



# Développement des systèmes d'information



## Cours Architectures logicielles

05- JSTL

**Mohamed ZAYANI**

2022/2023

# JSTL?

- **JSTL** est l'acronyme de **J**ava **s**erver **p**age **S**tandard **T**ag **L**ibraryOutils
- C'est un **ensemble de tags** (balises) **personnalisés** développé qui propose des fonctionnalités souvent rencontrées dans les JSP :
  - ❖ Tag de structure (itération, conditionnement ...)
  - ❖ Internationalisation
  - ❖ Exécution de requêtes SQL
  - ❖ Utilisation de documents XML

# Objectifs de JSTL

- Le but de **JSTL** est de simplifier le travail des auteurs de page JSP, c'est à dire la personne responsable de la couche présentation d'une application web JEE.
- **JSTL** permet d'éviter l'utilisation de code Java dans les pages JSP
- **JSTL** permet de réduire la quantité de code à écrire
- **JSTL** rend le code des pages JSP plus lisible

# JSTL en bref

- **JSTL** ou **JSP Standard Tag Library** met à disposition du développeur web des balises pour accomplir la plupart des tâches qui doivent être réalisées avec les JSP.
- Avec **JSTL**, on remplace les balises et le code Java par du XML spécifique.
- Ne plus écrire de Java directement dans vos JSP

# Utilisation de JSTL

- Pour pouvoir utiliser une bibliothèque personnalisée, il faut utiliser la directive **taglib**. Exemple déclarer la bibliothèque de base:

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

- Exemple:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <title>Exemple</title>
    </head>
    <body>
        <c:out value="Bonjour" />
        <br/>
    </body>
</html>
```

Signifie que toutes les balises de cette bibliothèque doivent commencer par le préfixe « C » puis « : »

- Il est aussi nécessaire d'ajouter le fichier descripteur de JSTL dans le dossier « lib » de l'application web. Ce fichier est téléchargeable depuis le lien suivant:

<http://tomcat.apache.org/download-taglibs.cgi>

# Fichier descripteur de JSTL

## Documentation

Tomcat 10.0 (alpha)  
Tomcat 9.0  
Tomcat 8.5  
Tomcat 7.0  
Tomcat Connectors  
Tomcat Native  
Wiki  
Migration Guide  
Presentations

## Problems?

Security Reports  
Find help  
FAQ  
Mailing Lists  
Bug Database  
IRC

## Get Involved

Overview  
Source code

### Standard-1.2.5

#### Source Code Distributions

- [Source README](#)
- [zip \(pgp, sha512\)](#)

#### Jar Files

- [Binary README](#)
- Impl:
  - [taglibs-standard-impl-1.2.5.jar \(pgp, sha512\)](#)
- Spec:
  - [taglibs-standard-spec-1.2.5.jar \(pgp, sha512\)](#)
- EL:
  - [taglibs-standard-jstlel-1.2.5.jar \(pgp, sha512\)](#)
- Compat:
  - [taglibs-standard-compat-1.2.5.jar \(pgp, sha512\)](#)



# Langage d'expression (EL)

- **JSTL** propose un langage particulier (**EL**) constitué d'expressions qui permettent d'utiliser et de faire référence à des objets Java accessibles dans les différents contextes de la page JSP.
- Le but est de **fournir un moyen simple d'accéder aux données nécessaires à une JSP**.
- La syntaxe de base est **`${xxx}`** où xxx est le nom d'une variable d'un objet Java défini dans un contexte particulier.
- La définition dans un contexte permet de définir la portée de la variable (**page**, **requête**, **session** ou **application**).
- **EL** permet facilement de s'affranchir de la syntaxe de Java pour obtenir une variable.

## • Exemple:

- Accéder à l'attribut nom d'un objet personne situé dans la session avec **Java**:

**`<%= session.getAttribute("personne").getNom() %>`**

- Accéder à l'attribut nom d'un objet personne situé dans la session avec **EL**

**`${sessionScope.personne.nom}`**

# La balise **set**

- Le tag **set** ( ou balise **set**) permet de stocker une variable dans une portée particulière (page, requête, session ou application).
- La balise **set** possède plusieurs attributs:

Attribut	Rôle
<b>value</b>	valeur à stocker
<b>target</b>	nom de la variable contenant un bean dont la propriété doit être modifiée
<b>property</b>	nom de la propriété à modifier
<b>var</b>	nom de la variable qui va stocker la valeur
<b>scope</b>	portée de la variable qui va stocker la valeur

- Exemple:
- `<c: set var="message" value="Hello" scope="page" />`
- `<c: set target="${bean}" property="login" value="Ali" />`



# La balise **out**

- Le tag **out**( ou balise **out**) permet d'envoyer dans le flux de sortie de la JSP le résultat de l'évaluation de l'expression fournie dans le paramètre " **value** ".
- Ce tag est équivalent au tag d'expression `<%= ... %>` de JSP.
- La balise **out** possède plusieurs attributs:

Attribut	Rôle
<b>value</b>	valeur à afficher (obligatoire)
<b>default</b>	définir une valeur par défaut si la valeur est null
<b>escapeXml</b>	booléen qui précise si les caractères particuliers (< > & ...) doivent être convertis en leurs équivalents HTML (&lt; &gt; &amp ; ...)

- Exemple:
- `<c:out value="$${personne.nom}" default="Inconnu" />`

Si la valeur est null et que l'attribut default n'est pas utilisé alors c'est une chaîne vide qui est envoyée dans le flux de sortie.

# La balise **if**

- Le tag **if** ( ou balise **if**) permet d'évaluer le contenu de son corps si la condition qui lui est fournie est vraie
- La balise **if** possède plusieurs attributs:
- Cette balise nécessite l'ajout du fichier «**jst-1.2.jar**» dans le dossier « **lib** »

Attribut	Rôle
<b>test</b>	condition à évaluer
<b>var</b>	nom de la variable qui contiendra le résultat de l'évaluation
<b>scope</b>	portée de la variable qui contiendra le résultat

- Exemple:
- `<c:if test="$ {empty personne.nom}" var="resultat" />`

L'opérateur **empty** si un objet est null ou vide si c'est une chaîne de caractère. Renvoie un booléen

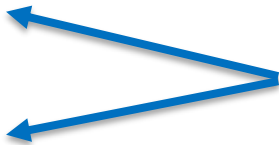
L'évaluation de la condition est stockée dans une variable **10**

# La balise **choose**

- Le tag **choose** ( ou balise **choose**) permet de traiter différents cas mutuellement exclusifs dans un même tag. Le tag choose ne possède pas d'attribut. Il doit cependant posséder un ou plusieurs tags fils « **when** ».

- Exemple:

```
<c:choose>
  <c:when test="{personne.etat_civil == 'Mr'}">
    Bonjour Monsieur
  </c:when>
  <c:when test="{personne.etat_civil == 'Mme'}">
    Bonjour Madame
  </c:when>
  <c:when test="{personne.etat_civil == 'Mlle'}">
    Bonjour Mademoiselle
  </c:when>
  <c:otherwise>
    Bonjour
  </c:otherwise>
</c:choose>
```



Si aucun des cas précédents n'est vrai , afficher le texte cité au corps de la balise **otherwise**

# La balise **forEach**

- Le tag **forEach** ( ou balise **forEach**) permet de parcourir les différents éléments d'une collection et ainsi d'exécuter de façon répétitive le contenu de son corps.».
- La balise **forEach** possède plusieurs attributs:

Attribut	Rôle
<b>var</b>	nom de la variable qui contient l'élément en cours de traitement
<b>items</b>	collection à traiter
<b>property</b>	nom d'une variable qui va contenir des informations sur l'itération en cours de traitement
<b>begin</b>	numéro du premier élément à traiter (le premier possède le numéro 0)
<b>end</b>	numéro du dernier élément à traiter
<b>step</b>	pas des éléments à traiter (par défaut 1)

# La balise **forEach**

- A chaque itération, la valeur de la variable dont le nom est précisé par la propriété **var** change pour contenir l'élément de la collection en cours de traitement.

- Exemple:

```
<c:forEach items="${ produits}" var="produit" varStatus="status">
  <p>Produit N°<c:out value="${ status.count }" /> :
  <c:out value="${ produit.designation }" /> !</p>
</c:forEach>
```

- L'attribut **varStatus** permet de définir une variable qui va contenir des informations sur l'itération en cours d'exécution. Cette variable possède plusieurs propriétés :
  - **index**: indique le numéro de l'occurrence dans l'ensemble de la collection
  - **count**: indique le numéro de l'itération en cours (en commençant par 1)
  - **first**: booléen qui indique si c'est la première itération
  - **last**: booléen qui indique si c'est la dernière itération

- Exemple:

```
<c:forEach begin="1" end="12" var="i" step="3" varStatus="vs">
  index = <c:out value="${vs.index}"/> :
  count = <c:out value="${vs.count}"/> :
  value = <c:out value="${i}"/>
  <c:if test="${vs.first}">
    : Premier element
  </c:if>
  <c:if test="${vs.last}">
    : Dernier element
  </c:if> <br>
</c:forEach>
```

- Résultat:

```
index = 1 : count = 1 : value = 1 : Premier element
index = 4 : count = 2 : value = 4
index = 7 : count = 3 : value = 7
index = 10 : count = 4 : value = 10 : Dernier element
```

# La balise **import**

- Le tag **import** ( ou balise **import**) d'accéder à une ressource grâce à son URL pour l'inclure ou l'utiliser dans les traitements de la JSP. La ressource accédée peut être dans une autre application.
- La balise **import** possède plusieurs attributs:

Attribut	Rôle
<b>url</b>	URL de la ressource (obligatoire)
<b>var</b>	nom de la variable qui va stocker le contenu de la ressource sous la forme d'une chaîne de caractères
<b>scope</b>	portée de la variable qui va stocker le contenu de la ressource
<b>context</b>	contexte de l'application web qui contient la ressource (si la ressource n'est pas l'application web courante)
<b>charEncoding</b>	jeu de caractères utilisé par la ressource

- Exemple:

```
<c:import url="/message.txt" var="message" />  
<c:out value="${message}" /><br/>
```

# La balise **redirect**

- Le tag **redirect** ( ou balise **redirect**) permet faire une redirection vers une nouvelle URL.
- Les paramètres peuvent être fournis grâce à un ou plusieurs tags fils **param**.
- Exemple:

```
<c:redirect url="liste.jsp">  
  <c:param name="id" value="123"/>  
</c:redirect>
```

# La balise **url**

- Le tag **url**( ou balise **url**) permet de formater une URL.
- Cette balise possède plusieurs attributs :
  - **value**: base de l'URL (obligatoire)
  - **var**: nom de la variable qui va stocker l'URL
  - **scope**: portée de la variable qui va stocker l'URL
- Le tag **url** peut avoir un ou plusieurs tags fils « **param** ».
- Le tag **param** permet de préciser un paramètre et sa valeur pour qu'ils soient ajoutés à l'URL générée.
- Cette balise possède plusieurs attributs :
  - **name** : nom du paramètre
  - **value**: valeur du paramètre
- Exemple:

```
<c:url value="/list.jsp">  
    <c:param name="paramName" value="paramValue"/>  
</c:url>
```



# La bibliothèque de formatage

- Cette librairie simplifie la localisation et le formatage des données.
- Cette bibliothèque est déclarée ainsi:

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

Signifie que toutes les balises de cette bibliothèque doivent  
Commencer par le préfixe « **fmt** » puis « : »

- Exemple:

```
<!-- Afficher une date selon un format spécifique : -->  
<fmt:formatDate value="${dateBeans}" pattern="dd/MM/yyyy"/>
```

# La balise **formatDate**

- Le tag **formatDate** ( ou balise **formatDate**) permet de formater une date afin de l'afficher à l'utilisateur.
- La balise **formatDate** possède plusieurs attributs:

Attribut	Rôle
<b>value</b>	La date à formater.
<b>type</b>	" <b>date</b> ", " <b>time</b> " ou " <b>both</b> " indiquant respectivement que l'on traite une <b>date</b> , une <b>heure</b> ou les <b>deux</b> (défaut : "date").
<b>dateStyle</b>	" <b>default</b> ", " <b>short</b> ", " <b>medium</b> ", " <b>long</b> " ou " <b>full</b> ", qui correspondent à un style de formatage de la date, et donc à un pattern d'analyse de la date. Ces différents styles varient selon la locale.
<b>timeStyle</b>	" <b>default</b> ", " <b>short</b> ", " <b>medium</b> ", " <b>long</b> " ou " <b>full</b> ", qui correspondent à un style de formatage de l'heure, et donc à un pattern d'analyse de la date. Ces différents styles varient selon la locale.
<b>pattern</b>	Le pattern a utilisé pour analyser la date/heure. Cet attribut est prioritaire sur <b>dateStyle</b> et <b>timeStyle</b> . Sa syntaxe est la même que celle utilisé avec la classe <code>java.text.SimpleDateFormat</code> .

# Exemple de formatage d'une date

- Exemple:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/fmt" prefix = "fmt" %>
<html>
  <head>
    <title>JSTL fmt:dateNumber Tag</title>
  </head>
  <body>
    <h3>Number Format:</h3>
    <c:set var = "now" value = "<%= new java.util.Date()%>" />

    <p>Formatted Date (1): <fmt:formatDate type = "time"
      value = "${now}" /></p>

    <p>Formatted Date (2): <fmt:formatDate type = "date"
      value = "${now}" /></p>

    <p>Formatted Date (3): <fmt:formatDate type = "both"
      value = "${now}" /></p>

    <p>Formatted Date (4): <fmt:formatDate type = "both"
      dateStyle = "short" timeStyle = "short" value = "${now}" /></p>

    <p>Formatted Date (5): <fmt:formatDate type = "both"
      dateStyle = "medium" timeStyle = "medium" value = "${now}" /></p>

    <p>Formatted Date (6): <fmt:formatDate type = "both"
      dateStyle = "long" timeStyle = "long" value = "${now}" /></p>

    <p>Formatted Date (7): <fmt:formatDate pattern = "yyyy-MM-dd"
      value = "${now}" /></p>
  </body> </html>
```

## Number Format:

Formatted Date (1): 15:33:40

Formatted Date (2): 20 mars 2020

Formatted Date (3): 20 mars 2020 15:33:40

Formatted Date (4): 20/03/20 15:33

Formatted Date (5): 20 mars 2020 15:33:40

Formatted Date (6): 20 mars 2020 15:33:40 GMT+01:00

Formatted Date (7): 2020-03-20

# La balise **parseDate**

- Le tag **parseDate** ( ou balise **parseDate**) permet de créer des dates en analysant une chaîne de caractères.
- La balise **parseDate** possède plusieurs attributs:

Attribut	Rôle
<b>value</b>	La date à formater.
<b>type</b>	" <b>date</b> ", " <b>time</b> " ou " <b>both</b> " indiquant respectivement que l'on traite une <b>date</b> , une <b>heure</b> ou les <b>deux</b> (défaut : "date").
<b>dateStyle</b>	" <b>default</b> ", " <b>short</b> ", " <b>medium</b> ", " <b>long</b> " ou " <b>full</b> ", qui correspondent à un style de formatage de la date, et donc à un pattern d'analyse de la date. Ces différents styles varient selon la locale.
<b>timeStyle</b>	" <b>default</b> ", " <b>short</b> ", " <b>medium</b> ", " <b>long</b> " ou " <b>full</b> ", qui correspondent à un style de formatage de l'heure, et donc à un pattern d'analyse de la date. Ces différents styles varient selon la locale.
<b>pattern</b>	Le pattern a utilisé pour analyser la date/heure. Cet attribut est prioritaire sur <b>dateStyle</b> et <b>timeStyle</b> . Sa syntaxe est la même que celle utilisé avec la classe <code>java.text.SimpleDateFormat</code> .

- Exemple:

```
<!-- Créer une date initialisé au premier janvier 2005 : -->  
<fmt:parseDate value="01/01/2005" pattern="dd/MM/yyyy" var="date"/><br/>
```