# Les Fonctions JSTL

Les fonctions de la JSTL sont regroupées en plusieurs bibliothèques qui permettent de traiter des aspects spécifiques du développement web avec JSP. Les principales bibliothèques de la JSTL et de leurs rôles :

1. **Gestion des expressions et des conditions** :
    - Cette partie permet de gérer des conditions logiques et de manipuler les données de manière dynamique. Par exemple, on peut tester si une certaine condition est vraie ou fausse et adapter le rendu de la page en conséquence.
2. **Manipulation des collections et des itérations** :
    - Elle permet de parcourir des listes, des tableaux, ou des ensembles de données. C'est très utile pour afficher des éléments de manière répétée, comme une liste de produits ou des résultats de recherche.
3. **Formatage des données** :
    - Les fonctions de formatage permettent de gérer l'affichage des nombres, des dates, des devises et d'autres données dans un format précis, adapté à une localisation spécifique. Elles aident à rendre les données lisibles pour les utilisateurs finaux.
4. **Manipulation des URL et des accès aux ressources** :
    - Cette bibliothèque est destinée à faciliter l'accès aux ressources, comme les liens et les chemins d'accès aux fichiers ou images, en les construisant dynamiquement en fonction de l'environnement de l'application.
5. **Internationalisation (i18n) et localisation (l10n)** :
    - Elle est dédiée à la gestion de l'affichage multilingue dans les pages JSP. Elle permet de prendre en compte les préférences linguistiques des utilisateurs pour afficher les textes et les formats de données dans leur langue et format préférés.
6. **Fonctions pour la gestion des chaînes de caractères** :
    - Ces fonctions offrent la possibilité de manipuler des chaînes de caractères, comme vérifier la longueur d'une chaîne, la découper, la convertir en majuscules ou en minuscules, et bien d'autres transformations courantes.

Ces bibliothèques rendent les pages JSP plus dynamiques et permettent aux développeurs de se concentrer sur la logique métier tout en simplifiant la gestion de la présentation et des données sur les pages web.

Pour inclure la bibliothèque de fonctions JSTL dans une page JSP, on utilise la syntaxe suivante :

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

Les 15 fonctions les plus courantes de la bibliothèque **fn** de JSTL :

1. **fn:contains()** :
    - Vérifie si une chaîne de caractères contient une sous-chaîne spécifique.
    - Retourne true si la sous-chaîne est trouvée, sinon false.

      **Exemple :**

      ```
      <c:set var="message" value="Hello, world!" />
      <c:if test="${fn:contains(message, 'world')}">
       <p>'world' est présent dans le message.</p>
      </c:if>
      ```

2. **fn:containsIgnoreCase()** :
    - Vérifie si une chaîne de caractères contient une sous-chaîne spécifique, en ignorant la casse.
    - Retourne true même si la casse ne correspond pas exactement.
      **Exemple :**

      ```
      <c:set var="message" value="Hello, world!" />
      <c:if test="${fn:containsIgnoreCase(message, 'WORLD')}">
        <p>'WORLD' est trouvé, sans tenir compte de la casse.</p>
      </c:if>
      ```

3. **fn:endsWith()** :
    - Vérifie si une chaîne de caractères se termine par une sous-chaîne donnée.
    - Utile pour vérifier les suffixes, comme les extensions de fichiers.
      **Exemple :**

      ```
      <c:set var="filename" value="document.pdf" />
      <c:if test="${fn:endsWith(filename, '.pdf')}">
        <p>Le fichier est un PDF.</p>
      </c:if>
      ```

4. **fn:startsWith()** :
    - Vérifie si une chaîne commence par une sous-chaîne spécifique.
    - Peut être utilisé pour des vérifications de préfixes.
      **Exemple :**

      ```
      <c:set var="message" value="Bonjour tout le monde" />
      <c:if test="${fn:startsWith(message, 'Bonjour')}">
        <p>Le message commence par 'Bonjour'.</p>
      </c:if>
      ```

5. **fn:length()** :

- Retourne la longueur d'une chaîne de caractères, d'une collection ou d'un tableau.
- Utile pour compter le nombre de caractères ou d'éléments.

  Exemple :

  <c:set var="message" value="JavaServer Pages" />
  Longueur du message : ${**fn:length(**message)}

6. **fn:substring()** :
   - Extrait une partie d'une chaîne de caractères en fonction des indices de début et de fin fournis.
   - Permet d'obtenir une sous-chaîne spécifique.

     **EXEMPLE**
     <c:set var="message" value="Hello, world!" />
     Sous-chaîne : ${**fn:substring(**message, 7, 12)}

7. **fn:substringAfter()** :
   - Extrait la partie de la chaîne située après la première occurrence d'une sous-chaîne spécifiée.
   - Pratique pour récupérer ce qui suit un mot-clé particulier.

     **Exemple**
     <c:set var="email" value="user@example.com" />
     Domaine : ${**fn:substringAfter(**email, '@')}

8. **fn:substringBefore()** :
   - Extrait la partie de la chaîne située avant la première occurrence d'une sous-chaîne donnée.
   - Idéal pour récupérer la partie avant un séparateur, comme un domaine avant le @ dans une adresse e-mail.

     **Exemple :**

     **<c:set var="email" value="user@example.com" />**
     Nom d'utilisateur : ${**fn:substringBefore(**email, '@')}

9. **fn:trim()** :
   - Supprime les espaces blancs en début et fin de chaîne.
   - Utile pour nettoyer les entrées des utilisateurs.

**Exemple :**

```
<c:set var="message" value="   Bonjour   " />
<p>Message sans espaces : '${fn:trim(message)}'</p>
```

10. **fn:toLowerCase()** :

- Convertit une chaîne de caractères en minuscules.
- Permet de normaliser les textes pour des comparaisons insensibles à la casse.

**Exemple :**

```
<c:set var="message" value="HELLO" />
<p>Minuscule : ${fn:toLowerCase(message)}</p>
```

11. **fn:toUpperCase()** :

- Convertit une chaîne de caractères en majuscules.
- Utile pour mettre en valeur des titres ou des messages.

**Exemple :**

```
<c:set var="message" value="hello" />
<p>Majuscule : ${fn:toUpperCase(message)}</p>
```

12. **fn:replace()** :

- Remplace toutes les occurrences d'une sous-chaîne par une autre dans une chaîne.
- Pratique pour faire des modifications de texte, comme remplacer des caractères spéciaux.

**Exemple :**

```
<c:set var="message" value="Java is fun, Java is powerful" />
<p>${fn:replace(message, 'Java', 'JSP')}</p>
```

13. **fn:join()** :

- Concatène les éléments d'une collection ou d'un tableau en une seule chaîne, séparés par un délimiteur donné.
- Idéal pour créer une liste de valeurs séparées par des virgules.

**Exemple :**

```
<c:set var="list" value="${['Java', 'JSP', 'Servlets']}" />
<p>${fn:join(list, ', ')}</p>
```

14. **fn:split()** :

- Sépare une chaîne en un tableau en utilisant un délimiteur.
- Par exemple, permet de diviser une liste de mots séparés par des virgules.

**Exemple :**

```
<c:set var="csv" value="apple,banana,cherry" />
```

```
<c:forEach var="fruit" items="${fn:split(csv, ',')}">
    <p>Fruit : ${fruit}</p>
</c:forEach>
```

15. **fn:indexOf()** :

- Retourne la position de la première occurrence d'une sous-chaîne dans une chaîne de caractères.
- Retourne -1 si la sous-chaîne n'est pas trouvée.

**Exemple :**

```
<c:set var="message" value="Hello, world!" />
<p>Position de 'world' : ${fn:indexOf(message, 'world')}</p>
```

**NB :** Il y a d'autres fonctions que je n'ai pas développer comme **fn:escapeXml()** donc c'est à vous de jouer