JSTL

Guillaume Dufrêne – Lionel Seinturier

Université de Lille – Sciences et Technologies

Définition

JSTL = JavaServer Pages Standard Tag Library

ensemble de balises (t*ags*) en JSP

- boucle
- test
- redirection
- URL
- exception

Utilisation

- Vues d'une application MVC
- Faciliter la manipulation des données à l'affichage
- Se prémunir des injections de code JavaScript (XSS)

- Organisées en librairies (*taglib*)
  - core
  - Spring form
  - XML
  - SQL
  - Internationalisation (i18n)

- Développer sa propre *taglib*

Ma première JSTL

Exemple – modèle (données)

```
public class Personne {

  private int id;
  private String nom;
  private int age;

  public Personne( int id, String nom, int age ) {
    this.id=id; this.nom=nom; this.age=age;
  }

  // les méthodes getId, getNom, getAge
  // les méthodes setId, setNom, setAge
}
```

Ma première JSTL

Exemple – contrôleur (servlet)

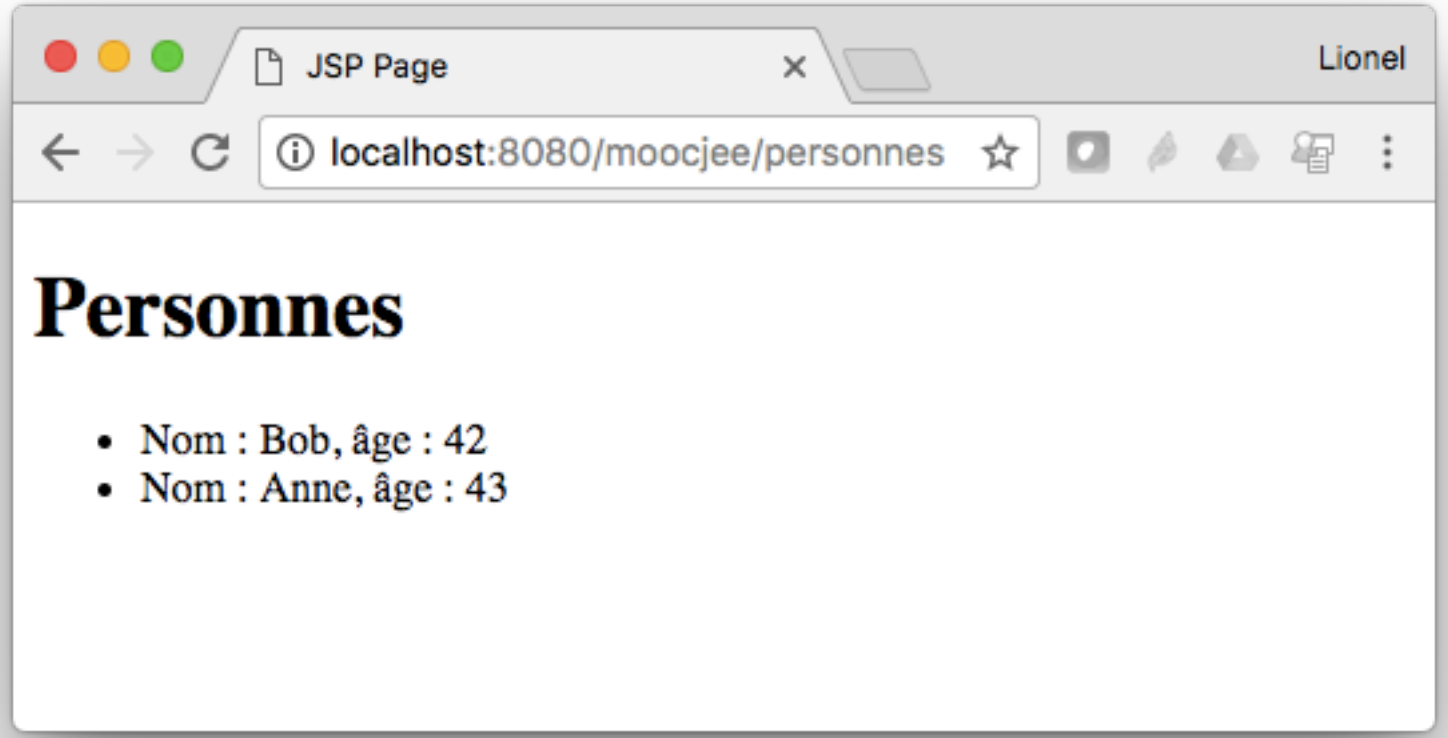
```
@WebServlet(urlPatterns={"/personnes"})
public class PersonnesServlet extends HttpServlet {

  public void service( HttpServletRequest req, HttpServletResponse resp )
  throws ServletException, IOException {

    List<Personne> l = new ArrayList<>();
    l.add( new Personne(1, "Bob", 42) );
    l.add( new Personne(2, "Anne", 43) );
    req.setAttribute("lesPersonnes", l);

    request.getRequestDispatcher("/vuePersonnes.jsp").forward(req,resp);
  }
}
```

Ma première JSTL

Exemple – vue (JSP)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <h1>Personnes</h1>
    <c:choose>
      <c:when test="${empty lesPersonnes}">
        Liste des personnes vide
      </c:when>
      <c:otherwise>
      <ul>
       <c:forEach items="${lesPersonnes}" var="p">
        <li>Nom : ${p.nom}, âge : ${p.age}</li>
       </c:forEach>
      </ul>
      </c:otherwise>
    </c:choose>
</body>
</html>
```

Résultat


JSP Page
Lionel
localhost:8080/moocjee/personnes
Personnes
Nom : Bob, âge : 42
Nom : Anne, âge : 43

Les principales balises de la taglib core

<c:catch>	gère les exception 'Throwable' qui surviennent dans le contenu du tag
<c:if>	test conditionnel
<c:choose>	similaire à un `switch`
<c:when>	sous-tag de 'choose' : équivalent d'un `case`
<c:otherwise>	sous-tag de 'choose' : équivalent d'un `default`
<c:import>	inclus le contenu d'une ressource
<c:forEach>	boucle sur un type Iterable ou tableaux
<c:forTokens>	boucle sur une chaine possédant un séparateur
<c:redirect>	redirige l'utilisateur vers une nouvelle URL
<c:url>	retourne l'URL absolue en fonction du contexte de la servlet
<c:param>	sous-tag de 'URL' permettant d'ajouter un paramètre

La taglig form de Spring

Faciliter les intéraction avec les formulaires HTML

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<body>
  <h1>Personne</h1>
  <form:form modelAttribute="unePersonne">
      <form:input type="hidden" path="id" /> <br/>
      Nom : <form:input path="nom" /> <br/>
      Age : <form:input path="age" /> <br/>
      <input type="submit" value="Enregistrer"/>
  </form:form>
</body>
</html>
```

La taglig form de Spring

Un contrôleur avec GET et POST

```
// Affichage du formulaire

@GetMapping(value="/personne/{id}.html")
public String edit( Model model, @PathVariable("id") int id ) {
  Personne p = dao.find(id);
  model.addAttribute("unePersonne", p);
  return "personne/edit";
}

// Enregistrement des données du formulaire

@PostMapping(value="/personne/{id}.html")
public String save( Model model, @ModelAttribute Personne unePersonne ) {
  dao.update(unePersonne);
  model.addAttribute("unePersonne", unePersonne);
  return "personne/edit";
}
```

En résumé

JSTL

- un ensemble de balises JSP
- traitement des données transmises aux vues

- différentes *taglibs* existantes
  - core
  - spring-form : manipulation de formulaires