Fiche

Exemple d'utilisation de `HttpServer`

Cette fiche présente comment développer un serveur HTTP en utilisant la classe Java `HttpServer`

Les différentes classes présentées ici correspondent à un serveur HTTP basé sur le serveur HTTP intégré au JDK. Une fois exécuté, le serveur peut être accédé directement depuis un navigateur via l'adresse `http://localhost:8080/index`. À noter que la page `index` est un contexte correspondant à l'exécution d'un code *Java*.

1 La classe principale

Le programme serveur doit d'abord construire un objet `HttpServer` en spécifiant le port d'écoute. Ici, nous choisissons le port 8080 (le port 80 est bloqué dans certains cas).

```
HttpServer serveur = null;
try {
  serveur = HttpServer.create(new InetSocketAddress(8080), 0);
} catch(IOException e) {
  System.err.println("Erreur lors de la création du serveur " + e);
  System.exit(0);
}
```

Une fois le serveur créé, il faut ajouter des contextes associés à des *handlers*. Un contexte est un URL : lorsque le serveur reçoit une requête HTTP sur cet URL, le code du *handler* associé est exécuté. Ici, nous créons un contexte `index` auquel nous associons le *handler* nommé `AccueilSimpleHandler`. La méthode `start` démarre le serveur.

```
serveur.createContext("/index", new AccueilSimpleHandler());
serveur.setExecutor(null);
serveur.start();
```

2 La classe correspondant au handler

L'objectif du *handler* est de récupérer les données éventuelles envoyées dans la requête HTTP et d'y répondre en créant une ressource (HTML, image, JSON, etc.). Ici, nous nous contentons de retourner du code HTML qui contient les données qui ont été envoyées en GET et en POST. La fonction principale du *handler* est la méthode `handler` :

```
public void handle(HttpExchange t) {
  ...
}
```

Pour récupérer les données envoyées en GET, nous utilisons la méthode `getRequestURI` sur l'objet `HttpExchange`.

```
URI requestedUri = t.getRequestURI();
String query = requestedUri.getRawQuery();
```

Pour récupérer les données envoyées en POST, nous utilisons la méthode `getRequestBody` sur l'objet `HttpExchange`. Nous utilisons un flux de type `InputStreamReader` encapsulé dans un flux `BufferedReader`.

```
BufferedReader br = null;
try {
  br = new BufferedReader(new InputStreamReader(
      t.getRequestBody(),"utf-8"));
} catch(UnsupportedEncodingException e) {
  System.err.println("Erreur lors de la récupération du flux " + e);
  System.exit(0);
}

try {
  query = br.readLine();
} catch(IOException e) {
  System.err.println("Erreur lors de la lecture d'une ligne " + e);
  System.exit(0);
}
```

Le *handler* peut ensuite envoyer l'en-tête HTTP. Nous récupérons le flux de sortie avec la méthode `getResponseHeaders`. Nous pouvons spécifier le type MIME des données de retour (ici du HTML) puis nous utilisons la méthode `sendResponseHeaders` pour envoyer l'en-tête (ici code 200 et la taille des données). La variable `response` contient le code HTML généré dans le *handler*.

```
try {
  Headers h = t.getResponseHeaders();
  h.set("Content-Type", "text/html; charset=utf-8");
  t.sendResponseHeaders(200, reponse.getBytes().length);
} catch(IOException e) {
  System.err.println("Erreur lors de l'envoi de l'en-tête : " + e);
  System.exit(0);
}
```

Les données sont envoyées à l'aide du flux de sortie que l'on récupère sur l'objet `HttpExchange` à l'aide de la méthode `getResponseBody`.

```
try {
  OutputStream os = t.getResponseBody();
  os.write(reponse.getBytes());
  os.close();
} catch(IOException e) {
  System.err.println("Erreur lors de l'envoi du corps : " + e);
}
```

3 Fichier HTML pour tester le serveur

Pour tester le serveur, il est possible d'utiliser un simple navigateur et d'aller sur la page `http://localhost:8080/index`. L'autre solution est d'utiliser le fichier HTML suivant qui permet d'envoyer des données soit en GET, soit en POST sur cette même adresse (n'oubliez pas de changer le champ action du formulaire en cas de modification du numéro de port).

```
<!DOCTYPE html>
<html lang='fr'>
  <head>
    <title>Formulaires pour tester le serveur Http simple</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
  </head>
  <body>
    <h1>Formulaires pour tester le serveur Http simple</h1>

    <form method="get" action="http://localhost:8080/index.html">
      <label>Nom</label><input type="text" name="nom"/>
      <label>Prénom</label><input type="text" name="prenom"/>
      <button>Valider en GET</button>
    </form>

    <form method="post" action="http://localhost:8080/index.html">
      <label>Nom</label><input type="text" name="nom"/>
      <label>Prénom</label><input type="text" name="prenom"/>
      <button>Valider en POST</button>
    </form>
  </body>
</html>
```

4 Compilation et exécution

Dans un premier temps, compilez les deux classes. Ensuite, exécutez le serveur qui se mettra alors en attente. Vous pouvez ensuite ouvrir le fichier HTML via un navigateur ou bien saisir directement l'URL `http://localhost:8080/index` dans un navigateur.