UNIVERSITE DE SOUSSE
Institut Supérieur d'Informatique
et des Techniques de Communication
Hammam Sousse

جـامعة سوسة
المعهد العالي للإعلامية وتقنيات الاتصال
بحمام سوسة

Cours Développement d'applications Web et Multimédia

Chap4 : Expression Language (EL) et librairie JSTL

Filière : 3ème année Licence Multimédia

Amen Ajroud
amen.ajroud.isitcom@gmail.com

Année Universitaire : 2021-2022

Plan

- Expression Language : EL
  - Définition
  - Objets implicites
  - Utilisation
  - Avantages
- Java Standard Tag Library (JSTL)
- Librairie Core
  - Gestion des variables
  - Structures conditionnelles
  - Structures itératives

Expression Language

- Depuis la version 2.0 des JSP, il est possible de placer à n'importe quel endroit d'une page JSP des expressions qui sont évaluées et le résultat de leur évaluation **sera affiché** : ce sont les Expressions Languages (EL) .
- Les EL permettent **d'afficher** les données au sein d'une page JSP plus simplement, au lieu d'utiliser les expressions Java (`<%= expres %>`) ou les instructions java `out.println()`.
- Une EL permet de référencer les variables et attributs des différents contextes (scope) de l'application web (page, request, session et application).

Expression Language

- La syntaxe d'une EL est de la forme suivante :

```
${expression}
```

- La chaîne `expression` correspond à l'expression à interpréter puis à afficher.
- Une EL peut désigner :
  - `${nom_attribut}` => sa valeur
  - `${ (10 + 2)*2 }` => *24*
  - `${ a >= 10 }` => *true or false*

Expression Language

- Les différents contextes (scope) :
  - **pageScope:** scope permettant d'accéder aux différents attributs du scope 'page‘.
  - **requestScope:** scope permettant d'accéder aux différents attributs du scope 'request'.
  - **sessionScope:** scope permettant d'accéder aux différents attributs du scope 'session'.
  - **applicationScope:** scope permettant d'accéder aux différents attributs du scope 'application'.

Expression Language

- Les scopes implicites (suite):
  - **param:** Map permettant d'accéder aux paramètres de la requête HTTP sous forme de String.
  - **paramValues:** Map permettant d'accéder aux paramètres de la requête HTTP sous forme de tableau de String.

Expression Language

- Exemple 1:
  - À l'expression d'affichage suivante :

```
<%= request.getAttribute("nomAttribut")%>
```

  - Correspond la EL suivante :

```
${requestScope.nomAttribut} ou bien
${nomAttribut}
```

- Exemple 2:
  - à l'expression :

```
<%= request.getParameter("nomVariable")%>
```

  - correspond à EL : `${param.nomVariable}`

Expression Language

- Opérateur sur EL : `empty`
- Exemple :
  - `${empty attribut}`
  - Cette expression est évaluée à :
    - `True`
      - si l'attribut est vide
      - si l'attribut n'existe pas dans la requête.
    - `False` sinon

Expression Language

- Avantages EL :
  - Gestion des Exceptions : Soit la donnée à afficher est valide alors elle s'affiche, soit elle n'est pas présente et on n'affiche rien... Cela permet de se passer d'effectuer plusieurs vérifications au sein des JSP avant d'afficher des données...
  - La notion d'EL a été introduite afin de faciliter la conception de pages JSP, en particulier afin de pouvoir accéder et utiliser des données dynamiques avec le minimum de code écrit en Java (scriptlet)

Java Standard Tag Library (JSTL)

- **JSTL** propose une librairie standard pour la plupart des fonctionnalités de base d'une application Java EE.
- Le but de la JSTL est de simplifier le travail des web designers (développeurs des pages JSP) **en remplaçant** au maximum les scriptlets java par des balises (qui est une syntaxe proche des langages utilisés par les web designers).
- JSTL se base sur l'utilisation des EL.

Java Standard Tag Library (JSTL)

- **Utilisation:**
  - Les JSTL disposent d'un ensemble de librairies de fonctionnalités différentes.
  - Pour inclure une librairie jstl à une page JSP il faut ajouter une déclaration en tête de page (directive taglib) de cette façon:

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

  - Cette ligne permet d'inclure la librairie "core" de la JSTL qui contient les actions de base d'une application web.
  - Voici une liste des différentes librairies existantes :

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

Librairie Core

- Cette librairie contient les actions de base d'une application web qui inclut :
- Gestion des variables :
  - **<c:out/> : Afficher une expression**
    - `<c:out value="${expression}" />`
    - Ou encore plus simplement : `${expression}`
  - **<c:set/> : Définir une variable d'un scope**
    - `<c:set var="maVariable" value="valeur" scope="..." />`
    - Ex : Stocker *valeur dans une variable maVariable ayant une portée sur la page.*
    - `<c:set var="compteur" value="0" scope="page" />`
  - **<c:remove/> : Supprimer une variable d'un scope.**
    - `<c:remove var="maVariable" scope="..." />`
    - Ex : Supprimer l'attribut *varName de la session:*
    - `<c:remove var="compteur" scope="page"/>`

Librairie Core

- **Structures conditionnelles :**
  - **<c:if/> : Traitement conditionnel (le même que celui du langage Java).**

Ex : Tester sur la valeur d'un compteur pour afficher un message

```
<c:if test="${compteur == 0}"> Aucun résultat</c:if>
```

  - **<c:choose /> : Traiter différents cas mutuellement exclusifs (Switch en Java).**

Ex : Afficher un message d'accueil différent en fonction du paramètre de la requête *civilite* .

```
<c:choose>
<c:when test="${civilite=='Mr'}">Bonjour Monsieur</c:when>
<c:when test="${civilite=='Mme'}">Bonjour Madame</c:when>
<c:when test="${civilite=='Mlle'}">Bonjour Melle</c:when>
<c:otherwise>Bonjour</c:otherwise>
</c:choose>
```

Librairie Core

- Structures itératives :
- **<c:forEach/> :** Permet d'effectuer simplement des itérations sur plusieurs types de collection de données (idem que `for` et `while` en Java).
  - Ex : Afficher « 1,2,3,4,5,6,7,8,9,10,»

```
<c:forEach var="i" begin="1" end="10" step ="1" >
     ${i} ,
</c:forEach>
```

  - Afficher le contenu d'un tableau de chaines de caractères nommé *customers*

```
<c:forEach var="customer" items="${customers}">
    ${customer}<br>
</c:forEach>
```

Librairie Core

- **<c:forTokens /> :** Permet de découper des chaînes de caractères selon un délimiteur.
  - Ex : Décomposer *str et afficher chaque sous-chaîne sur une ligne différente.*

```
<c:forTokens var="mot" delims=";" items="${str}">
   ${mot} <br />
</c:forTokens>
```