| AP-5 | | TD n°2 |
|---|---|---|
| J2EE | | 5 novembre 2013 |

# Objectifs

*Gestion des informations temporaires, sessions et cookies*
*JSP et servlets, mise en place d'un contrôleur*

# Prérequis

- **Java Runtime Environnement** (http://www.java.com/fr/download/) (JRE Java 7)
- IDE Eclipse utilisé : ***Eclipse kepler*** *for java EE developers* (http://www.eclipse.org/downloads/)
- serveur Web (conteneur de Servlet) : **Apache Tomcat 7**

# Liens

JSP 2.0 référence : http://java.sun.com/products/jsp/syntax/2.0/syntaxref20.html
Java EE API : http://docs.oracle.com/javaee/7/api/

# Session

- **Utilisation de l'objet session (HttpSession)**
  **API** http://java.sun.com/javaee/5/docs/api/javax/servlet/http/HttpSession.html

**Pour les pages JSP qui utilisent l'objet session (true par défaut)**

```
<%@ page session="true" %>
```

**Création d'une variable de session :**

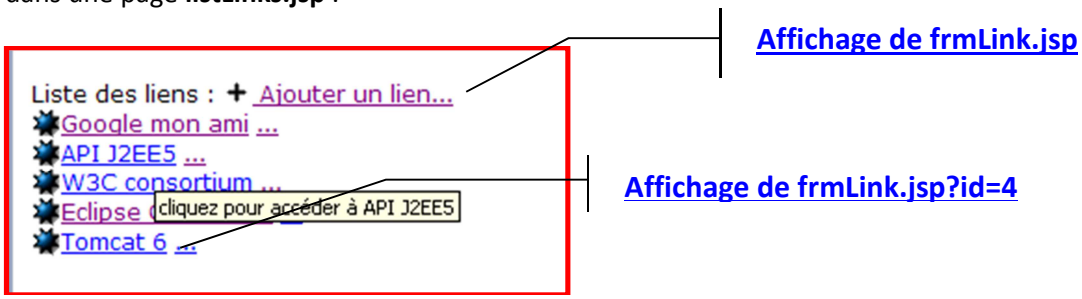
session.setAttribute("nom",value);
et Value peut être un objet :-)

**Lecture d'une variable de session :**

String aValue= session.getAttribute("nom");
Il peut être nécessaire de trans-typer le retour
KMaClasse aValue= (KMaClasse)session.getAttribute("instance");

## Ex4 : sessions

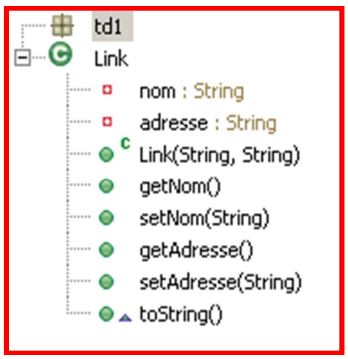
Il s'agit de stocker une collection de liens internet dans une variable de session pour l'afficher dans une page **listLinks.jsp** :




Chaque lien sera une instance de la classe **Link**, classe à créer.
Il faudra stocker dans une variable de session une ArrayList de Link.

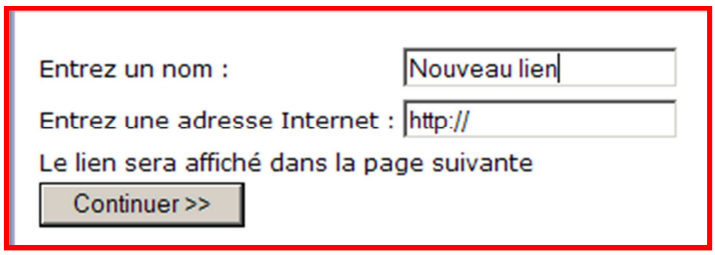
La liste des liens permet d'accéder à la page d'ajout d'un nouveau lien (**frmLink.jsp**).
Chaque lien est suivi d'un autre lien(les points de suspension) permettant d'accéder à la page de modification du lien, l'index du lien dans la variable de session (ArrayList) est passé dans l'URL par la variable id.
La classe **Link** est à créer dans le package **net.td.metier** :

La navigation, La saisie ou la modification des liens sera contrôlée par la seule servlet **Slink**, qui devra vérifier que le lien saisi et validé est conforme à un lien (commence par http:// ou https:// par exemple).
Le lien ne sera ajouté ou modifié que s'il est conforme. Dans le cas contraire, un message sera affiché.

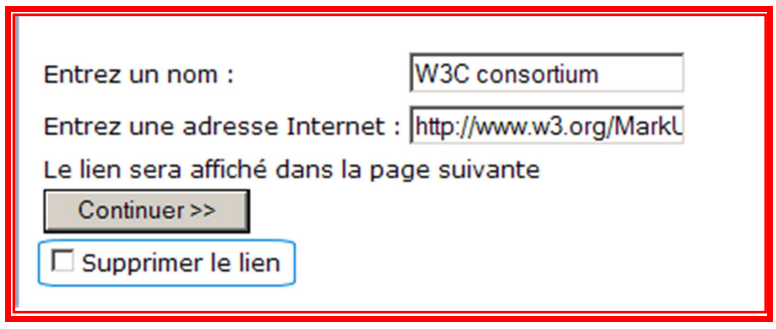



## Vues

Création d'un nouveau lien : **frmLink.jsp**




Modification d'un lien existant : **frmLink.jsp?id=2**




Dans l'ordre, il vous faut :

- Créer la classe métier **Link** ;
- Créer les classes techniques **Gateway** (lien métier/session) et **Gui** (gestion de l'affichage)
- Créer les vues
  - **frmLink.jsp** : ajout/modification d'un lien ;
  - **listLinks.jsp** : affichage de la liste des liens
  - **error.jsp** : page erreur sur ajout de lien
- Créer la servlet de contrôle **SLink**, et associer lui les url en ***.do**

# Cookies

- **Utilisation des cookies (Cookie)**

**API** http://docs.oracle.com/javaee/7/api/javax/servlet/http/Cookie.html

- **Ex5 :**

Les cookies permettent de stocker des informations sur le poste client (si celui-ci les accepte), sous forme de texte. Les informations sont limitées et ne doivent en aucun cas être nécessaires au bon fonctionnement de l'application.
Chaque Cookie permet de stocker une chaîne, associée à une clef (comme les maps).

**Exemple de création :**

```
Cookie user = new Cookie("userName", "Bob");
user.setMaxAge(3600); // en secondes
user.setComment("Nom d'utilisateur");
response.addCookie(user);
```

**Récupération du tableau de cookies :**

Cookie[] cookies = request.getCookies();
utiliser ensuite cookies[i].getName() ou cookies[i].getValue() pour lire le nom ou la valeur du cookie.

**Suppression de cookies :**

Pour supprimer un cookie, il suffit d'utiliser la méthode setMaxAge(0) sur un objet Cookie, puis d'envoyer à nouveau le cookie vers le serveur dans la réponse (response.addCookie(cookieToDelete)).

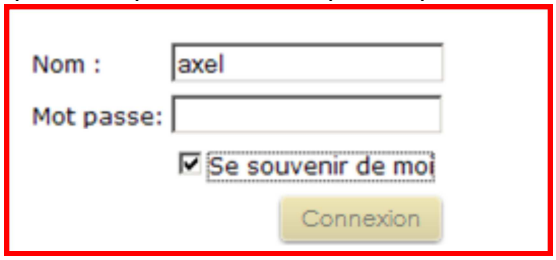
Créer une classe technique **GCookie** munie des méthodes statiques suivantes :

- void add (HttpServletResponse response,String name,String value, int duree)
- String getValue(HttpServletRequest request,String name)
- boolean exists(HttpServletRequest request,String name)
- void delete(HttpServletRequest request ,HttpServletResponse response,String name)

**Tests de la classe GCookie :**

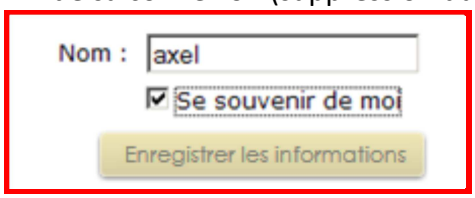
Dans le projet de l'exercice précédent (liens), nous allons faire en sorte que l'application se souvienne de l'utilisateur (sur toutes les pages du site, mais aussi d'une connexion à l'autre). La page listLinks.jsp sera considérée comme page principale.

Créer la vue : formulaire de connexion (**connexion.html**), la case se souvenir de moi permet de mémoriser les informations de l'utilisateur (son nom) d'une connexion à l'autre, pour qu'il n'ait plus besoin de passer par le formulaire de connexion.



Modifier la Servlet (**SLink**) pour gérer la connexion :

- Traiter les données envoyées par le formulaire (méthode post), si la case « se souvenir de moi » est cochée, un Cookie de nom **userName** doit être créé, stockant le nom de l'utilisateur.
- La servlet redirige ensuite vers la page **de liste des liens (listLinks.jsp)**, qui affiche le nom de l'utilisateur, ainsi qu'un bouton de déconnexion.
- Si l'utilisateur n'a pas validé de formulaire, et qu'il n'y a pas de Cookie userName, **la servlet** affiche le formulaire de connexion.
- Si un Cookie existe, la page de **liste des liens** est affichée, avec le nom de l'utilisateur connecté.
- La servlet doit également permettre l'affichage d'une vue de modification des informations de l'utilisateur, lui permettant de modifier son nom, et la mémorisation de sa connexion (suppression du cookie). (Lien sur la page principale)



# Eléments de script JSP

| | | |
|---|---|---|
| Directives: | <%@ directive %> | permettent de contrôler la structure de la servlet générée |
| Déclarations: | <%! déclaration %> | permettent d'ajouter des membres à la servlet générée (données ou méthodes) |
| Expressions: | <%= expression %> | Evitent l'utilisation de out.println() |
| Fragments de code/scriptlet: | <% code java %> | le code généré est ajouté à la méthode service de la servlet |
| Commentaires JSP: | <%-- commentaire --%> | invisible côté client |
| Commentaire HTML : | <!-- commentaires --> | visible côté client |

# Objets implicites (automatiquement déclarés) JSP

| Objet | Classe | Description |
|---|---|---|
| request | javax.servlet.ServletRequest | requête du client |
| response | javax.servlet.ServletResponse | réponse de la page JSP. Cet objet utilisé avec les servlets ne l'est généralement pas avec les JSP, puisque le code HTML est directement créé |
| pageContext | javax.servlet.jsp.PageContext | informations sur l'environnement du serveur |
| session | javax.servlet.http.HttpSession | session en cours |
| application | javax.servlet.ServletContext | contexte de servlet |
| out | javax.servlet.jsp.JspWriter | flux de sortie |
| config | javax.servlet.ServletConfig | configuration de la servlet |
| exception | java.lang.Throwable | exception non interceptée |
| page | java.lang.Object | page elle-même (équivalent à this) |

## Directive page

Exemple :

```
<%@ page import="java.util.*" %>
```

| Option | Valeur | Description |
|---|---|---|
| autoFlush | true/false | indique si le flux en sortie de la servlet doit être vidé quand le tampon est plein. Si la valeur est false, une exception est |

| | | |
|---|---|---|
| | | levée dès que le tampon est plein. On ne peut pas mettre false si la valeur de buffer est none |
| contentType | type MIME | contentType="mimeType [ ; charset=characterSet ]" \| "text/html;charset=ISO-8859-1"<br>Cette option permet de préciser le type MIME des données générées.<br>équivalente à <% response.setContentType("mimeType"); %> |
| errorPage | URL | errorPage="relativeURL"<br>permet de préciser la JSP appelée au cas où une exception est levée<br>Si l'URL commence pas un '/', alors l'URL est relative au répertoire principale du serveur web sinon elle est relative au répertoire qui contient la JSP |
| extends | Classe | extends="package.class"<br>permet de préciser la classe qui sera la super classe de l'objet Java créé à partir de la JSP. |
| import | classe ou package | import= "{ package.class \| package.* }, ..."<br>permet d'importer des classes contenues dans des packages utilisées dans le code de la JSP. Elle s'utilise comme l'instruction import dans un source Java.<br>Chaque classe ou package est séparé par une virgule.<br><br>peut être présente dans plusieurs directives page. |
| isErrorPage | true | permet de préciser si la JSP génère une page d'erreur. La valeur true permet d'utiliser l'objet Exception dans la JSP |
| session | true/false | permet de préciser si la JSP est incluse dans une session ou non. La valeur par défaut (true) permet l'utilisation d'un objet session de type HttpSession qui permet de gérer des informations dans une session |