université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 13 mai 2022

Services et applications Web
M1 ISD

TP Noté

durée 2h.

Consignes Ce TP est noté sur 20 points. Il doit être rendu exclusivement au moyen de l'interface WEB dont le lien est donné sur la page du cours. Merci d'utiliser toujours la même adresse email pour soumettre votre projet, de préférence votre adresse u-psud. Si vous n'utilisez pas votre adresse mail `universite-paris-saclay.fr` merci d'indiquer dans le fichier `reponses.txt` vos noms et prénoms. Tous les documents (y compris l'accès internet) sont autorisés.

1 Service de redirection (17 points)

Le but de cet exercice est de créer un service de redirection qui permet d'associer à des URLs plus longues des URLs de tailles fixes [1].

Il est conseillé de remplir le code dans l'ordre de l'énoncé afin que le site se mette à fonctionner progressivement.

Architecture générale du site

Le point d'entrée de notre site est un fichier `index.jsp` provoquant l'affichage de la figure 1.a. On y trouve trois formulaires ayant chacun une action particulière.

Le premier formulaire attend une URL dans son champ de texte. En cliquant sur « `Générer un lien` », le site affiche (figure 1.b) :

— Un *hash* (entier entre 0 et $2^{31} - 1$)
— Un *mot de passe* (chaîne de longueur au plus 8 représentant un entier hexadécimal)
— Un lien vers le servlet `GoServlet` de notre site, passant le *hash* comme un paramètre `GET h`

Le second formulaire demande un `hash` et un `mot de passe`. Sur saisie de ces derniers, et si une telle paire de *hash* et *mot de passe* sont associés à une URL précédemment saisie, alors cette URL est oubliée et un message de succès est affiché (figure 1.c). Si une telle URL n'existe pas, un message d'échec est affiché (1.d).

Le troisième formulaire attend un mot de passe d'administrateur. Si ce dernier est correct, il provoque l'affichage de toutes les URLs avec leur hash, le nombre de fois où elles ont été chargées et leur mot de passe (dans cet ordre) dans une liste HTML non énumérée (`ul/li`) séparés par des virgules (figure 1.e).

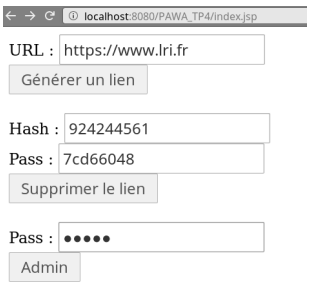
Toutes les informations sont stockées dans une table `URL_`$toto$:

```
CREATE TABLE URL_TOTO(url TEXT, hash INTEGER,
                      count INTEGER, pass VARCHAR(8), PRIMARY KEY (url, hash))
```

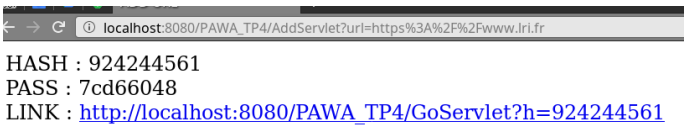
1.1 Génération de liens (7 points)

On s'intéresse ici aux classes `URLDB`, `AddServlet` et au fichier `WEB-INF/jsp/add_view.jsp`. On note que le constructeur de la classe `URLDB`, en plus de créer une connexion crée une table `URL_FOO` si elle n'existe pas, avec le schéma décrit précédemment.

1. Dans notre cas, les URLs générées ne seront pas si courtes que ça car elles seront de la forme : `http://localhost:8080/PAWA_TP4_ACOMPLETER/GoServlet?h=11231243`, mais on peut imaginer acheter un nom de domaine court et déployer ce projet à la racine du serveur qu'on y aura installé.



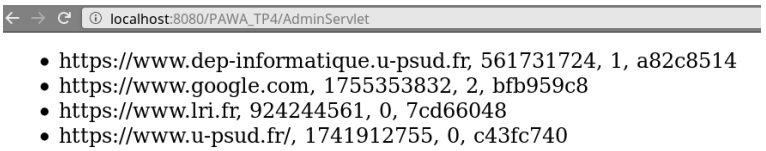
(a) `index.jsp`



(b) `Ajout d'URL`

(c) `Suppression, succès`

(d) `Suppression, échec`



(e) Page d'administration

Figure 1 – Interface graphique du site

Questions

1. (0 point, **mais très important, si vous ne le faites pas rien ne marche**) modifier l'attribut `SUFFIX` de la classe `URLDB` pour qu'il soit votre nom de famille, sans espaces, sans accents, sans tirets et en majuscule. Une fois que vous avez choisi un tel suffixe, ne plus le modifier pour toute la durée du TP.
2. (1 point) Compléter la méthode statique `URLDB.getFromSession`. Cette dernière prend en argument une session HTTP (de type Java `HttpSession`). Si une variable de session `"db"` contient un objet de type `URLDB` alors ce dernier est renvoyé. Sinon un nouvel objet `URLDB` est créé et stocké dans la session sous le nom `"db"` puis renvoyé.
   Note dans tous les *Servlets*, vous utiliserez cette méthode auxiliaire pour récupérer une instance de `URLDB`.
3. (1.5 point) Compléter la fin de la méthode `URLDB.insertURL`. Cette dernière prend en argument une URL et clacule un *hash* et un *mot de passe* (déjà fait), puis les insère dans la base, avec un compteur valant 0, puis renvoie un objet `URLInfo` décrivant les valeurs insérées.
   Note On rappelle que la syntaxe du `INSERT` est :

   ```
   INSERT INTO T VALUES (v1, v2, ..., vn)
   ```

   où `T` est le nom de la table et `vi` les valeurs à insérer. Les chaînes de caractères sont délimitées par des « ' » en SQL. Vous veillerez à utiliser l'attribut statique `TABLE` qui contient le nom de votre table (avec le suffixe).
4. (2.5 points) Compléter la méthode `doGet` de la classe `AddServlet`. Cette dernière doit récupérer le paramètre `url` passé par le formulaire. Si la valeur de ce paramètre ne commence pas par `http://` ou `https://`, alors le *servlet* effectue une redirection **externe** vers `index.jsp`. Sinon, le *servlet* récupère une instance de la classe `URLDB` au moyen de la méthode statique `URLDB.getFromSession`, et insère l'URL avec la méthode `.insertURL`. L'objet de type `URLInfo` renvoyé est stocké dans un *attribut de requête* `"info"` et la vue `/WEB-INF/jsp/add_view.jsp` est chargée au moyen d'une redirection **interne**.
5. (2 points) Compléter le fichier `WEB-INF/jsp/add_view.jsp` pour afficher, comme dans la figure 1.b le *hash*, le *mot de passe* ainsi qu'un lien de la forme :

   ```
   <a href="http://localhost:8080/PAWA_TP4_ACOMPLETER/GoServlet?h=123456789">
     http://localhost:8080/PAWA_TP4_ACOMPLETER/GoServlet?h=123456789
   </a>
   ```

   où `123456789` est le *hash* calculé.

1.2 Redirections (3 points)

On s'intéresse ici aux classes `URLDB` et `GoServlet`, il n'y pas de fichier `.jsp`.

Questions

1. (1.5 point) Compléter la méthode `URLDB.getURLbyHash`. Cette dernière exécute une requête sur la base pour trouver l'URL dont le *hash* est donné, avec une requête de la forme :

   ```
   SELECT * FROM T WHERE hash=123456789
   ```

   Si une entrée existe dans la table, alors la méthode :
   — exécute une mise à jour pour incrémenter `counter` :
   ```
   UPDATE T SET count = count + 1 WHERE hash=123456789
   ```
   — renvoie un objet de type `URLInfo` contenant les informations sur l'URL (le `count` est celui avant incrément). Vous utiliserez votre nom de table à la place de « `T` » dans les requêtes.
   sinon, la méthode renvoie `null`.
2. (1.5 point) Compléter la méthode `doGet` de la classe `GoServlet`. Cette dernière récupère le paramètre `GET` « `h` » et appelle la méthode `URLDB.getURLbyHash` pour trouver l'URL associée à ce *hash*. Elle effectue ensuite une redirection **externe** vers cette URL. Si la valeur de `h` n'est pas un entier valide ou qu'il n'y a aucune entrée dans la base, alors le *servlet* effectue une redirection **externe** vers `index.jsp`

1.3 Suppression (3 points)

On s'intéresse ici à `RemoveServlet` et `remove_view.jsp`. La méthode `URLDB.deleteByHash` est déjà codée. La lire mais **ne pas la modifier**.

Questions

1. (1.5 point) Compléter le code de `RemoveServlet`. Ce dernier récupère les paramètres `GET "hash"` et `"pass"`. La valeur du paramètre `"hash"` est convertie en entier et utilisée avec celle de `"pass"` pour appeler la méthode `deleteByHash`. Si cette méthode renvoie `true`, alors `Boolean.TRUE` valant est stocké dans l'attribut de requête `"removed"`.

   Dans tous les autres cas (la méthode renvoie `false` ou une erreur de conversion se produit), la valeur `new Boolean.FALSE` est stocké dans l'attribut de requête `"removed"`.

   Le *servlet* effectue ensuite une redirection **interne** vers `/WEB-INF/jsp/remove_view.jsp`.
2. (1.5 point) Compléter le code de `WEB-INF/jsp/remove_view.jsp` pour afficher « `URL retirée avec succès` » si l'attribut de requête `"removed"` vaut vrai et pour afficher « `HASH ou PASS incorrect` » sinon.

1.4 Admin (4 points)

On s'intéresse ici à `AdminServlet`, `URLDB` et `admin_view.jsp`.

Questions

1. (1.5 points) Compléter la méthode `URLDB.getAll()` pour qu'elle renvoie un `ArrayList<URLInfo>` pour toutes les URLs présentes dans la base, triée par ordre d'url (le tri doit être fait côté base).
2. (1 point) Compléter le code de `AdminServlet`. Ce dernier récupère le paramètre `GET "password"`. Si ce dernier vaut « `"admin"` », alors le *servlet* récupère au moyen de `URLDB.getAll()` le vecteur de toutes les URLs, le place dans un attribut de requête `"infos"` et effectue une redirection **interne** vers `/WEB-INF/jsp/admin_view.jsp`. Si le mot de passe n'est pas correct, le servlet effectue une redirection **externe** vers `index.jsp`
3. (1.5 point) Compléter `WEB-INF/jsp/admin_view.jsp` pour afficher dans une liste non énumérée (`ul/li`) la liste de toutes les URLs. Pour chaque entrée de liste, vous afficherez dans l'ordre, l'URL, son hash, sa valeur de compteur, son mot de passe, séparés par des virgules.

2 Questions de cours (3 points)

Répondre aux questions se trouvant dans le fichier `reponses.txt`, à la racine du projet Eclipse.