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

20 octobre 2021

Web 1
Agrégation d'Informatique

TP n° 2

Le but de ce TP est d'écrire de petits serveurs Web en utilisant le module `http.server` de Python. Ces derniers sont là uniquement pour illustrer les concepts du cours :

— passage de paramètres
— *Cookies*
— Sessions HTTP
— architecture REST

En particulier, ce n'est pas un bon *framework* pour faire des applications Web modernes. On utilisera plutôt des *framework* éprouvés (par exemple *Django* en Python, Java/JSP en Java, OCsigen en OCaml, ...) pour créer des applications Web réalistes.

1 Les classes `HTTPHandler` et `HTTPServer`

En Python, un serveur Web peut s'écrire assez simplement de la façon suivante :

```
1  import http.server
2  import socketserver
3
4  #Petit wrapper pour qui crée un serveur TCP/IP permettant d'être interrompu
5  #en pressant CTRL-C dans la console
6  class MyHttpServer(socketserver.TCPServer):
7
8      def server_bind(self):
9          self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
10         self.socket.bind(self.server_address)
11
12     def serve_until_interrupted(self):
13         try:
14             self.serve_forever()
15         except KeyboardInterrupt:
16             self.shutdown()
17         finally:
18             self.server_close()
19
20
21 if __name__ == "__main__":
22     HTTPD = MyHttpServer(("localhost", 8080), http.server.SimpleHTTPRequestHandler)
23     HTTPD.serve_until_interrupted()
```

Le fonctionnement est le suivant. Lorsqu'une connection TCP est établie sur le port 8080, le contenu du message est traîté par le code de la classe `SimpleHTTPRequestHandler`. Cette dernière dérive de `BaseHTTPRequestHandler` et possède plusieurs attributs et méthodes utiles, documentées ici :

`https://docs.python.org/3/library/http.server.html#http.server.BaseHTTPRequestHandler`

1.1 Framework Web, à la main

Créer une classe `CustomHTTPRequestHandler` (ainsi que des classes auxiliaires) qui permette d'avoir le comportement suivant (on pourra rajouter incrémentalement chacune de ces fonctionalités) :

— redéfinit la méthode `do_GET`. Si la ressource demandée est un chemin existant, alors on appelle simplement `super().do_GET()`. Le comportement sera alors de renvoyer le fichier en HTTP. Sinon, la fonction décode l'URL (utiliser la fonction `parser_url` du module `urllib.parse`) et recherche dans la variable globale `HTTP_ACTION` (supposée contenir un dictionnaire) une entrée qui correspond au chemin demandé. Si cette entrée existe, on suppose qu'elle est associée à une fonction attendant quatre paramètres : l'objet `self`, le chemin demandé, les paramètres de requête et un booléen qui vaut `True` si la requête est de type `GET`. Ainsi, si on accède à l'URL `http://localhost:8080/test?param=1&foo=bar`, on s'attend à ce que la méthode `do_GET` appelle

```
HTTP_ACTION['/test'](self, '/test', {'param': '1', 'foo': 'bar'}, True)
```

Si aucune action n'est disponible, le serveur renvoie une erreur 404 et un petit fichier HTML contenant un message d'erreur.

— modifier la classe `CustomHTTPRequestHandler` pour que cette dernière dispose de deux méthodes `getCookies()` renvoyant tous les cookies sous forme d'un dictionnaire et `setCookie(n, v, a)` qui permette de positionner un cookie de nom n, de valeur v et d'expiration a. On utilisera l'attribut `headers` et on redéfiniera la méthode `end_headers()`

— Une fois les cookies mis en place, ajouter un mécanisme de session. Ce dernier sera implémenté par un dictionnaire globale `HTTP_SESSION` et une méhode `getSession()` sur la classe `CustomHTTPRequestHandler`. Cette dernier renvoie un dictionnaire associé à la session (éventuellement vide si la session vient d'être crée).

2 Applications

Se servir de la classe `CustomHTTPRequestHandler` pour créer des serveurs répondants aux spécifications suivantes :

1. Un compteur : le serveur supporte une unique ressource `/count` qui renvoie la valeur d'un compteur. Ce dernier est incrémenté à chaque rechargemde la ressource. Le compteur doit être stocké dans la session. On vérifiera en lançant une fenêtre de navigation privée et en constante que l'on a des compteurs différents.
2. Une *todo* list. Le serveur propose une page HTML contenant un formulaire permettant de créer une page avec une liste de choses à faire, stockée dans la session.
   — L'URL `/reset` réinitialise la liste.
   — L'URL `/add?todo=tache` ajoute une tâche à la liste.
   — L'URL `/remove?id=n` supprime la tâche d'identifiant n.