

De l'esprit à la machine
L'approche Professo-Académique

J2EE

Abdelahad SATOUR

Séances 17–18

JSP

Compléments sur l'API

Méthodes appelables sur l'objet prédéfini `request`

- `String getProtocol()`
retourne le protocole implanté par le serveur (ex. : HTTP/1.1)
- `String getServerName()` / `String getServerPort()`
retourne le nom/port de la machine serveur
- `String getRemoteAddr()` / `String getRemoteHost()`
retourne l'adresse/nom de la machine cliente (ayant invoqué la servlet)
- `String getScheme()`
retourne le protocole utilisé (ex. : http ou https) par le client

Suivi de session

- HTTP protocole non connecté
- pour le serveur, 2 requêtes successives d'un même client sont **indépendantes**

Objectif : être capable de "suivre" l'activité du client sur +sieurs pages

Notion de session

- ⇒ les **requêtes** provenant d'un **utilisateur** sont associées à une même session
- ⇒ les sessions ne sont pas éternelles, elles **expirent** au bout d'un délai fixé

Objet prédéfini `session` de type `HttpSession`

- ⇒ la session courante ou une nouvelle session

Suivi de session

Méthodes appelables sur l'objet prédéfini `session`

- `void setAttribute(String name, Object value)`
ajoute un couple (name, value) pour cette session
- `Object getAttribute(String name)`
retourne l'objet associé à la clé name ou null
- `void removeAttribute(String name)`
enlève le couple de clé name
- `java.util.Enumeration getAttributeNames()`
retourne tous les noms d'attributs associés à la session
- `void setMaxIntervalTime(int seconds)`
spécifie la durée de vie maximum d'une session
- `long getCreationTime() / long getLastAccessedTime()`
retourne la date de création / de dernier accès de la session
en ms depuis le 1/1/1970, 00h00 GMT → `new Date(long)`

Partage de données entre JSP

Notion de contexte d'exécution

= ensemble de couples (name, value) partagées par toutes les JSP **instanciées**

⇒ objet prédéfini `application`

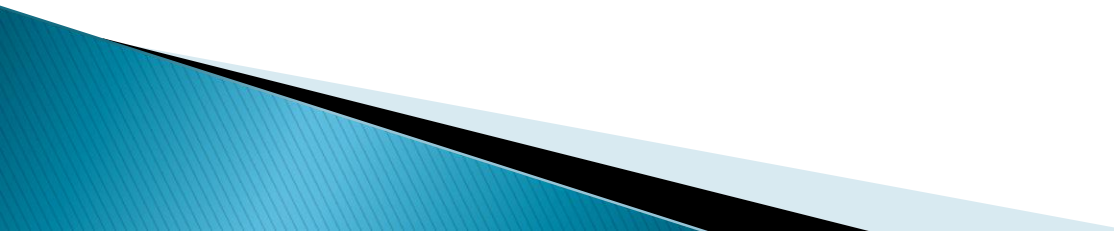
Méthodes appelables sur l'objet prédéfini `application`

- `void setAttribute(String name, Object value)`
ajoute un couple (name, value) dans le contexte
- `Object getAttribute(String name)`
retourne l'objet associé à la clé name ou null
- `void removeAttribute(String name)`
enlève le couple de clé name
- `java.util.Enumeration getAttributeNames()`
retourne tous les noms d'attributs associés au contexte

Conclusion

Permettent d'étendre le comportement des serveurs Web avec des prog. Java

Résumé des fonctionnalités

- + code embarqué dans un fichier HTML
 - + portabilité, facilité d'écriture (**Java**)
 - + gestion des applications requérant un suivi entre plusieurs programmes (**persistance des données**)
 - + JSP chargée et instanciée **une seule fois**
 - + JSP exécutée avec des processus légers (*threads*)
- 

Les tags JSP (ou actions)

- ▶ Les tags sont des actions incluses dans une page Web suivant la syntaxe XML
 - `<mod:tag attr="value">`
 - `<mod:tag attr="value">body</mod:tag>`
- ▶ Les actions de base font partie de la librairie jsp:
 - `<jsp:useBean>`
 - `<jsp:setProperty>`
 - `<jsp:getProperty>`
 - `<jsp:include>`
 - `<jsp:forward>`
 - `<jsp:text>`

Java Beans

- But : avoir le moins de code Java possible dans une page JSP (HTML)
- Sous-traiter le code à un Java bean
- balise XML : `<jsp:useBean>`

Java Beans

- Syntaxe générale :

```
<jsp:useBean id="nomInstanceJavaBean"  
class="nomClasseDuBean"  
scope="request|session|application|page"  
>  
</jsp:useBean>
```

- Le bean est alors utilisable par *nomInstanceJavaBean*
- balise sans corps donc utilisation de `<jsp:useBean ... />`

l'attribut scope

- Il indique la portée du bean.

valeur	Description
request	Le bean est valide pour cette requête. Il est utilisable dans les pages de redirection de la requête (<code><jsp:forward></code>). Il est détruit à la fin de la requête.
page	Similaire à <code>request</code> , mais le bean n'est pas transmis aux pages de redirection <code><jsp:forward></code> . C'est la portée par défaut
session	Le bean est valide pour la session courante. S'il n'existe pas encore dans la session courante, il est créé et placé dans la session du client. Il est réutilisé jusqu'à ce que la session soit invalidée
application	Le bean est valide pour l'application courante. Il est créé une fois et partagé par tous les clients des JSP.

JSP et Java beans : exemple

- Soit le bean :

```
public class SimpleBean implements java.io.Serializable
{
    private int compter;

    public SimpleBean() {
        compter = 0;
    }

    public void setCompter(int theValue) {
        compter = theValue;
    }

    public int getCompter() {
        return compter;
    }

    public void increment() {
        compter++;
    }
}
```

Utilisation du bean dans une JSP

- Utilisation à l'aide de son nom
- Récupération des propriétés :
 - Par appel de méthode `getXXX()` :
 - Par la balise `<jsp:getProperty ...>`

```
<p> on repere le bean par le nom nomBean<br>
<jsp:useBean id="nomBean" class="SimpleBean" scope="session">
</jsp:useBean>
<p> On accede a une propriete avec une expresion:
<br> compteur = <%= nomBean.getCompter() %>
<hr>
On incremente le compteur <% nomBean.increment(); %>
<p>On peut acceder à la propriété par une balise :<br>
<jsp:getProperty name="nomBean" property="compter" />
```

Utilisation du bean dans une JSP

- Par appel de méthode `setXXX (...)` :
- Par la balise `<jsp:setProperty ...>`

```
<p> on repere le bean par le nom nomBean<br>  
<jsp:useBean id="nomBean" class="SimpleBean" scope="session">  
</jsp:useBean>
```

```
<p> On positionne une propriété avec une expresion:  
<br> compteur = <%= nomBean.setCompter(6) %>
```

```
<p>ou par une balise :<br>  
<jsp:setProperty name="noBean" property="compter" value="6" />
```

Un JavaBean

- ▶ Composant simple.
Respecte des
conventions
d'écriture

```
public class MyBean {  
    private String nom;  
    private int compte;  
    private Date date;  
  
    public String getNom() {  
        return nom;  
    }  
  
    public void setNom(String nom) {  
        this.nom = nom;  
    }  
  
    public int getCompte() {  
        return compte;  
    }  
  
    public void setCompte(int compte) {  
        this.compte = compte;  
    }  
  
    public Date getDate() {  
        return date;  
    }  
}
```


Exemple UseBean

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h2>Use Bean</h2>
    <jsp:useBean id="mybean" scope="page" class="exemple.MyBean" />
    <jsp:setProperty name="mybean" property="nom" value="Test" />
    <jsp:setProperty name="mybean" property="compte" value="0" />

    <jsp:getProperty name="mybean" property="nom"/>
  </body>
</html>
```

Autre exemple

```
<body>
  <h2>Use Bean 2</h2>
  <jsp:useBean id="bean2" scope="page" class="exemple.MyBean" />
  <jsp:setProperty name="bean2" property="nom" param="nom" />
  Valeur du paramètre = <jsp:getProperty name="bean2" property="nom" />
</body>

<body>
  <h2>Use Bean 3</h2>
  <jsp:useBean id="mybean" scope="request" class="exemple.MyBean" />
  <jsp:setProperty name="mybean" property="*" />
  Nom= <jsp:getProperty name="mybean" property="nom" /> <br>
  Compte = <jsp:getProperty name="mybean" property="compte" />
</body>
```

 <http://localhost:8084/CoursWeb/useBean3.jsp?nom=test&compte=100>

Use Bean 3

Nom= test

Compte = 100

useBean et scope

```
<body>
  <h2>Scope 1</h2>
  <jsp:useBean id="aBean" scope="session" class="exemple.MyBean" />
  <jsp:setProperty name="aBean" property="*" />
</body>
```

```
<body>
  <h2>Scope 2</h2>
  <jsp:useBean id="aBean" scope="session" class="exemple.MyBean" />
  <jsp:getProperty name="aBean" property="nom" />
</body>
```

Scope 2

test

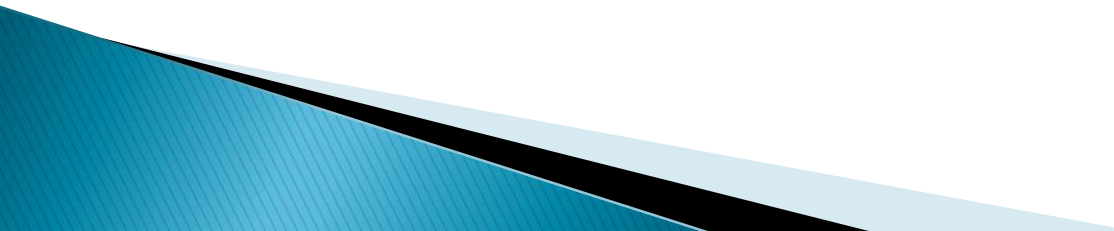
Code généré

```
synchronized (session) {  
    aBean = (exemple.MyBean) _jspx_page_context.getAttribute("aBean", PageContext.SESSION_SCOPE);  
    if (aBean == null){  
        aBean = new exemple.MyBean();  
        _jspx_page_context.setAttribute("aBean", aBean, PageContext.SESSION_SCOPE);  
    }  
}
```

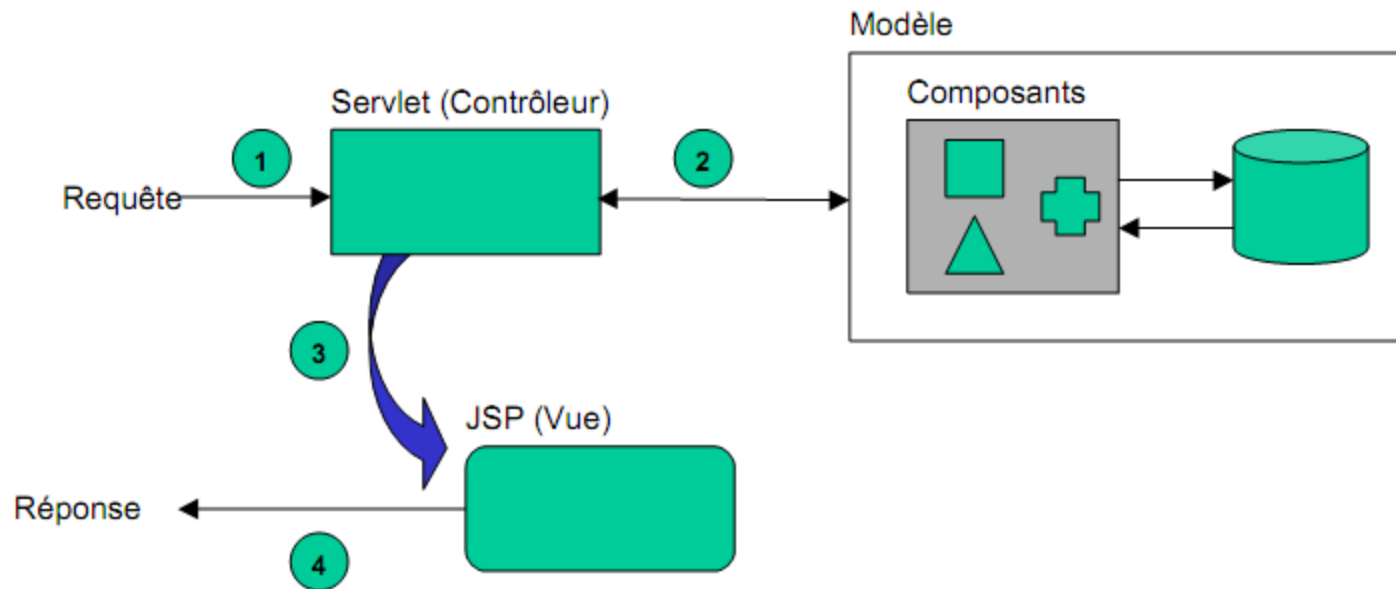
JavaBean et JSP

- ▶ Les action `useBean`, `setProperty` et `getProperty` permettent de manipuler des JavaBean sans programmation
 - `jsp:usebean` pour nommer, créer ou désigner un bean
 - `jsp:getProperty` pour récupérer une propriété d'un bean
 - `jsp:setProperty` pour changer la valeur d'une propriété.

MVC

- modèle = les données accédées par un code Java (JDBC, RMI, EJB, etc.)
 - vues = JSP
 - contrôleur = servlets
- 

MVC



- Syntaxe dans la servlet pour lancer la JSP :

```
public void doPost(HttpServletRequest request, HttpServletResponse response){  
    ServletContext context = getServletContext(); // héritée de GenericServlet  
    RequestDispatcher dispatcher =  
        context.getRequestDispatcher("/maPageMiseEnForme.jsp");  
    dispatcher.forward(request, response);  
}
```


MVC

- La servlet peut passer des valeurs à la JSP appelé grâce à `setAttribute()`

```
public void doPost(HttpServletRequest request, HttpServletResponse response) {  
    // appelle les méthodes sur les objets métiers  
    ArrayList theList = // un objet à passer  
    // ajoute à la requête  
    request.setAttribute("nomDelObjet", theList);  
    ServletContext context = getServletContext();  
    RequestDispatcher dispatcher = context.getRequestDispatcher("/jspAAppeler.jsp");  
    dispatcher.forward(request, response);  
}
```

- La JSP extrait les objets de request grâce à `getAttribute()`

```
<% ArrayList theList = (ArrayList)  
    request.getAttribute("nomDelObjet");  
    // maintenant, utiliser l'ArrayList  
%>
```

Programmation Web

JSP et Taglib

1. Les TagLibs

- ▶ La spécification des JSP prévoit la possibilité de définir ses **propres tags** XML
- ▶ Un ensemble de tag est regroupé au sein d'une **taglib** qui possède URI comme identifiant unique

Versions

- ▶ La **JSTL 1.0** nécessite au minimum un conteneur **JSP 1.2 (J2EE 1.3)**.
- ▶ La **JSTL 1.1** nécessite au minimum un conteneur **JSP 2.0 (J2EE 1.4)**.

JSP Standard Tag Library

- ▶ Ensemble de tags pré-programmés permettant d'effectuer les opérations standard
- ▶ Initiative Apache spécifiée via la JSR052
- ▶ Plusieurs parties :
 - Core (<http://java.sun.com/jstl/core>),
 - Internationalisation (<http://java.sun.com/jstl/fmt>),
 - XML (<http://java.sun.com/jstl/xml>),
 - SQL (<http://java.sun.com/jstl/sql>).

JSP Standard Tag Library 1.2

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

JSP Standard Tag Library

- ▶ Ensemble de tags pré-programmés permettant d'effectuer les opérations standard
- ▶ Initiative Apache spécifiée via la JSR052
- ▶ Plusieurs parties :
Core (<http://java.sun.com/jstl/core>),
Internationalisation (<http://java.sun.com/jstl/fmt>),
XML (<http://java.sun.com/jstl/xml>),
SQL (<http://java.sun.com/jstl/sql>).

Scope

Scope	Description
page	La nouvelle valeur n'affecte que la page JSP courante.
request	La nouvelle valeur affecte toute la requête courante (pages JSP forwardées/incluses compris).
session	La nouvelle valeur affecte toute la session de l'utilisateur.
application	La nouvelle valeur affecte tous les utilisateurs.

2. `<c:/>` : Librairie de base



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

2.1. Gestion des variables de scope

- ▶ Cette section comporte les actions de base pour la gestion des variables de scope d'une application web :
 - L'affichage de variable
 - La création/modification/suppression de variable de scope
 - La gestion des exceptions

2.1. Gestion des variables de scope

► `<c:out/>` : Afficher une expression

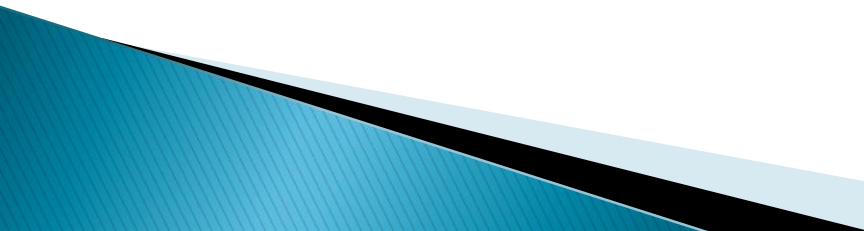
Attribut	R e q.	Type	Description
value	o ui	Object	L'expression qui sera évaluée et affichée. Si le type réel implémente java.io.Reader , alors c'est son contenu qui sera affiché.
default		Object	Valeur à afficher si l'expression value est null (défaut : "").
escapeXML		booleen	Détermine si les caractères <code><</code> , <code>></code> , <code>&</code> , <code>'</code> , <code>"</code> doivent être remplacés par leurs codes respectifs : <code>&lt;</code> , <code>&gt;</code> , <code>&amp;</code> , <code>&#039;</code> , <code>&#034;</code> (défaut : true).

2.1. Gestion des variables de scope

- ▶ TP

JSTL Expression Language

Variables prédéfinies :

- ▶ **applicationScope** ensemble des variables disponible par ordre de visibilité
 - ▶ **cookie** Collection de tous les cookies
 - ▶ **header** entête HTTP vue comme une String
 - ▶ **headerValues** entête vue comme une collection de String
 - ▶ **initParam** Collection des paramètre d'initialisation
 - ▶ **pageContext** objet représentant la page courante
 - ▶ **pageScope** collection des variables de la page
 - ▶ **param** paramètres sous forme de String
 - ▶ **paramValues** paramètres sous forme d'une collection de String
 - ▶ **requestScope** collection des variables de la requête
 - ▶ **sessionScope** collection des variables de la session
- 

2.1. Gestion des variables de scope

- ▶ **<c:set/> : Définir une variable de scope ou une propriété**

2.1. Gestion des variables de scope

Attribut	R e q.	Type	Description
value		Object	L'expression à évaluer.
var		String	Nom de l'attribut qui contiendra l'expression dans le scope.
scope		String	Nom du scope qui contiendra l'attribut var (page , request , session ou application) (défaut : page).
target		Object	L'objet dont la propriété définit par property sera modifiée. Il doit correspondre soit à un bean avec la méthode mutateur correspondante (setProperty), soit à un objet de type java.util.Map .
property		String	Nom de la propriété qui sera modifiée.

2.1. Gestion des variables de scope

Attribut	R e q.	Type	Description
value		Object	L'expression à évaluer.
var		String	Nom de l'attribut qui contiendra l'expression dans le scope.
scope		String	Nom du scope qui contiendra l'attribut var (page , request , session ou application) (défaut : page).
target		Object	L'objet dont la propriété définit par property sera modifiée. Il doit correspondre soit à un bean avec la méthode mutateur correspondante (setProperty), soit à un objet de type java.util.Map .
property		String	Nom de la propriété qui sera modifiée.