Plan

JEE

JSTL : **J**ava **S**tandard **T**ag **L**ibrary

- Composant de la plate-forme **JEE**.
- Solution pour remplacer les scriptlets.
- Permettant de mieux respecter les bonnes pratiques et en particulier l'architecture **MVC**.
- Objectif : plus de code **Java** dans les pages **JSP**.
- Utilisant des nouvelles balises + **EL** pour remplacer le code **Java**.

JEE

Avantages

- Simplification du code
- Meilleure lisibilité
- Que des balises dans le code
- Maintenance et réutilisation plus facile
- Se protéger des failles **XSS**

JEE

Solution

- Télécharger la bibliothèque à partir du lien suivant

  `https://course.oc-static.com/ftp-tutos/cours/java-ee/jstl-1.2.jar`

- Placer le `.jar` téléchargé (sans le décompresser) dans le répertoire `lib` situé dans `WEB-INF`

Inclure la bibliothèque

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

JEE

Inclure la bibliothèque

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

Afficher un premier message Hello World

```
<c:out value="Hello World" />
```

JEE

Inclure la bibliothèque

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

Afficher un premier message Hello World

```
<c:out value="Hello World" />
```


- `prefix="c"` : indique le préfixe à utiliser pour la bibliothèque `core`
- `c:out` : utilisation de ce préfixe pour afficher un message

JEE

Contenu de la page JSP

```
<%@ page language="java" contentType="text/html; charset=
   UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix
   ="c" %>
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
       charset=UTF-8">
    <title>Projet JEE</title>
  </head>
  <body>
    <c:out value="Hello World" />
  </body>
</html>
```

JEE

Question

Faudrait-il inclure la bibliothèque **JSTL** dans toutes les pages **JSP** ?

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

JEE

Question

Faudrait-il inclure la bibliothèque **JSTL** dans toutes les pages **JSP** ?

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


Réponse

Non, on peut utiliser l'auto-chargement.

JEE

Démarche

- Créer un fichier JSP dans `WEB-INF` que nous appellerons par exemple `jstlLib.jsp`
- Déplacer les directives JSP dans `jstlLib.jsp`

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

- Configurer l'auto-chargement (autoload) depuis `web.xml`

JEE

Ajouter l'auto-chargement dans `web.xml`

```
<jsp-config>
    <jsp-property-group>
        <url-pattern> *.jsp </url-pattern>
        <include-prelude>/WEB-INF/jstlLib.jsp</
           include-prelude>
    </jsp-property-group>
</jsp-config>
```

JEE

Ajouter l'auto-chargement dans `web.xml`

```
<jsp-config>
    <jsp-property-group>
        <url-pattern> *.jsp </url-pattern>
        <include-prelude>/WEB-INF/jstlLib.jsp</
           include-prelude>
    </jsp-property-group>
</jsp-config>
```

Explication

- `<url-pattern> *.jsp </url-pattern>` : pour indiquer les fichiers ciblés par l'auto-chargement
- `<include-prelude> /WEB-INF/jstlLib.jsp </include-prelude>` : le chemin du fichier à auto-charger
- Cela nous évite de faire `<%@ include file="/WEB-INF/taglibs.jsp" %>` dans chaque JSP

JEE

La **JSTL** est composée de 5 librairies

- **Core** : pour les principaux de l'algorithmique (déclaration et gestion de variables, les structures conditionnelles et itératives...)
- **Function** : pour le traitement des chaînes de caractères
- **Format** : pour le formatage de données et l'internationalisation
- **XML** : pour la manipulation des fichiers **XML**
- **SQL** : pour les requêtes **SQL**

JEE

Principales balises de la librairie **Core**

- `out` : pour afficher un message ou le contenu d'une variable
- `set` : pour déclarer ou modifier la valeur d'une variable
- `if`, `choose` et `when` : pour effectuer un traitement conditionnel
- `forEach` et `forTokens` : pour avoir une structure de contrôle itérative
- `url` et `param` : pour construire des liens hypertextes avec et sans paramètre

JEE

Principales balises de la librairie **Core**

- `out` : pour afficher un message ou le contenu d'une variable
- `set` : pour déclarer ou modifier la valeur d'une variable
- `if`, `choose` et `when` : pour effectuer un traitement conditionnel
- `forEach` et `forTokens` : pour avoir une structure de contrôle itérative
- `url` et `param` : pour construire des liens hypertextes avec et sans paramètre

Toutes ces balises s'utilisent avec un préfixe `c` :

JEE

Afficher une valeur

```
<c:out value="JEE" />
<%-- Affiche JEE --%>
```

JEE

Afficher une valeur

```
<c:out value="JEE" />
<%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" />
<%-- Affiche true --%>
```

JEE

Afficher une valeur

```
<c:out value="JEE" />
<%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" />
<%-- Affiche true --%>
```

Afficher le contenu d'une variable avec utilisation de valeur par défaut

```
<c:out value="${ JEE }" default="JSTL"/>
<%-- Affiche le contenu de la variable JEE si elle existe, sinon
   affiche JSTL --%>
```

JEE

Afficher une valeur

```
<c:out value="JEE" />
<%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" />
<%-- Affiche true --%>
```

Afficher le contenu d'une variable avec utilisation de valeur par défaut

```
<c:out value="${ JEE }" default="JSTL"/>
<%-- Affiche le contenu de la variable JEE si elle existe, sinon
   affiche JSTL --%>
```

Une deuxième utilisation de la valeur par défaut

```
<c:out value="${ JEE }" > JSTL </c:out>
```

JEE

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

JEE

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

Exemple

```
<c:out value="<p> Bonjour 'John Wick'. </p>" />
<%-- affiche <p> Bonjour 'John Wick'. </p> --%>
```

JEE

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

Exemple

```
<c:out value="<p> Bonjour 'John Wick'. </p>" />
<%-- affiche <p> Bonjour 'John Wick'. </p> --%>
```

Pour désactiver cette option (excapeXml)

```
<c:out value="<p> Bonjour 'John Wick'. </p>"  escapeXml="false"
    />
<%-- affiche Bonjour 'John Wick'. --%>
```

JEE

Déclarer une variable

```
<c:set var="JEE" value="J'aime la plateforme JEE"
   scope="request" />
```

JEE

Déclarer une variable

```
<c:set var="JEE" value="J'aime la plateforme JEE"
   scope="request" />
```

Explication

- On a déclaré une variable `JEE`
- On l'initialise avec la valeur `J'aime la plateforme JEE`
- On lui affecte la porté `request`

Déclarer une variable de type entier

```
<c:set var="x" value="${ 0 }" />
```

Déclarer une variable de type entier

```
<c:set var="x" value="${ 0 }" />
```

Pour modifier (incrémenter) la valeur de x, on utilise aussi `set`

```
<c:set var="x" value="${ x + 1 }" />
<c:out value="${ x }" />
<%-- Affiche 1 --%>
```

Créer un objet de type Personne à partir de l'objet `perso` défini dans la servlet et ajouté comme attribut de requête

```
<c:set scope="session" var="p" value="${ perso }" />
```

JEE

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

JEE

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

La modification d'un attribut de l'objet

```
<c:set target="$ {perso }" property="nom" value="Travolta" />
<%-- l'objet p aura comme nouveau nom Travolta --%>
<c:out value="${ p.nom } ${ p.prenom }" />
<%-- affiche Travolta John --%>
```

JEE

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

La modification d'un attribut de l'objet

```
<c:set target="$ {perso }" property="nom" value="Travolta" />
<%-- l'objet p aura comme nouveau nom Travolta --%>
<c:out value="${ p.nom } ${ p.prenom }" />
<%-- affiche Travolta John --%>
```

La suppression d'une variable

```
<c:remove var="JEE" />
<%-- supprime la variable JEE --%>
```

JEE

Les structures conditionnelles sans sinon (else)

```
<c:if test="${ 3 > 2 and 2 > 1 }" >
        c'est facile
</c:if>
<%-- affiche c'est facile car la condition est vraie --%>
```

JEE

Les structures conditionnelles sans sinon (else)

```
<c:if test="${ 3 > 2 and 2 > 1 }" >
        c'est facile
</c:if>
<%-- affiche c'est facile car la condition est vraie --%>
```

Explication

- L'attribut `test` est obligatoire
- On peut ajouter deux autres attributs optionnels `scope` et `var`
  - `var` : pour stocker le résultat du test
  - `porté` : pour définir la porté de cette variable

Exemple avec `var` **et** `scope`

```
<c:if test="${ 3 > 2 and 2 > 1 }" var="result" scope
   ="session">
     <c:out value="${ result }" />
</c:if>
<%-- affiche true --%>
```

JEE

Les structures conditionnelles avec un ou plusieurs sinon (else (if))

```
<c:choose>
    <c:when test="${ condition }"> resultat </c:when>
    ...
    <c:otherwise>résultat par défaut</c:otherwise>
</c:choose>
```

JEE

Les structures conditionnelles avec un ou plusieurs sinon (else (if))

```
<c:choose>
    <c:when test="${ condition }"> resultat </c:when>
    ...
    <c:otherwise>résultat par défaut</c:otherwise>
</c:choose>
```


Explication

- `c:choose` : équivalent de `switch`
- `c:when` : équivalent de `case` dans le `switch`
- `c:otherwise` : équivalent de `default` dans le `switch`

JEE

Les structures itératives

```
<c:forEach var="i" begin="0" end="10" step="1">
        <c:out value="${ i }"/>
</c:forEach>
<%-- affiche 0 1 2 3 4 5 6 7 8 9 10 --%>
```

JEE

Les structures itératives

```
<c:forEach var="i" begin="0" end="10" step="1">
        <c:out value="${ i }"/>
</c:forEach>
<%-- affiche 0 1 2 3 4 5 6 7 8 9 10 --%>
```

Explication

- `var` : n'est pas obligatoire. On l'ajoute quand on a besoin d'utiliser la valeur du compteur
- `begin` : valeur initiale du compteur
- `end` : valeur finale de notre compteur
- `step` : le pas à ajouter au compteur après chaque itération

JEE

Pour parcourir une collection

```
<c:forEach items="${ list }" var="element">
    <c:out value="${ element['nom'] }" />
</c:forEach>
```

JEE

Pour parcourir une collection

```
<c:forEach items="${ list }" var="element">
    <c:out value="${ element['nom'] }" />
</c:forEach>
```

Explication

- `items` : pour définir la liste à parcourir
- `var` : pour récupérer l'élément courant de la liste
- On peut aussi ajouter un attribut `varStatus` pour récupérer des informations sur l'itération courante

JEE

Pour parcourir une collection

```
<c:forEach items="${ list }" var="element" varStatus="status">
    Element n : <c:out value="${ status.count }"/> valeur : <c:out
      value="${ element['nom'] }" />
</c:forEach>
```

JEE

Pour parcourir une collection

```
<c:forEach items="${ list }" var="element" varStatus="status">
    Element n : <c:out value="${ status.count }"/> valeur : <c:out
      value="${ element['nom'] }" />
</c:forEach>
```

Les différentes propriétés de `varStatus`

- `first` : contient true si c'est la première itération
- `last` : contient true si c'est la dernière itération
- `step` : contient la valeur de l'attribut `step`
- `count` : contient l'indice de l'itération courante (commence de 1) (sinon `index` commence de 0)
- ...

JEE

Pour parcourir une chaîne de caractère en considérant les trois séparateurs ; ,

```
<c:forTokens var="sousChaine" items="bonjour, c'est
   John;Wick" delims="; , ">
     ${ sousChaine }<br>
</c:forTokens>
```

JEE

Pour parcourir une chaîne de caractère en considérant les trois séparateurs ; ,

```
<c:forTokens var="sousChaine" items="bonjour, c'est
   John;Wick" delims="; , ">
     ${ sousChaine }<br>
</c:forTokens>
```


Explication

- On parcourt une chaîne de caractère par token
- On peut définir un ou plusieurs séparateurs

JEE

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${ monLien }">lien</a>
```

JEE

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${ monLien }">lien</a>
```

`/tapage` est la route d'une Servlet définie soit dans `web.xml` soit avec l'annotation `@WebServlet`

JEE

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${ monLien }">lien</a>
```

`/tapage` est la route d'une Servlet définie soit dans `web.xml` soit avec l'annotation `@WebServlet`

Pour ajouter un lien avec paramètre

```
<c:url value="/tapage" var="monLien">
  <c:param name="nom" value="Wick"/>
  <c:param name="prenom" value="John"/>
</c:url>
<a href="${ monLien }">lien</a>
```

Pour importer un fichier

```
<c:import url="header.jsp"></c:import>
```

Pour importer un fichier

```
<c:import url="header.jsp"></c:import>
```

L'équivalent en scriptlet

```
<%@ include file="header.jsp"%>
```

JEE

Objets implicites

- `pageContext` : pour récupérer le contexte de la page **JSP** (par exemple `pageContext.request.contextPath` pour récupérer le nom du projet)
- `pageScope` : pour récupérer une variable qui a une portée `page`
- `requestScope` : pour récupérer une variable qui a une portée `request`
- `sessionScope` : pour récupérer une variable qui a une portée `session`
- `applicationScope` : pour récupérer une variable qui a une portée `application`
- `param` : pour récupérer les paramètres de requête
- `cookie` : pour récupérer une variable stockée dans un cookie
- ...

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
${ fn:length("chaine") }
<%-- Retourne 6 --%>
```

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
${ fn:length("chaine") }
<%-- Retourne 6 --%>
```

Pour tester si une chaîne contient une autre sous-chaîne de caractère

```
fn:contains("Bonjour", "Bon")
<%-- Retourne true --%>
```

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
${ fn:length("chaine") }
<%-- Retourne 6 --%>
```

Pour tester si une chaîne contient une autre sous-chaîne de caractère

```
fn:contains("Bonjour", "Bon")
<%-- Retourne true --%>
```

Pour extraire une sous-chaîne

```
fn:substring("John Wick", 5, 8)
<%-- Retourne Wick --%>
```

JEE

Autres fonctions

- `fn:trim(String)` : élimine les espaces au début et à la fin de la chaîne
- `fn:toUpperCase(String)` : retourne la chaîne passée en paramètre en majuscule
- `fn:toLowerCase(String)` : retourne la chaîne passée en paramètre en minuscule
- `fn:escapeXml(String)` : élimine les caractères spéciaux en les remplaçant par leur code HTML (Exemple : `${ fn:escapeXml("Les balises <p> & <b>") }` retourne `"Les balises < p > & < b >"`
- ...

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Pour convertir une valeur en monnaie

```
<c:set var="montant" value="112233.44" />
montant = <fmt:formatNumber value="${ montant }"
   type="currency"/>
```

```
<%-- Affiche montant = 112 233,44 € --%>
```

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Pour convertir une valeur en monnaie

```
<c:set var="montant" value="112233.44" />
montant = <fmt:formatNumber value="${ montant }"
   type="currency"/>
```

```
<%-- Affiche montant = 112 233,44 € --%>
```

L'attribut `type` peut prendre d'autres valeurs telles que `percent` et `number`

JEE

Quelques autres attributs

```
<c:set var="montant" value="112233.44" />
montant = <fmt:formatNumber value="${ montant }"
   type="currency"  currencySymbol="$"
   maxIntegerDigits="3"/>
<%-- Affiche montant = 233,44 $     --%>
```

JEE

Autres attributs

- `groupingUsed` : prend `true` pour préciser si les nombres doivent être groupés, `false` sinon.
- `maxFractionDigits` : indique le nombre maximum de chiffres dans la partie décimale
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- `minIntegerDigits`, `minFractionDigits`...

JEE

Pour convertir en nombre

```
<fmt:parseNumber value="${ param.id }" var="id"/>
```

Autres attributs

- `integerOnly` : prend `true` pour un résultat de type entier, float si `false`.
- `scope` : précise la portée de cette variable
- ...

JEE

Pour formatter une date

```
<jsp:useBean id="now" class="java.util.Date" />
Aujourd'hui, c'est le <fmt:formatDate value="${ now
   }" type="date" dateStyle="short"/>
```

Autres valeurs de l'attribut `dateStyle`

- `long` : remplace l'indice du mois par son nom (janvier, février...)
- `full` : même chose que `long` + le nom du jour (lundi, mardi...)
- Autres valeurs : `medium` et `default`

JEE

Autres attributs de `formatDate`

- `timeStyle` : permet de formatter l'heure et prend les mêmes valeurs que `dateStyle`
- `type` : prend une des valeurs suivantes : `date`, `time` ou `both`
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- ...

JEE

Autres attributs de `formatDate`

- `timeStyle` : permet de formatter l'heure et prend les mêmes valeurs que `dateStyle`
- `type` : prend une des valeurs suivantes : `date`, `time` ou `both`
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- ...

Il existe également une balise `parseDate` qui permet de convertir en date et qui prend les mêmes attributs que `formatDate`

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

JEE

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

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

Considérant le fichier XML `personnes.xml` situé dans la racine du projet

```
<personnes>
        <personne id="1">
                <nom>wick</nom>
                <prenom>john</prenom>
        </personne>
        <personne id="2">
                <nom>white</nom>
                <prenom>alain</prenom>
        </personne>
</personnes>
```

JEE

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/TestJstlAtos/
   personnes.xml" var="personnes" />
```

JEE

Pour importer le fichier `Xml`

```
<c:import url="file:/C:/.../eclipse-workspace/TestJstlAtos/
   personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${ personnes }" var="list"  />
```

JEE

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/TestJstlAtos/
   personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${ personnes }" var="list"  />
```

Pour récupérer une personne de la liste des personnes

```
<x:set var="personne" select="$list/personnes/personne[@id=1]"
   />
```

JEE

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/TestJstlAtos/
   personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${ personnes }" var="list"  />
```

Pour récupérer une personne de la liste des personnes

```
<x:set var="personne" select="$list/personnes/personne[@id=1]"
   />
```

Pour afficher le contenu de la balise nom

```
<br/> nom = <x:out select="$personne/nom"/>
```

JEE

Remarques

Comme la librairie **Core**, la librairie **XML** dispose de balises

- `set` pour déclarer une variable,
- `out` pour afficher,
- `if` et `choose` : pour tester
- `forEach` : pour itérer
- ...

JEE

Dépendance à ajouter dans le `pom.xml` d'un `Maven Project` pour utiliser la JSTL

```
<!-- https://mvnrepository.com/artifact/javax.servlet/
   jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

JEE

N'oublions pas de définir un préfixe dans la page JSP

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <body>
    <h2>Hello World!</h2>
    <c:out value="Bonjour" />
  </body>
</html>
```

JEE

N'oublions pas de définir un préfixe dans la page JSP

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <body>
    <h2>Hello World!</h2>
    <c:out value="Bonjour" />
  </body>
</html>
```


Et d'activer `Expression Language`

```
<%@ page isELIgnored="false" %>
```