AP-5		TD n°1
Java EE		23 octobre 2013

Objectifs

Aborder les notions de base JSP
Manipuler les objets implicites et les tags

Prérequis

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

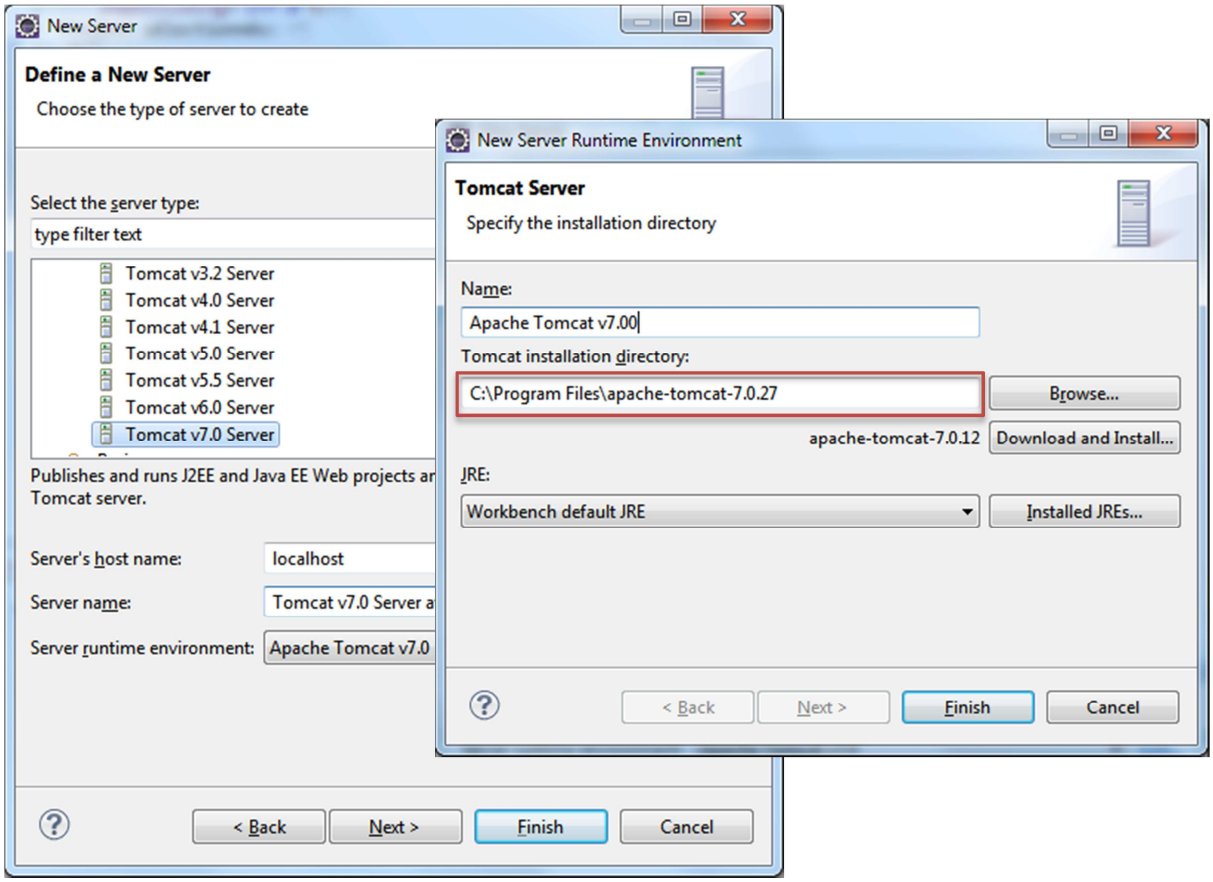
Liens

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/

Intro

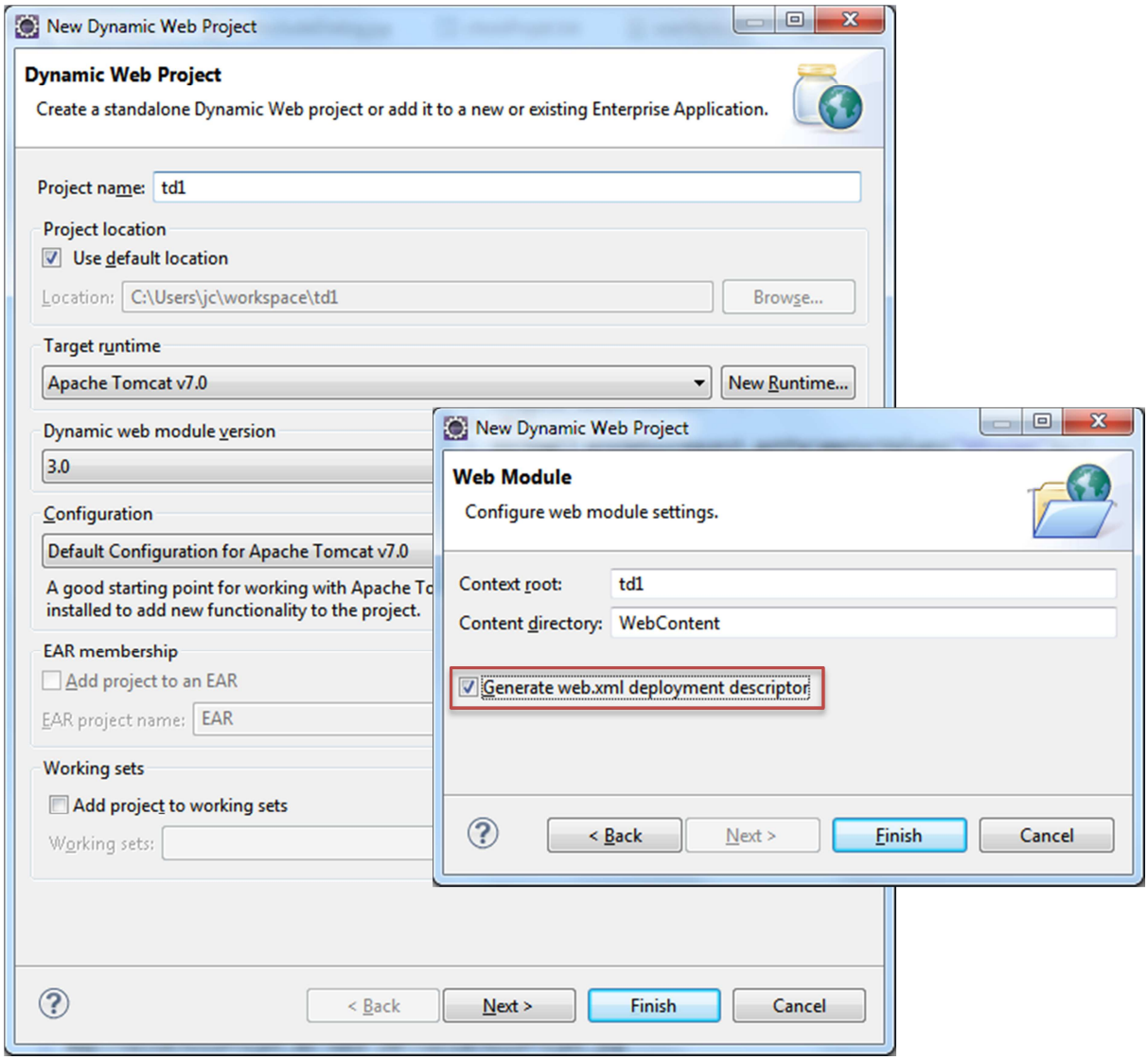
Lancer **Eclipse** :

- nouveau server



  - Créer un nouveau Server de type **tomcat v7 server**
    n'oubliez pas de spécifier le répertoire d'installation de tomcat.
- Nouveau projet

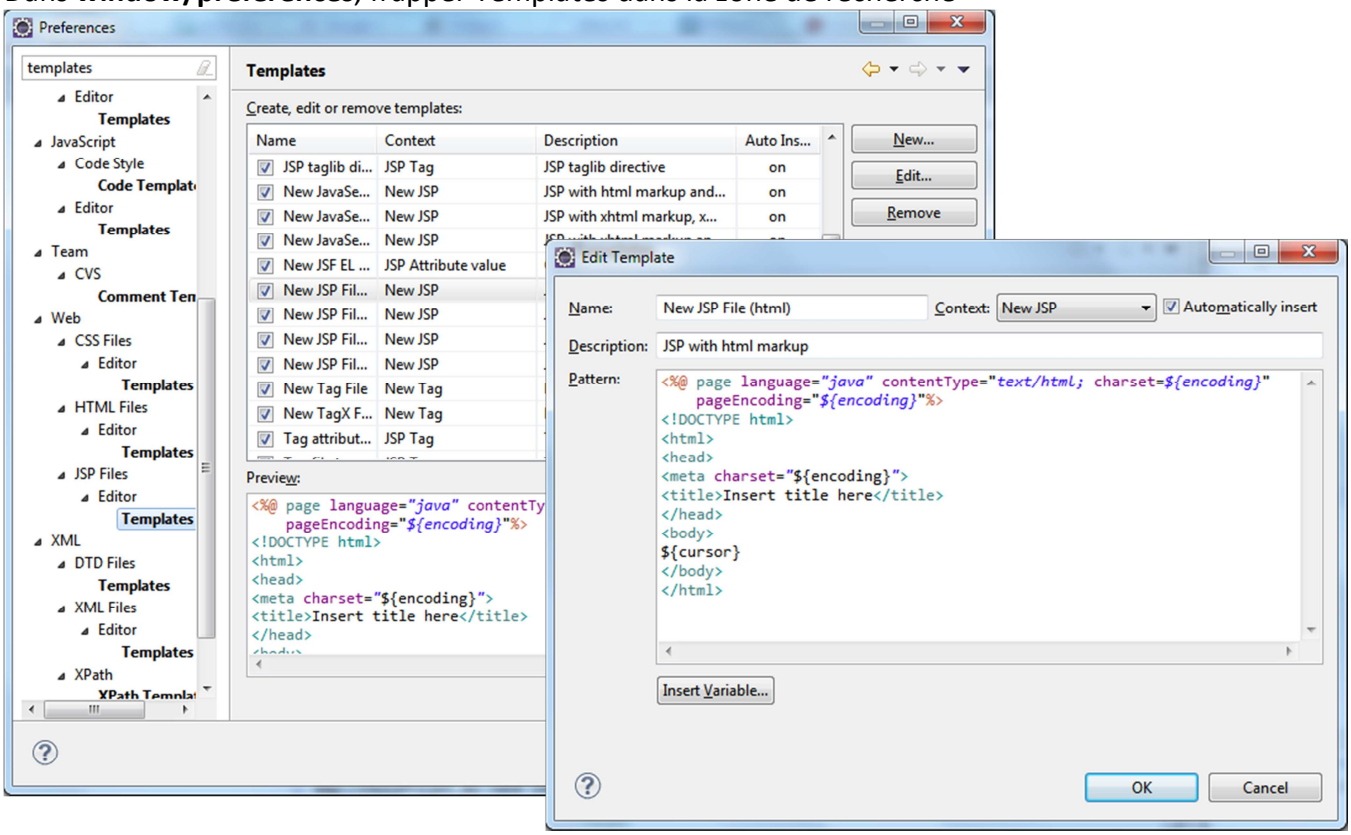
- Créer un nouveau projet de type **Dynamic Web Project**, nommez le **td1**



Configuration d'Eclipse

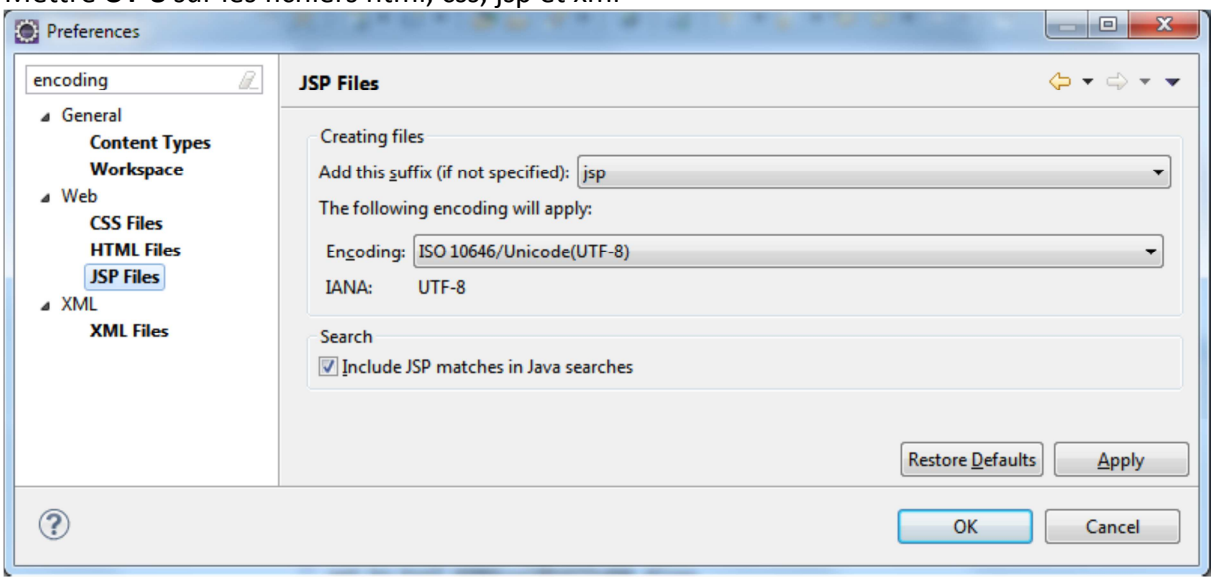
1 Template JSP

Modification du template de création des JSP pour qu'il soit conforme HTML 5 :
Dans **window/preferences**, frapper Templates dans la zone de recherche



2 Encodage par défaut

Toujours dans **Window/preferences**, frapper **Encoding**
Mettre **UT-8** sur les fichiers html, css, jsp et xml



AP-5		TD n°1
Java EE		23 octobre 2013

Objet request, méthodes post et get

- Utilisation de l'objet request (HttpServletRequest)

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

ex 1 : 1ère JSP

Les jsp sont destinées à l'affichage (vues)

Créer la jsp **ex1.jsp** dans le dossier webContent (root de l'application web)

- Le code entouré des balises <% et %> sera exécuté côté serveur, et permet l'accès aux objets implicites JSP, dans le cas présent le PrintWriter **out** , permettant de générer le flux de sortie, et **request**, correspondant à la requête.
- La balise spécifique <%= %> équivaut à <%out.print(…) %>

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Ex1</title>
</head>
<body>
Vous êtes sur la page <b><%=request.getRequestURI() %></b><br>

<fieldset>
      <%
      out.print("le code suivant est exécuté côté serveur");
      %>
</fieldset>
</body>
</html>
```

- A sa première exécution, la JSP est traduite en Servet (classe java héritant de HttpServlet) puis compilée.

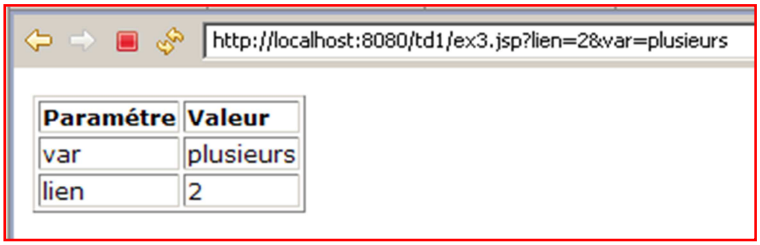
Ex2 : get

Le moyen le plus simple (et le moins sécurisé) de passer des informations d'une page à l'autre est de les passer dans l'URL, par la méthode get.
Créer une page HTML comportant plusieurs liens du style :

```
<a href="ex2.jsp?lien=1">Lien1 avec une variable</a>
<a href="ex2.jsp?lien=2&var=plusieurs">Lien2 avec plusieurs variables</a>
<a href="ex2.jsp">Lien3 sans variable</a>
```

Créer maintenant la jsp ex2.jsp correspondant aux liens et affichant les paramètres de la requête dans un tableau HTML

Exemple :
ex2.jsp?lien=2&var=plusieurs
devra produire le résultat :

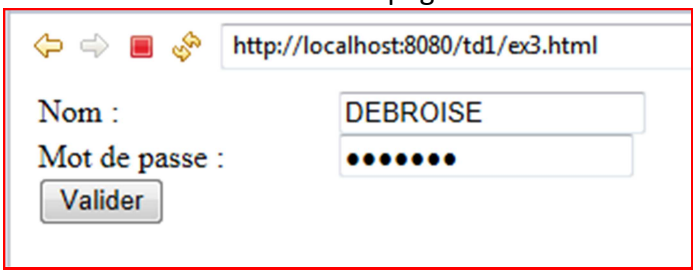


vous pourrez utiliser l'objet **request** et les méthodes **getParameterNames** et **getParameter(String name)**

Ex3 : post et servlet

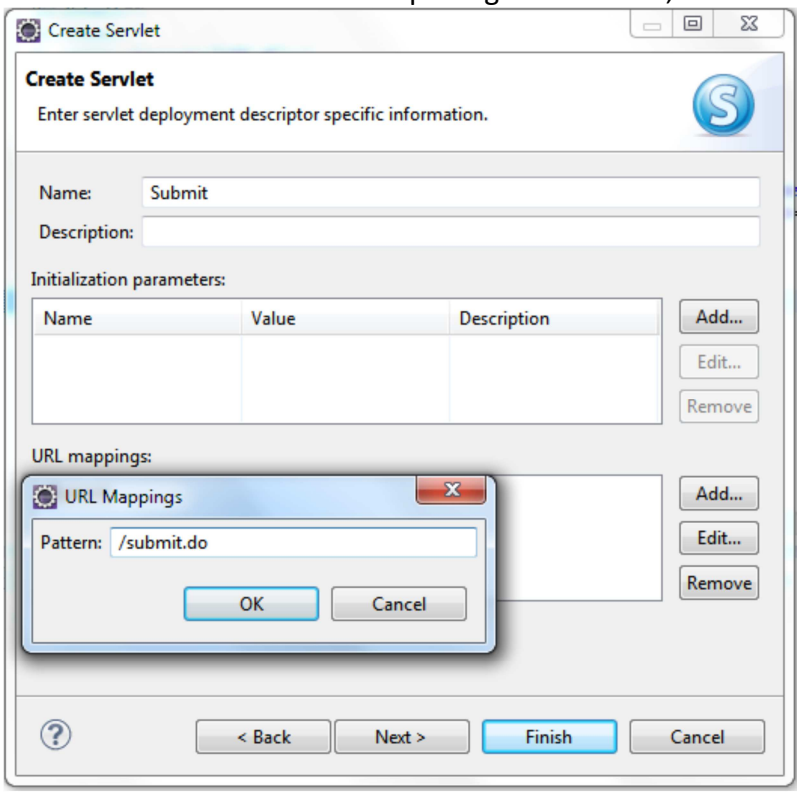
Les servlets sont destinées au contrôle (Contrôleur)

Passage d'informations saisies dans un formulaire, par la méthode POST
Créer le formulaire suivant dans une page ex3.html




```
<form method="post" name="frm" action="submit.do">
<label for="name">Nom :</label><input type="text" id="name" name="name"><br>
<label for="password">Mot de passe :</label><input type="password" id="password"
name="password"><br>
<input type="submit" value="Valider"><br>
</form>
```

Créer la servlet **Submit** dans le package **td1.servlet**, et associée au mapping **submit.do** :



Vérifier que la servlet créée comporte bien l'annotation :

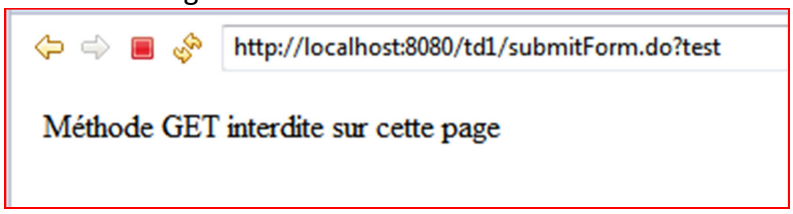
```
@WebServlet(name="Submit", urlPatterns = { "/submit.do" })
```

Une requête vers la page submitForm.do appellera la servlet.

Surdéfinir la méthode get de la servlet pour qu'elle affiche l'information sur l'interdiction d'utiliser le passage de paramètres dans l'url :

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
	PrintWriter out=response.getWriter();
	out.print("Méthode GET interdite sur cette page");
}
```

Résultat d'un get :

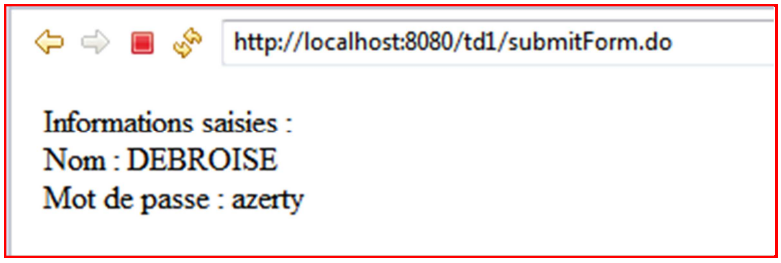


Surdéfinir la méthode post de la servlet pour qu'elle affiche les résultats de la validation du formulaire :

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
	PrintWriter out=response.getWriter();
	if(request.getParameter("name")!=null){
		out.print("Informations saisies :<br>");
		out.print("<div>Nom : "+request.getParameter("name")+"</div>");
		if(request.getParameter("password")!=null)
```

```
                    out.print("<div>Mot de passe :
"+request.getParameter("password")+"</div>");
        }
}
```

Résultat d'un POST :



Session

- **Utilisation de l'objet session (HttpSession)**
  API http://docs.oracle.com/javaee/7/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 :

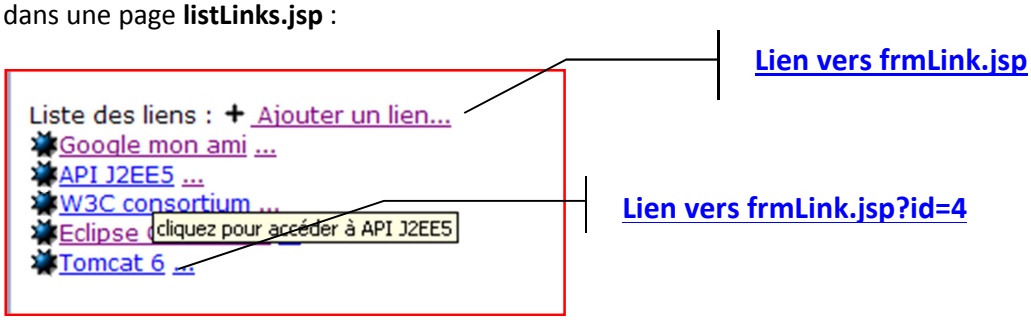
session.setAttribute("nom",value);
et Value peut être un objet :-)

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");

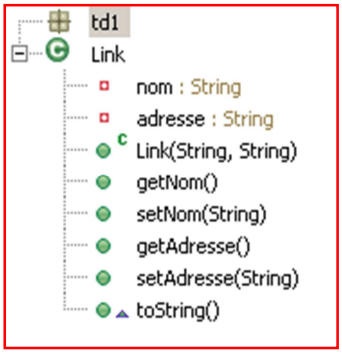
Ex4 : sessions

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

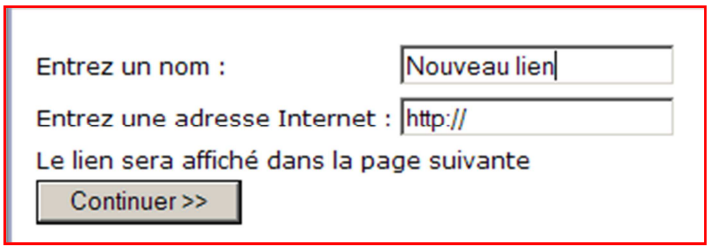


Chaque lien sera une instance de la classe **Link**, classe à créer.
Il faudra stocker dans une variable de session une ArrayList de Link.

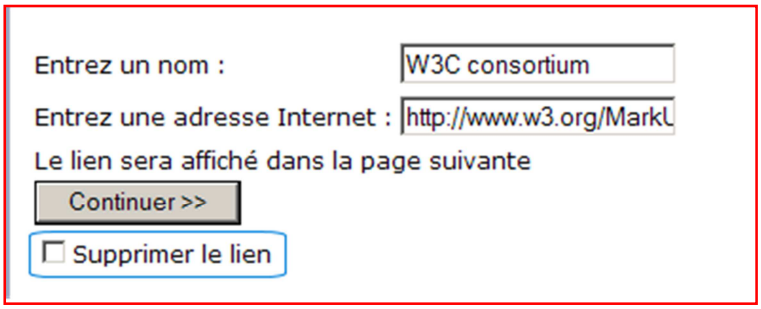
La liste des liens permet d'accéder à la page d'ajout d'un nouveau lien (**frmLink.jsp**).
Chaque lien est suivi d'un autre lien(les points de suspension) permettant d'accéder à la page de modification du lien, l'index du lien dans la variable de session (ArrayList) est passé dans l'URL par la variable id.
La classe **Link** est à créer dans le package **td3.session** :
La saisie ou la modification d'un lien sera contrôlée par la 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, la page de saisie est à nouveau affichée.



Création d'un nouveau lien : frmLink.jsp



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



Dans l'ordre, il faut :

- Créer la classe **Link** ;
- Créer le formulaire **frmLink.jsp**, d'ajout et de modification ;
- Créer la servlet de contrôle **SLink**;
- Créer la page d'affichage des liens **listLinks.jsp**.

Vous aurez peut-être besoin de :

Faire une redirection en java vers une page **redirectURL** :

```
request.getRequestDispatcher(rediectURL).forward(request, response);
```

En ajoutant éventuellement une variable dans la requête avant redirection :

```
request.setAttribute("varName",value);
request.getRequestDispatcher(rediectURL).forward(request, response);
```

Et en la récupérant dans une autre page par :

```
KmaClasse obj=(KmaClasse)request.getAttribute("varName")
```

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" Cette option permet de préciser le type MIME des données générées. équivalente à <% response.setContentType("mimeType"); %>
errorPage	URL	errorPage="relativeURL" permet de préciser la JSP appelée au cas où une exception est levée 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" 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.* }, ..." 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. Chaque classe ou package est séparé par une virgule. 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