# Module de découverte Développement web en Java

Mathieu Chatelain

# Sommaire

- JSTL-EL
  - Présentation
  - 1er exemples
  - EL – Utilisation
  - JSTL –Utilisation
    - Librairie de base – c
    - Librairie de formatage - fmt

# Présentation

- JSTL = Java Standard Tag Library.
  - Simplifier les pages JSP.
  - Utiliser une syntaxe XML.
  - Le développeur de l'interface n'a pas besoin de connaitre Java.
  - Plusieurs versions : 1.0, 1.1 et 1.2. Version 1.1 dans ce document.
- EL = Expressions Languages.
  - Accéder aux beans des différents scope (page, request, session et application) simplement.
  - Disponible depuis JSP 2.0
  - Directement implémenté dans le conteneur JSP (Tomcat).
- But = ne plus utiliser les scriptlets et rendre les pages facilement maintenables.

# 1er exemples

- Affichage de l'user-agent  envoyé par le navigateur
- Version scriptlet

```
<%
   String userAgent = request.getHeader("user-agent");
   out.println(userAgent);
%>
```

- Version JSTL

```
<c:out value="${header['user-agent']}" default="Inconnu"/>
```

- Version EL

```
${header['user-agent']}
```

# EL – Utilisation

- Les objets implicites :

- **pageContext** : Accès à l'objet **PageContext** de la page JSP.
- **pageScope** : Map permettant d'accéder aux différents attributs du scope **'page`.**
- **requestScope** : Map permettant d'accéder aux différents attributs du scope **'request'**.
- **sessionScope** : Map permettant d'accéder aux différents attributs du scope **'session'**.
- **applicationScope** : Map permettant d'accéder aux différents attributs du scope **'application'**.
- **param** : Map permettant d'accéder aux paramètres de la requête HTTP sous forme de **String**.

# EL – Utilisation

- Les objets implicites (suite) :

- **paramValues** : Map permettant d'accéder aux paramètres de la requête HTTP sous forme de **tableau de String**.
- **header** : Map permettant d'accéder aux valeurs du Header HTTP sous forme de **String**.
- **headerValues** : Map permettant d'accéder aux valeurs du Header HTTP sous forme de **tableau de String**.
- **cookie** : Map permettant d'accéder aux différents Cookies.
- **initParam** : Map permettant d'accéder aux **init-param**s du web.xml.

# EL – Utilisation

- Fonctionnement :
- Expression EL

  ${nomVariable}
- Correspond à :

  <%= pageContext.findAttribute("nomVariable"); %>
- Effectue une recherche successive dans les différents scopes de l'application. Ordre : page, request, session et application.
- Risque de conflits, utiliser les objets implicites lorsque vous ne recherchez pas dans le scope page :

  requestScope[" nomVariable"]

# EL – Utilisation

- Propriétés des beans standards :
- Deux méthodes :
  - Opérateur . - point

    ${personne.nom}
  - Opérateur [] – crochet

    ${personne["nom "]}

    Ou

    ${personne['nom ']}

- Equivalent scriptlet :

```
<%@ page import="package.MonBean" %>
<% Personne personne = (Personne) pageContext.findAttribute ("personne");
   if (personne!= null) out.print ( personne.getNnom() ); %>
```

- Avantages:
  - Simplicité, lisibilité,
  - Gestion des exceptions,
  - Pas de cast donc pas besoin d'import.

# EL – Utilisation

- Propriétés des List et tableaux :
- Utilisation de l'opérateur crochet []

  ${list[0]}

  Ou

  ${list["0"]} => Conversion automatique en entier.

- Fonctionnement :

- Utilisation de la méthode get(int) de l'interface List
- Ou
- Accès au éléments du tableau par son index.

# EL – Utilisation

- Propriétés des Map
- Utilisation de l'opérateur crochet []

  ${map ["cle"]}

  Ou

  ${map['cle']}

- Fonctionnement :

- Utilisation de la méthode get(Object) de l'interface Map.

- Les opérateurs arithmétiques = données numériques
- +
- -
- *
- / ou div
- % ou mod
- Les opérateurs relationnels = equals d'Object ou compareTo
- == ou eq
- != ou ne
- < ou lt
- > ou gt
- <= ou le
- >= ge

# EL – Utilisation

# EL – Utilisation

- Opérateurs logiques = booléens
- && ou and
- || ou ot
- ! ou not

- Les autres opérateurs
- Empty
- ()
- ?: condition **?** Valeur true **:** valeur false

# JSTL - Utilisation

- JSTL se compose de plusieurs librairies

| **Librairie** | **URI** | **Préfixe** |
|---|---|---|
| core | http://java.sun.com/jsp/jstl/core | c |
| Format | http://java.sun.com/jsp/jstl/fmt | fmt |
| XML | http://java.sun.com/jsp/jstl/xml | x |
| SQL | http://java.sun.com/jsp/jstl/sql | sql |
| Fonctions | http://java.sun.com/jsp/jstl/functions | fn |

- Pour ceux qui utilisent tomcat 5 dans le fichier web.xml :

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">
…..
</web-app>
```

# JSTL - Utilisation

- Ajouter <%@ page isELIgnored="*false*" %> dans vos pages JSP.

- Pour tout le monde :
- Déclarer les librairies que vous utilisez :

<%@ taglib uri="*http://java.sun.com/jsp/jstl/core*" prefix="*c*" %>

- Ajouter au projet les librairies téléchargées

- Afficher une expression

```
<c:out value="${pageContext.request.requestURL}"
default="Vide"/>

<c:out value="${pageContext.request.requestURL}"> Vide
</c:out>
```

# Librairie de base - c

# Librairie de base - c

- Définir une variable ou une propriété
- Utilisation du tag <c:set/>
- Attributs :
  - value : l'expression
  - var : nom de l'attribut qui contiendra l'expression dans le scope
  - scope : page, request, session ou application
  - target : l'objet dont la propriété  définit par property sera modifiée
  - property : nom de la propriété modifée

- Exemple

  <c:set scope=*"request" var= " var" value= "var en request"* />

- Supprimer une variable de scope
- Utilisation du tag <c:remove/>
- Attributs
  - var : nom de la variable
  - scope : scope de la variable

- Exemple

  <c:remove var= *"var" scope="request"*/>

# Librairie de base - c

# Librairie de base - c

- Catcher des exceptions
- Utilisation du tag <c:catch/>
- Ignorer les exceptions lancées entre les tags « catch » et enregistrer l'exception dans une variable

- Exemple

```
<c:catch var="uneException">
        <c:set target="${requestScope.personne}"
            property="uneProperty" value="1"/>
</c:catch>
```

# Librairie de base - c

- Conditions
- Utilisation du tag <c:if/>
- Attributs
  - test : le test à réaliser
  - var : variable qui contient le résultat du test
  - scope : scope de var

- Exemple

```
<c:if test="${1 eq 1}">
1 = 1
</c:if>
```

- Conditions
- Utilisations des tags <c:choose/>, <c:when/> et <c:otherwise/>
- Attributs
  - test : avec le tag <c:when/>

- Exemple

```
<c:choose>
        <c:when test="${valeur==1}"> valeur = 1  </c:when>
        <c:when test="${valeur==2}"> valeur = 2  </c:when>
        <c:when test="${valeur==3}"> valeur = 3  </c:when>
        <c:otherwise>
                valeur = ${valeur}
        </c:otherwise>
</c:choose>
```

# Librairie de base – c

# Librairie de base - c

- Les itérations
- Utilisation du tag <c:forEach/>
- Attributs
  - var : variable qui représente l'élément courant
  - varStatus : information sur le status de l'itération
  - Begin : index de départ
  - End : index de fin
  - Step : itération sur N éléments
- VarStatus
  - count : compteur
  - current : représente l'élément courant
  - index : index dans la collection
  - first : est le 1$^{er}$ ?
  - last : est le dernier ?

# Librairie de base - c

- Exemples d'itération

```
<c:forEach var= " liste" items="${uneListe}" >
</c:forEach>
```

```
<c:forEach var= " liste" items="${uneListe}"  varSatus= "status"
>
// utilisation des propriétés de status
// exemple : status.last
</c:forEach>
```

- Exemple d'itération sur une Map

```
<c:forEach var= " map" items="${uneMap}" >
${map.key}
${map.value}
</c:forEach>
```

# Librairie de base - c

- Itération sur un Objet String
- Utilisation du tag <c:forTokens/>
- Attributs
  - items : la chaine à parser
  - delims : le délimiteur
  - Et idem forEach
- Exemple
- <c:forTokens var="mot" items="un,deux,trois"  delims="," >
- ${mot}<br />
- </c:forTokens>

# Librairie de base - c

- Gestion des URLs
- Utilisations des tags :
- <c:url/>
  - Attributs
    - value : le lien
    - contexte : par défaut request.getContextPath()
    - var : variable String contenant le lien
    - scope : scope de var
- <c:param/>
  - Attributs
    - name : nom de paramètre
    - value : sa valeur
- Exemple

```
<c:url value="/vitesse.jsp">
        <c:param name="action" value="accelerer"/>
</c:url>
```

# Librairie de base - c

- Redirection
- Utilisation du tag <c:redirect/>
- Possibilité d'utiliser le tag <c:param/> comme pour les URLs
- Le reste de la page ne sera pas interprété

- Exemple
  <c:redirect url="http://lpsil.iut.univ-metz.fr"/>

# Librairie de base - c

- Importer une ressource
- Utilisation du tag <c:import/>
- Ressemble au tag <jsp:include/>
- Possibilité de récupérer des ressources sur des serveurs distants
- Attributs
  - url : l'url de la ressource
  - context : idem <c:url/>
  - var :variable contenant  le contenu de la ressource sous forme de String
  - scope : scope de var
  - charEncoding : encodage
  - varReader : variable contenant le contenu de la ressource sous forme de Reader
- Exemple
- <c:import url="*/helloWorld.jsp*" />

# Librairie de formatage - fmt

- Utilisation
- <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
- Internationalisation d'une application
- Utilisation de fichiers properties pour définir le texte d'une page
- Exemple fichier properties

  msg.titre=Titre de la page
- Un fichier par langue de la forme :

  nom_fichier_fr_FR.properties

  nom_fichier_fr.properties
- La langue sera recherchée dans la propriété « Accept-Language » du header HTTP
- Possibilité de configurer la langue dans le fichier web.xml

# Librairie de formatage - fmt

- Exemple détaillé
- Définir les fichiers de ressources

  src/resources/message_fr.properties

  src/resources/message_en.properties
- Définir la Locale

  <fmt:setLocale value="fr" scope="session"/>
- Définir le ResourceBundle

  <fmt:setBundle basename=*"resources.message"*
  *var="messageBundle" />*
- Afficher le message

  <fmt:message key=*"msg.titre"*
  *bundle="${messageBundle}" />*

# Librairie de formatage - fmt

- Définir la Locale
- Utilisation du tag <fmt:setLocale/>
- Attributs
  - value : valeur de la Locale
  - variant : définir une variant pour un navigateur ou système
  - scope : scope de la Locale

# Librairie de formatage - fmt

- Définir un ResourceBundle
- Utilisation du tag <fmt:setBundle/>
- Attributs
  - basename : nom du ResourceBundle
  - var : variable pour appeler le ResourceBundle
  - scope : scope de var

# Librairie de formatage - fmt

- Afficher des messages
- Utilisation du tag <fmt:message>
- Attributs
  - key : clé
  - bundle : le ResourceBundle
  - var : variable qui contient la valeur de key
  - scope : scope de var
- Les paramètres des messages
- Fichier properties

  msg.accueil=Bonjour {0}
- Utilisation du tag <fmt:param/>
- Attribut
  - value : la valeur du paramètre
- Exemple

```
<fmt:message key="msg.accueil" bundle="${messageBundle}">
    <fmt:param value="Mathieu" />
</fmt:message>
```

# Librairie de formatage - fmt

- Formater un nombre
- Utilisation du tag <fmt:formatNumber/>
- Attributs
  - value : valeur numérique
  - type : number, currency ou percent
  - pattern : pattern comme java.text.DecimalFormat
  - currencyCode : pour currency, code iso
  - currencySymbol : pour currency, code monétaire
  - groupingUsed : utiliser des séparateurs pour les grand nombre (défaut : true)
  - maxIntegerDigits : partie entière, max de digits
  - minIntegerDigits : partie entière; min de digits
  - maxFractionDigits : partie décimale, max de digits
  - minFractionDigits : partie décimale, min de digits
  - var : variable
  - scope : scope de var
- Exemple

  <fmt:formatNumber value="*25*" *type="number"*/>