Java Server Pages

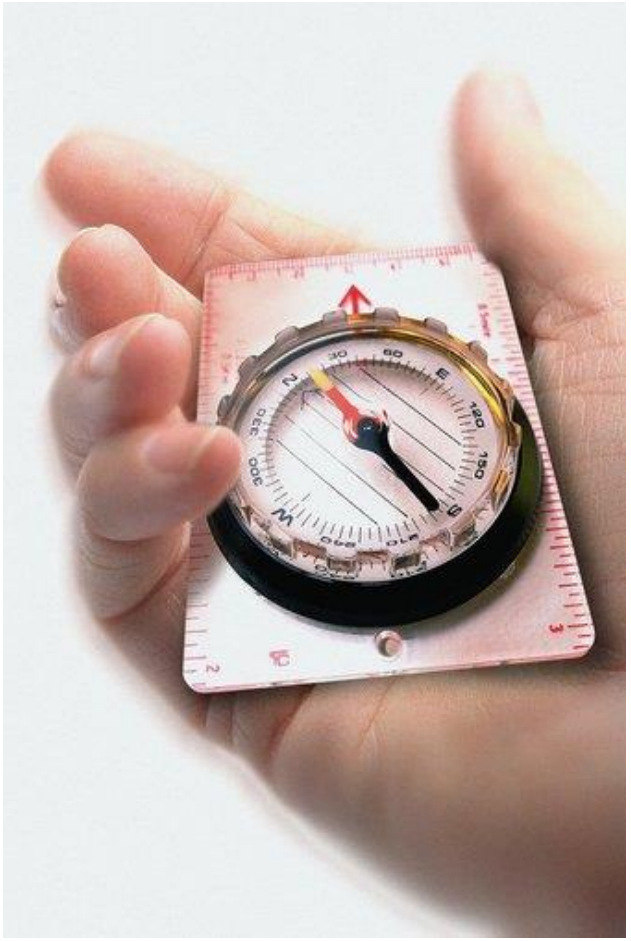
Java web pages

Objectifs du cours

En complétant ce cours, vous serez en mesure de:

- Expliquer ce que sont les JSP
- Utiliser les pages du serveur Java
- Utiliser l'architecture JSP modèle 2

Plan de cours



- Presentation
- Éléments d'action
- Langage d'expression
- Bibliothèques de balises
- JSP et servlets

Java Server Pages

PRESENTATION

Que sont les JSP ?

- Pages Web, mais version Java
- Les JSP et les servlets fonctionnent de manière complémentaire
  - Utilisé pour séparer le traitement des requêtes et la génération de code HTML
- Peut contenir du code Java
  - Comme les pages PHP contiennent… du code PHP
  - Grâce aux éléments de script
- Peut contenir des balises spéciales pour éviter le code Java

Presentation

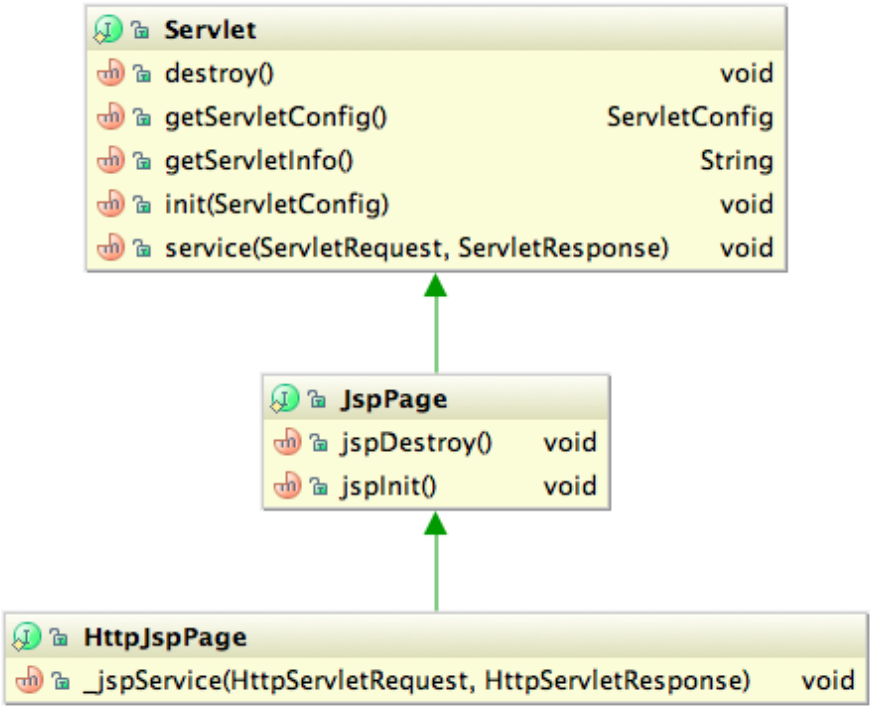
Que sont les JSP ?

- JSP exemple:

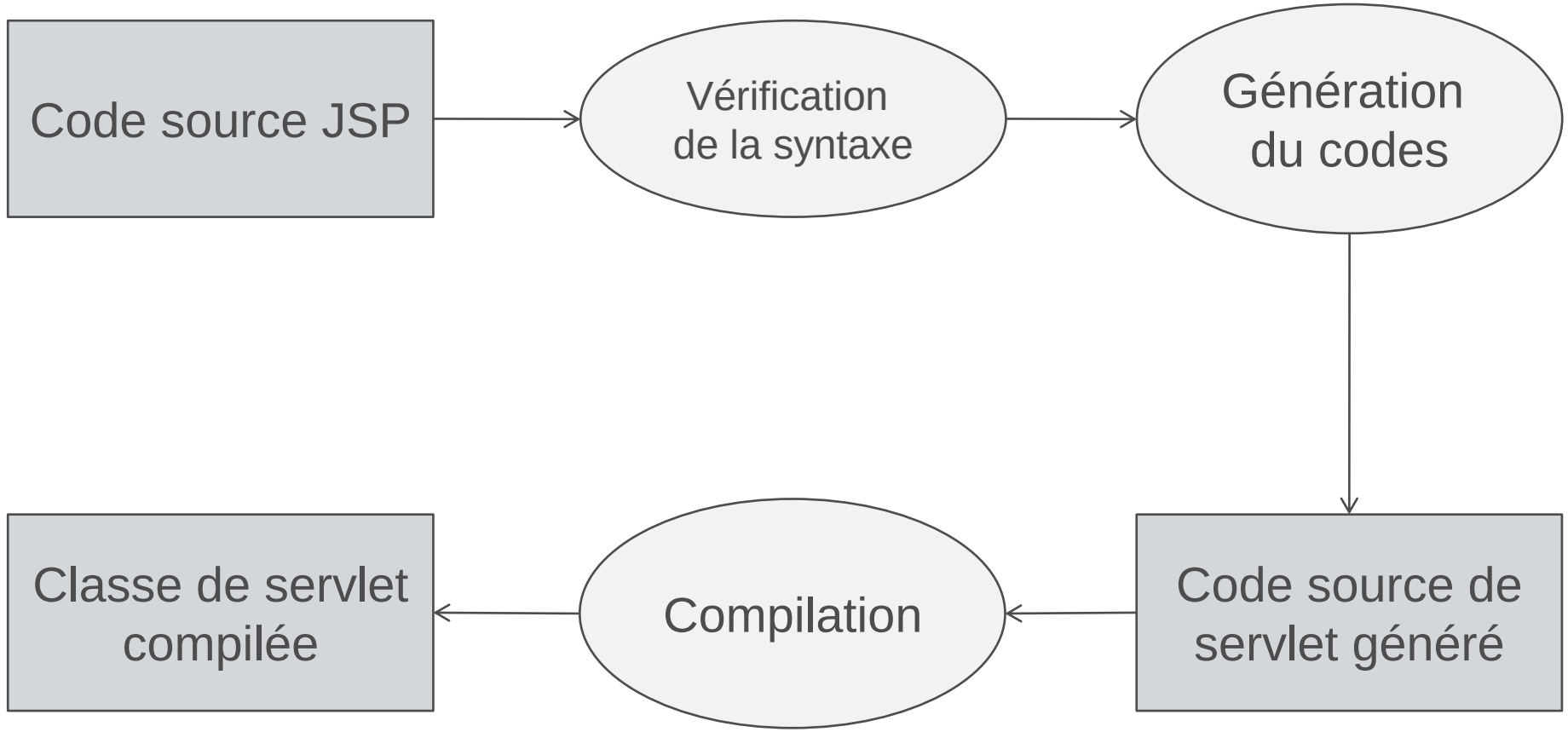
```
<%@ page language="java" %>
<%@ page import="java.util.Date" %>
<html>
    <head>
        <title>When am I ?</title>
    </head>
    <body>
        <%= new Date() %>
    </body>
</html>
```

Cycle de vie JSP

- A la première requête, les pages JSP sont traduites en code Java (plus exactement en une classe HttpJspPage)
  - Qui étend Servlet ☺



Cycle de vie JSP



Presentation

Cycle de vie JSP

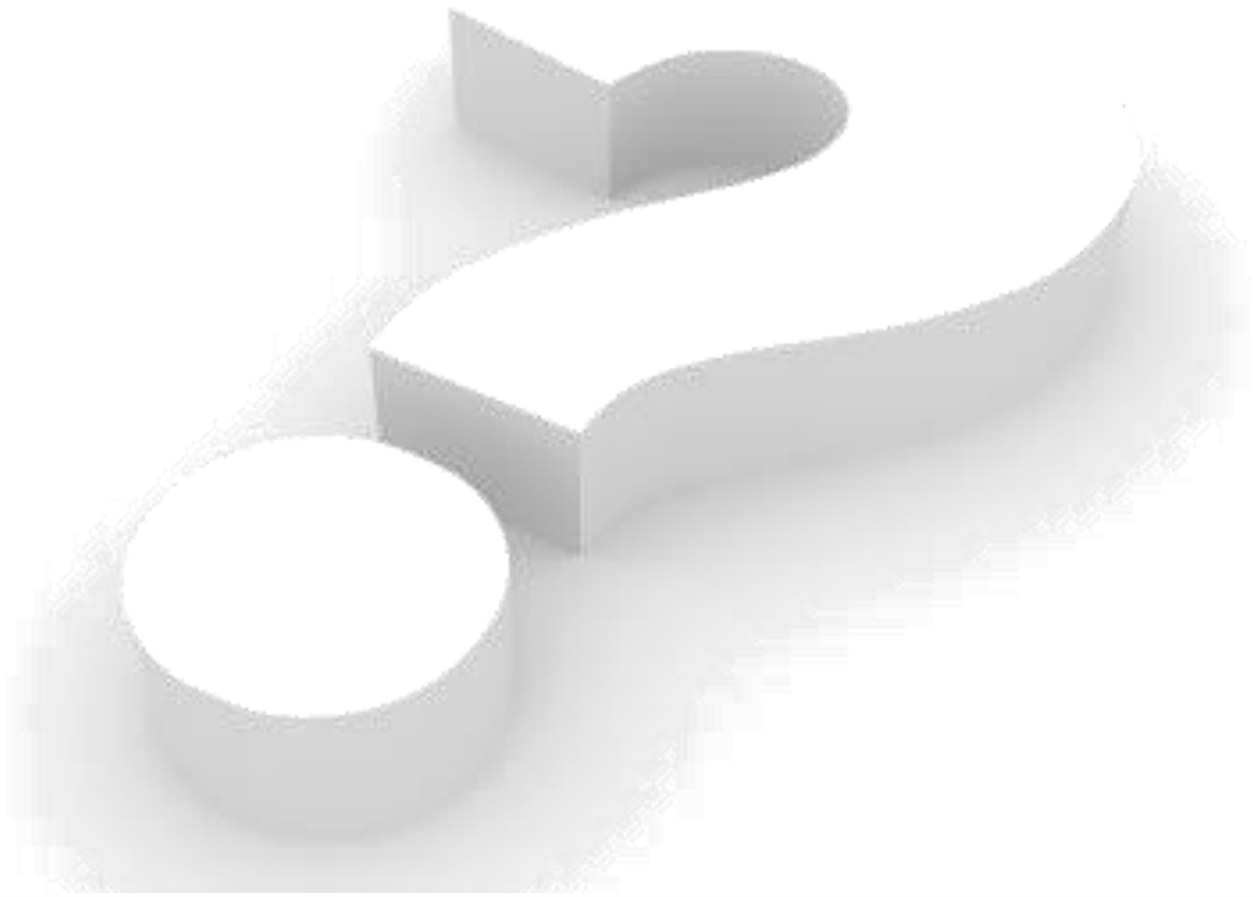
- Exemple JSP traduit :

```
<%@ page language="java" %>
<%@ page import="java.util.Date" %>
<html>
    <head>
        <title>When am I ?</title>
    </head>
    <body>
        <%= new Date() %>
    </body>
</html>
```

Translated JSP Example

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response) {
   //...
   PageContext pageContext = null;
   JspWriter out = null;
   //...
   pageContext = _jspxFactory.getPageContext(this,
      request, response, null, true, 8192, true);
   out = pageContext.getOut();
   //...
   out.write("\t<body>\n");
   out.print( new Date() );
   out.write("\n");
   out.write("\t</body>\n");
   //...
}
```

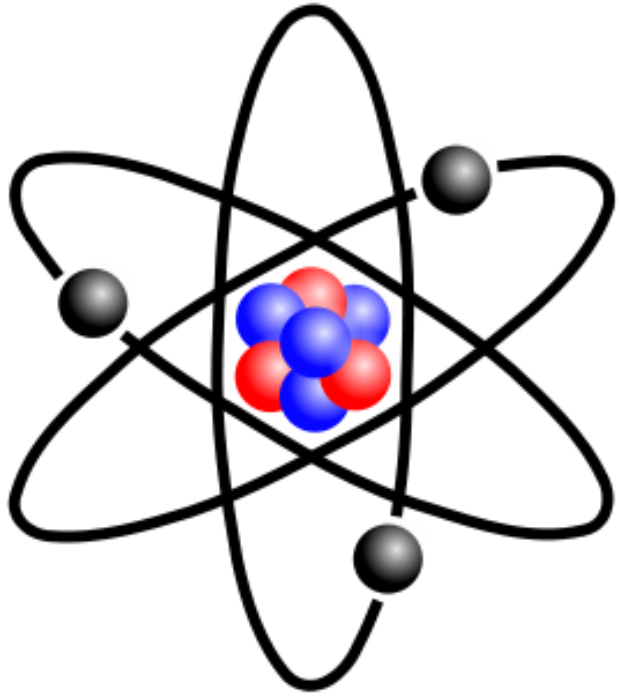
Questions ?



Java Server Pages

ÉLÉMENTS JSP

Directive, action et script



Présentation

- Trois types d'élément JSP :
  - Éléments de script
  - Éléments directifs
  - Éléments d'action

Presentation

Element	Description
<% %>	**Scriptlets** : permettent d'écrire du code Java à l'intérieur des JSP.
<%= %>	**Expression** : Affiche la valeur d'une variable ou retournée par une méthode.
<%! %>	**Déclaration** : déclare un morceau de code (variable, méthode,…) en dehors de la méthode _jspService(…). *Attention* : son utilisation pour déclarer une variable modifiable n'est pas sûre pour le multithreading !
<%-- --%>	**Commentaires** : insérer des commentaires qui ne seront pas visibles dans la réponse HTML générée.

Éléments de script

```
<% String s1 = new String("Hello ! "); %>
<% for(int i = 0; i < 3 ; i++)  { %>
     <%= s1 %>
<% } %>

<br />

<%-- No compilation error because s2
                        is in a declaration ! --%>
<%= s2 %>

<%! String s2 = new String("How are you ?"); %>
```

Hello ! Hello ! Hello !
How are you ?

Directives

- Dans une page JSP, des directives existent pour répondre à vos besoins
- Trois types de directives
  - **Page** directives
    - *<%@page ... %>*
  - **Inclusion** directives
    - *<%@include ... %>*
  - **Taglib** directives
    - *<%@taglib ... %>*

Page directive

- Définir certaines propriétés de la page
- Les directives de page peuvent avoir plusieurs attributs
  - Exemples:

Property	Syntax
import	<%@ page import="java.util.List" %>
session	<%@ page session="true" %>
contentType	<%@ page contentType="text/html" %>
isELIgnored	<%@ page isELIgnored="true" %>

Include directive

- Inclure le contenu d'une page HTML ou JSP
  - Se produit pendant la traduction
  - Pas une opération runtime !
- Un attribut obligatoire : **file**

- Exemple:

```
<%@ include file="template/header.html" %>
```

Taglib directive

- Référencer une bibliothèque de balises
  - Pour rendre des actions personnalisées disponibles dans la page JSP

    <%@taglib prefix="tags" "uri=http://cci.com/taglibs/tags" %>

- Nous en verrons plus sur les taglibs plus tard ☺

Objets implicites – 1/2

- Sur une JSP vous avez un accès aux objets implicites
- Ci-dessous les plus utiles

Variable name	Type	Description
out	JspWriter	Comme le PrintWriter que vous obtenez sur un ServletResponse. Ecrire quelque chose sur la page
request	HttpServletRequest	La servlet request

Objets implicites – 2/2

- Sur une JSP vous avez un accès aux objets implicites
- Ci-dessous les plus utiles

Variable name	Type	Description
response	HttpServletResponse	La servlet response
session	HttpSession	La session utilisateur en cours
application	ServletContext	Contexte du servlet de la page JSP
config	ServletConfig	La configuration des servlets

Objets implicites – Exemple

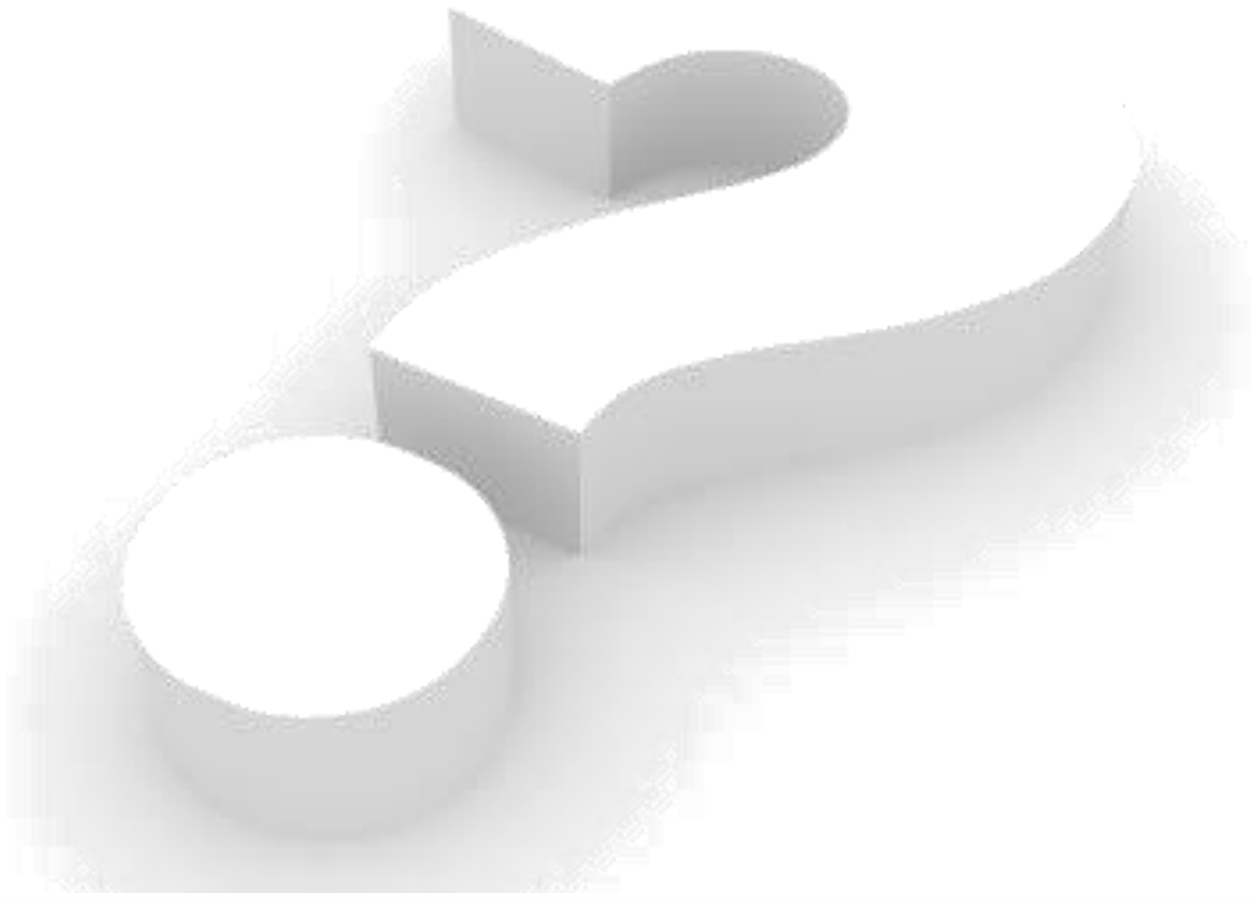
- Un formulaire sur une page **page1.jsp**:

```
<form method="post" action="page2.jsp">
      <input type="text" name="username" />
      <input type="submit" value="OK" />
</form>
```

  - Sur la **page2.jsp** nous affichons la valeur soumise:

```
<% out.println(request.getParameter("username")); %>
<%-- Or more simplier ... --%>
<%= request.getParameter("username") %>
```

Questions ?



Exercices (1/5)

- Il est maintenant temps d'utiliser JSP dans notre projet !

- Créer une page JSP nommée : **listProduct.jsp**
  - Afficher tous les produits stockés en mémoire

- Créer une page JSP nommée : **showProduct.jsp**
  - Afficher les détails du produit avec l'identifiant dans l'URL

Exercices (2/5)

- Créer une nouvelle page JSP
  - Avec le nom : **addProduct.jsp**

- Avec un formulaire HTML contenant des champs pour définir un nouveau produit

- Mettez-le dans un dossier nommé **auth**
  - Pour exécuter le filtre…

Exercices (3/5)

- Créer un HttpServlet
  - Nommez-le **AddProductServlet**
  - Liez-le à **/auth/addProduct** url-pattern
  - Remplacer la méthode **doPost(…)**
    - Récupérer les paramètres du formulaire
    - Créer un nouvel objet **SupProduct**
    - Ajoutez-le en mémoire avec le DAO
    - Redirigez l'utilisateur vers le **ShowProductServlet** pour afficher le nouveau produit

Exercices (4/5)

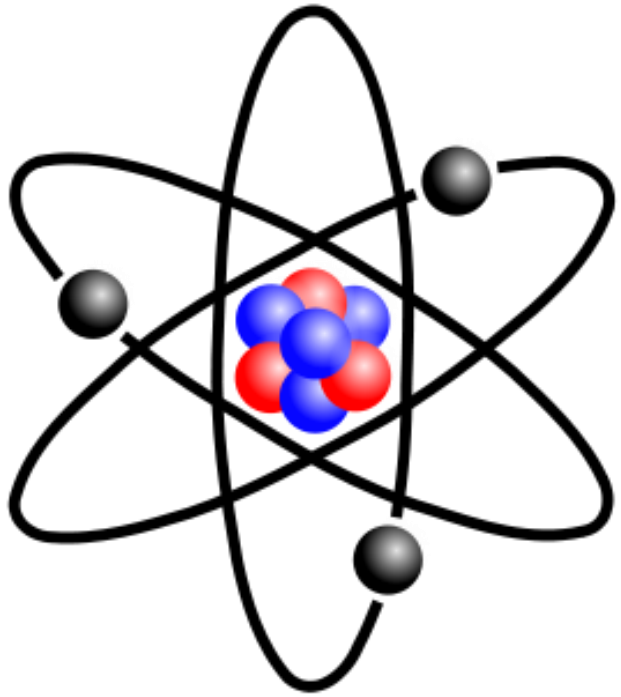
- Créer un HttpServlet
  - Nommez-le **LogoutServlet**
  - Liez-le à **/logout** url-pattern
  - Remplacer la méthode **doGet(…)**
    - Supprimer l'attribut de session "**username**"
    - Rediriger l'utilisateur vers la page de connexion
- Remplacer la méthode **doPost(…)**
  - Appeler la méthode doGet(…) pour faire la même chose pour les méthodes GET ou POST

Exercices (5/5)

- Créez une page jsp nommée : header.jsp
  - Avec des liens HTML vers :
    - Page de liste des produits
    - Page d’ajout d’un produit (si l'utilisateur est connecté)
    - Logout ou Login Servlet (en fonction de si l'utilisateur est connecté ou non)
  - Incluez-le dans toutes vos pages JSP
- Créez une page jsp nommée : footer.jsp
  - Avec quelques informations de votre choix
  - Incluez-le dans toutes vos pages JSP

Java Server Pages

ÉLÉMENTS D'ACTION



Introduction

- Il offre
  - Inclusions de fichiers
  - Redirection
  - Instanciation et manipulation d'objets
  - …

- Syntaxe : **<jsp:action … >**, où action sera remplacée par l'action que vous décidez d'utiliser

useBean

- Permet l'utilisation de JavaBeans dans une page JSP

- Les JavaBeans sont des classes suivant ces caractéristiques
  - Avoir un constructeur public sans arguments
  - N'ont pas d'attributs publics
  - Avoir un getter et un setter pour chaque attribut
  - Implémenter sérialisable

useBean

- Comment utiliser cette action ?
  <jsp:useBean … />
- Vous pouvez transmettre des attributs à ce balisage
  - **id** : nom de l'instance du bean
  - **class** : nom de la classe Java du JavaBean à utiliser
  - **scope** : page | request | session | application
  - **beanName** : instancie un bean à partir d'une classe ou d'un modèle sérialisable
  - **type** : changez le type de bean s'il existe déjà

useBean – Exemple

- Ces extraits sont équivalents :

```
<jsp:useBean id="student"
                  class="com.cci.sun.beans.Student"
                  scope="session" />
```

```
<%
   Student student = null;
   student = (Student) session.getAttribute("student");

   if(student == null) {
      student = new Student();
   }
%>
```

setProperty

- Autoriser à définir une propriété d'un JavaBean
- Doit **toujours** être appelé **après** un <jsp:useBean … />

- Trois façons de définir des propriétés à l'aide d'attributs

1. property="propName" value="Terry"
2. property="propName" param="paramName"
3. property="*"

setProperty example

```
<jsp:useBean id="student"
   class="com.cci.sun.beans.Student"
   scope="session" />

<jsp:setProperty
   name="student"
   property="idBooster"
   value="300" />

<%-- Set the firstName value with the value of a
request parameter named firstName --%>
<jsp:setProperty
   name="student"
   property="firstName"
   param="firstName" />

<%-- Set all the property values of the bean with
request parameters with the same name --%>
<jsp:setProperty name="student" property="*" />
```

getProperty

- Permettre d'obtenir une propriété d'un JavaBean
- Doit **toujours** être appelé **après** un <jsp:useBean … />

```
<jsp:useBean id="student"
                  class="com.cci.sun.beans.Student"
                  scope="session" />

<%-- Displays the value of the firstName property --%>
<jsp:getProperty name="student" property="firstName" />
```

include

- Permet d'inclure le résultat d'une autre ressource dans celle en cours
  - Opération d'exécution (runtime)
  - Similaire à RequestDispatcher.include(…)
- Utilisez l'attribut page pour indiquer la ressource à inclure

```
<jsp:include page="aPage.jsp" />
```

forward

- Permet le transfert de l'utilisateur vers une autre page
- Utilisez l'attribut de page pour spécifier la destination du transfert

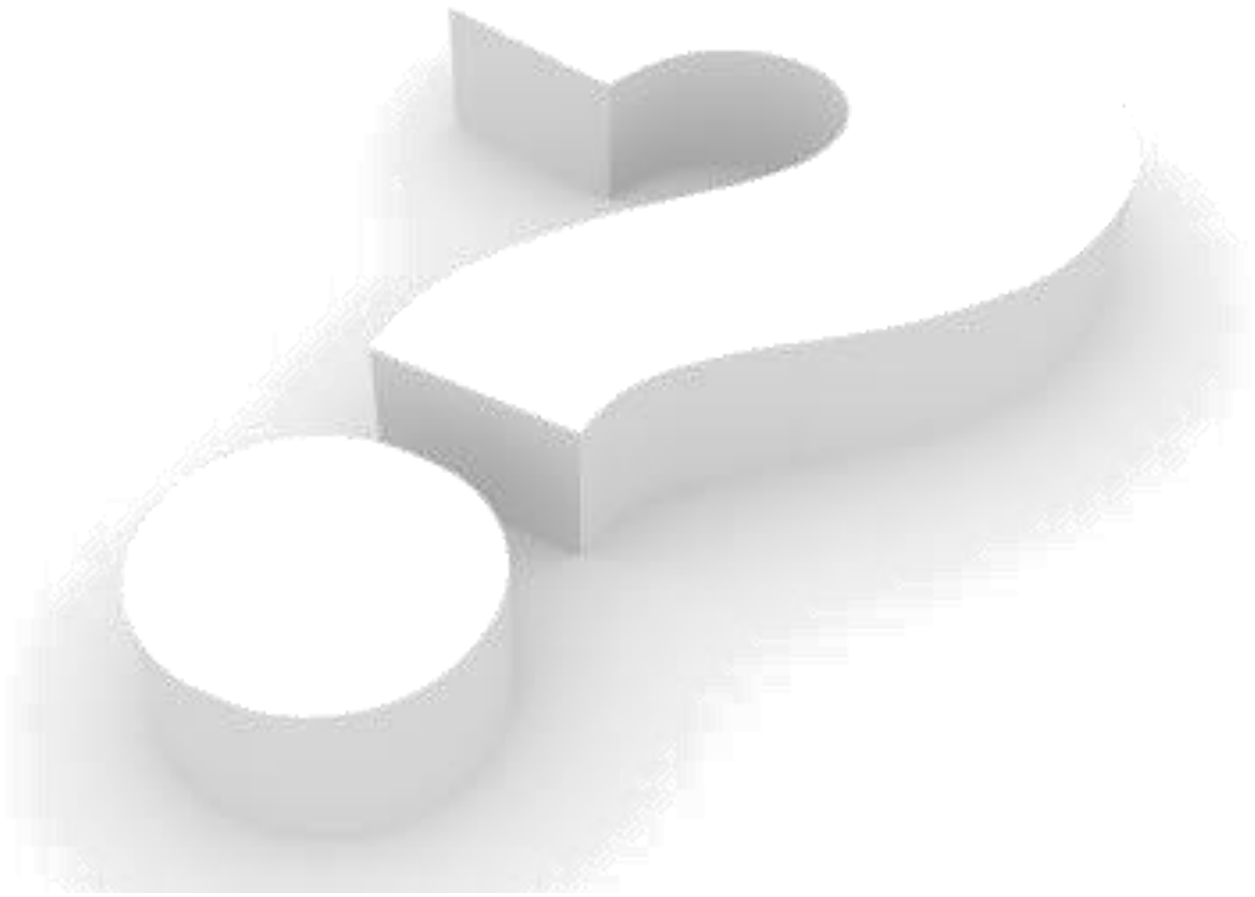
```
<jsp:forward page="aPage.jsp" />
```

param

- Permet l'envoi de paramètres à
  - <jsp:include … /> ou <jsp:forward … />
- Utiliser l'attribut
  - **name** pour indiquer le nom du paramètre
  - **value** pour spécifier la valeur du paramètre

```
<jsp:forward page="aPage.jsp">
      <jsp:param name="idBooster" value="300" />
      <jsp:param name="firstName" value="Goku" />
      <jsp:param name="lastName" value="SON" />
</jsp:forward>
```

Questions ?



Java Server Pages

EXPRESSION LANGUAGE

Présentation

- Langage permettant d'utiliser des objets Java sur JSP
  - Ne remplace pas les scriptlets
  - Mais JSTL le fait…
- Syntaxe succincte et robuste : **${expression}**
- Exemple : **${sessionScope.user.name}**

EL Literals

- Cinq types de littéral EL
  - Boolean : **${true}**
  - Integer : **${12345}**
  - Floating point : **${1.16876}**
  - Strings : **${"Hello"}** or **${'World'}**
  - Null : **${null}**

EL opérateurs

- Arithmétique:

Opération	Opérateur
Addition	**+**
Soustraction	**-**
Multiplication	*****
Division	**/** ou **div**
Modulo	**%** ou **mod**

EL opérateurs

- Relationnel :

Opération	Opérateur
Supérieur	**>** ou **gt**
Supérieur ou égal	**>=** ou **ge**
Egal	**==** ou **eq**
Inférieur	**<** ou **lt**
Inférieur ou égal	**<=** ou **le**
Non Egal	**!=** ou **ne**

EL opérateurs

- Logique :

Opération	Opérateur
Logique « and »	**&&** ou **and**
Logique « or »	**||** ou **or**
Logique « not »	**!** ou **not**

EL opérateurs

- empty: renvoie true si null ou vide
  - Syntaxe :

${empty obj}

- Renvoie true si :
  - obj est une chaîne vide
  - obj est un tableau vide
  - obj est une carte ou une collection vide

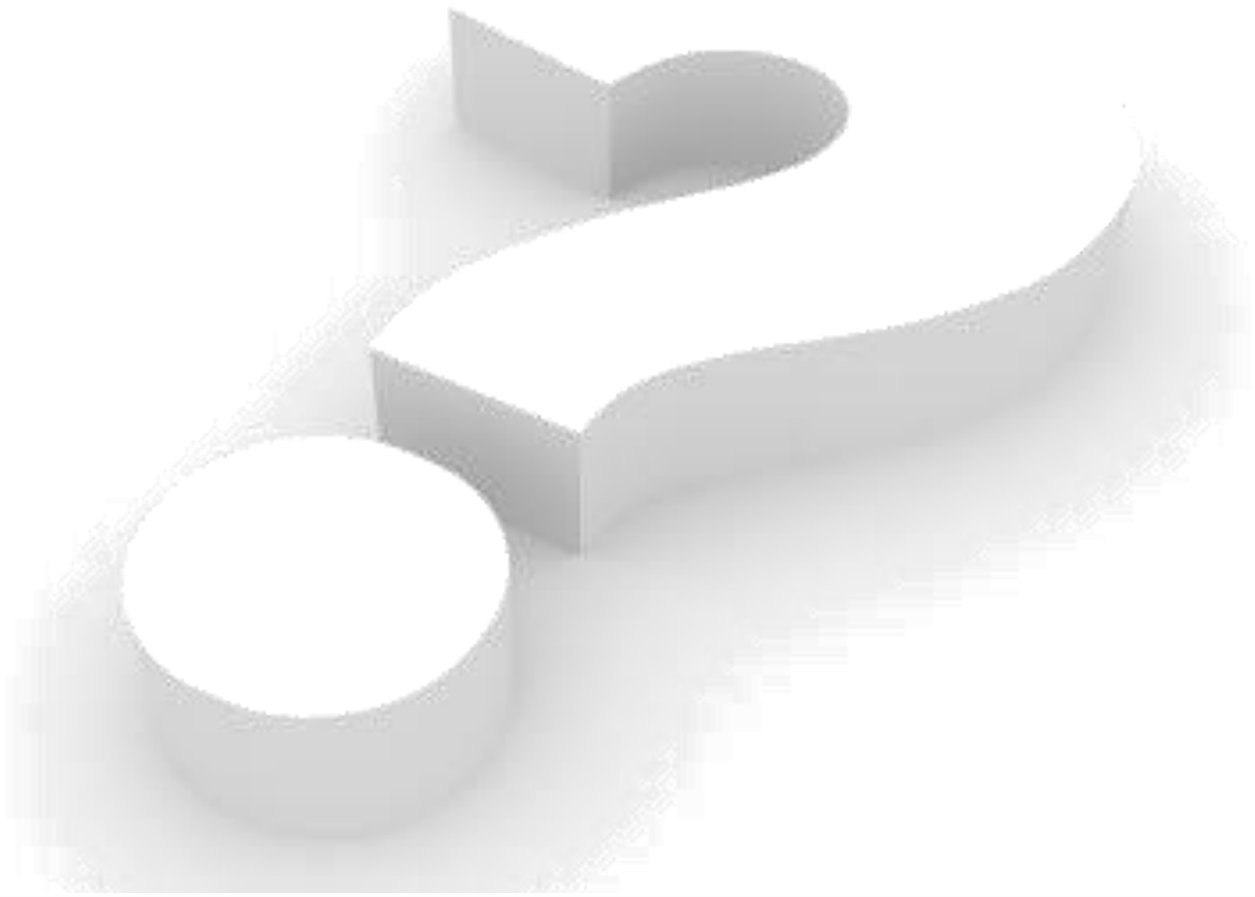
EL opérateurs

- Tout attribut dans n'importe quelle portée peut être affiché avec les EL
- Si un attribut appelé "title" existe
  - Vous pouvez le récupérer avec cette expression : **${title}**
- Si un attribut appelé person existe et s'il s'agit d'un objet complexe
  - Pour appeler la méthode getName() dessus, vous pouvez utiliser
    - **${person.name}**
    - **${person["name"]}**
    - **${person['name']}**

Objets implicites

- Certains objets implicites existent
  - **pageScope** : pour accéder aux attributs de la portée de la page
  - **requestScope** : pour accéder aux attributs de la portée de la requête
  - **sessionScope** : pour accéder aux attributs de la portée de la session
  - **applicationScope** : pour accéder aux attributs de la portée de l'application
  - **param** : pour accéder aux paramètres de la requête
  - **cookie** : pour accéder aux cookies
  - …

Questions ?

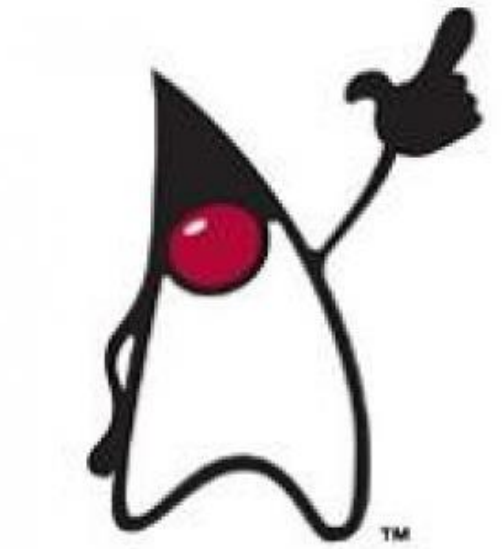


Exercices

- Créer un HttpServlet
  - Nommez-le **RemoveProductServlet**
  - Liez-le à **/auth/removeProduct** (url-pattern)
  - Remplacer la méthode **doPost(…)**
    - Récupérer l'identifiant (id) du produit dans les paramètres de la requête
    - Supprimer l'objet correspondant en mémoire
    - Rediriger l'utilisateur vers la liste des produits
- Mettez à jour votre page **listProduct.jsp**
  - Ajouter un lien vers le **RemoveProductServlet** pour chaque produit

Java Server Pages

TAGLIBS



Présentation

- Étiquettes personnalisées
- Comme les JavaBeans, fournit une séparation de code Java et JSP
- Simplifie la gestion d'une application Web

Comment ça marche

- On a un gestionnaire de balises pour chaque ensemble de balises
  - Classe Java implémentant javax.servlet.jsp.tagext.Tag
- Un fichier descripteur de bibliothèque de balises (TLD)
  - Lien entre les pages JSP et le gestionnaire de balises
  - Décrit un ensemble de balises
- Le chemin vers le TLD peut être spécifié dans le descripteur de déploiement (facultatif)

Déploiement

- Code plus optimisé
  - Modifier le chemin uniquement dans le fichier **web.xml**

```
<jsp-config>
      <taglib>
            <taglib-uri>uriName</taglib-uri>
            <taglib-location>TLDPath</taglib-location>
      </taglib>
</jsp-config>
```

Déclaration et utilisation

- Déclarez d'abord votre taglib
  - Grâce à
    <%@taglib uri="uriLocation" prefix="aPrefix" %>
  - Pour l'utiliser
    <aPrefix:tagName attribute="blue" … />

Déclaration et utilisation

- Taglib inclusion

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

<html>
      <head><title>My Page</title></head>
      <body>
            <c:out value="Hello !" />
      </body>
</html>
```

- Affiche “Hello !”

Bibliothèque de balises standard Java

- Aussi connu sous le nom de JSTL
- Spécifications Sun
- 4 bibliothèques
  - **Core** : fonctions de base
  - **XML** : traitements XML
  - **Format** : Internationalisation
  - **Database**: requêtes SQL
- Évitez l'utilisation de scriptlets !

JSTL: Core

- Certaines balises disponibles dans la bibliothèque principale
  - **out** : affiche dans le flux de sortie
  - **set** : instancier ou modifier une variable
  - **if** : définir un bloc conditionnel
  - **catch** : intercepter les exceptions
  - …
- Informations Taglib :
  - URI : http://java.sun.com/jsp/jstl/core
  - Préfixe généralement : c

JSTL Core example

```
<%@taglib
   uri="http://java.sun.com/jsp/jstl/core" prefix="c"
   %>
...
<table>
   <c:forEach items="${tickets}" var="t">
      <tr>
         <td><c:out value="${t.name}" /></td>
         <td><c:out value="${t.content}" /></td>
         <td>
            <c:if test="${t.editable}">
               <a href="edit?id=${t.id}">Edit</a>
            </c:if>
         </td>
      </tr>
   </c:forEach>
</table>
...
```

JSTL: Format

- Bibliothèque de formats de balises disponibles
  - **setBundle** : spécifiez la ResourceBundle à utiliser
  - **message** : imprime un message associé à une clé dans le ResourceBundle
  - **param** : utilisé pour le message dynamique
- Rappellez-vous
  - Un ResourceBundle est composé de fichiers **.properties**. Chaque fichier est une traduction pour une langue spécifique

JSTL: Format exemple

- Considérant le fichier Msg_en.properties au chemin :
  - com/cci/sun/supcommerce/lang/Msg_en.properties
  - Ce fichier contient :

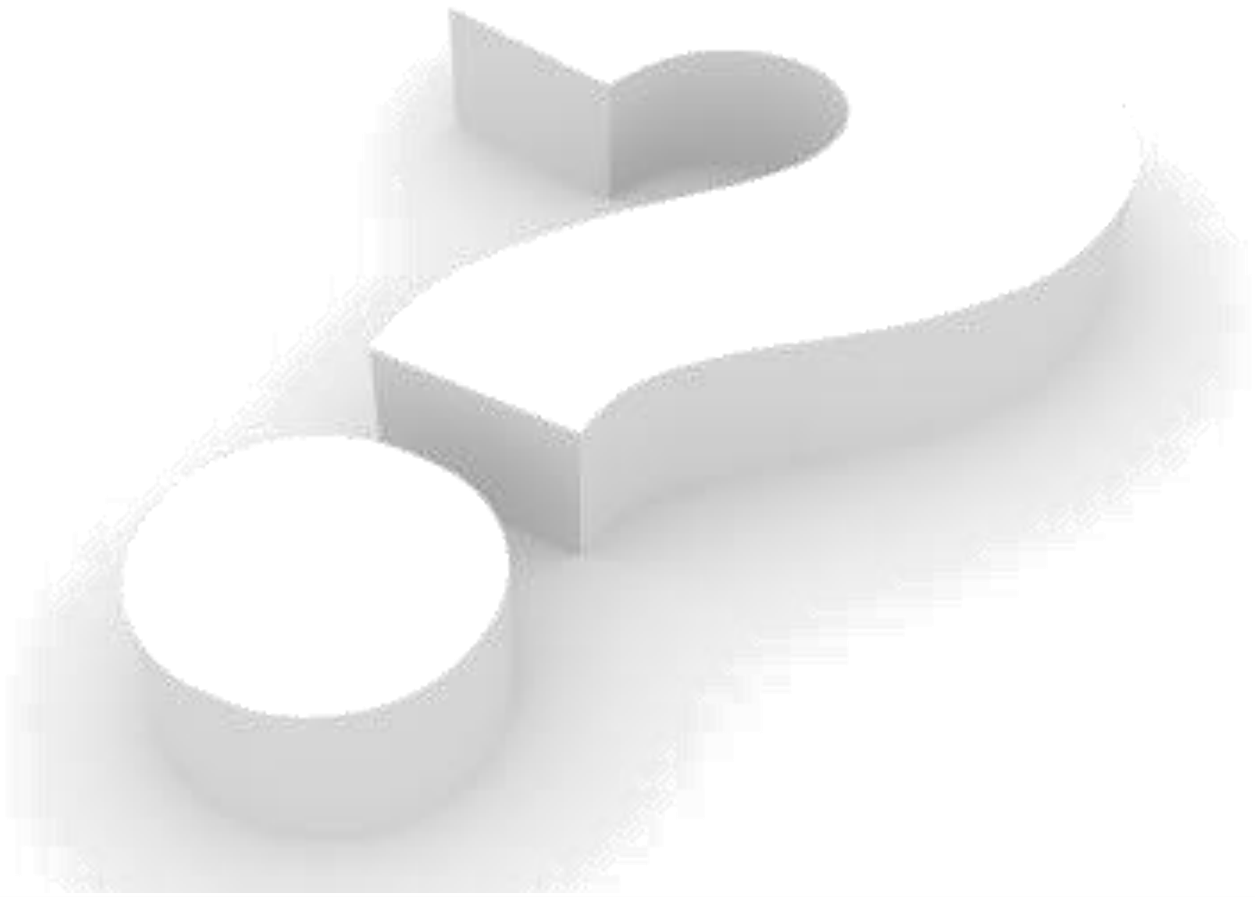
    message.hello=Bonjour le monde !

Comment utiliser les paramètres régionaux :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
   prefix="fmt" %>
<fmt:setBundle
   basename="com.cci.sun.supcommerce.lang.Msg" />
<fmt:message key="message.hello" />
```

- Affiche “Hello !”

Questions ?



Exercices

- Téléchargez les bibliothèques JSTL et placez-les dans le répertoire lib de votre projet Web

- Refactorisez votre page **listProduct.jsp** pour utiliser JSTL au lieu des itérations Java
  - Utilisez Expression Language au lieu d'éléments de script lorsque vous le pouvez

Java Server Pages

SERVLETS & JSP

Best practices

- N'abusez pas du code Java dans les pages HTML
  - Évitez le code spaghettis
  - Facile à lire = > Facile à entretenir

- Utiliser des balises personnalisées
  - Plus facile à lire pour les développeurs de contenu HTML

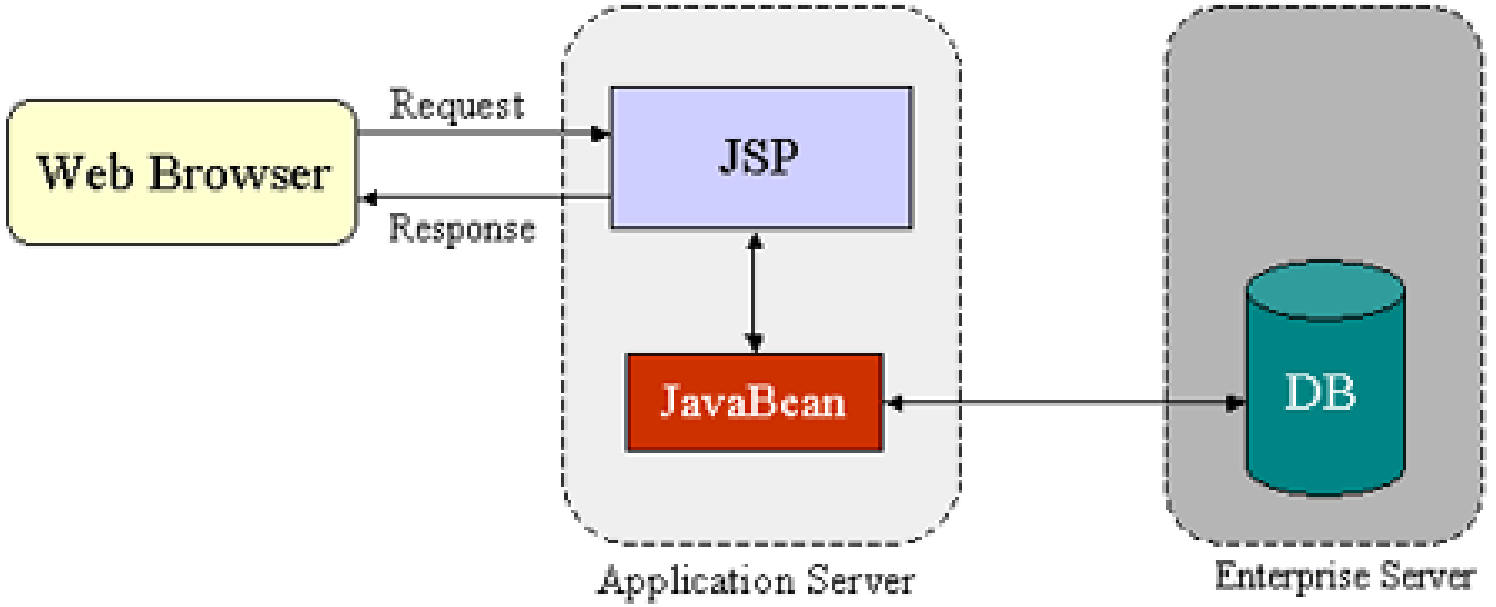
Best practices

- Utiliser le langage d'expression (EL)
  - Syntaxe simplifiée

- Redirigez l'utilisateur au lieu de transmettre sa demande lorsque vous le pouvez
  - URL pouvant être mise en signet
  - Mieux pour les boutons **Précédent** et **Suivant** du navigateur

JSP Model 1 Architecture

- L'une des deux approches pour utiliser JSP
- La page JSP est responsable du traitement des demandes

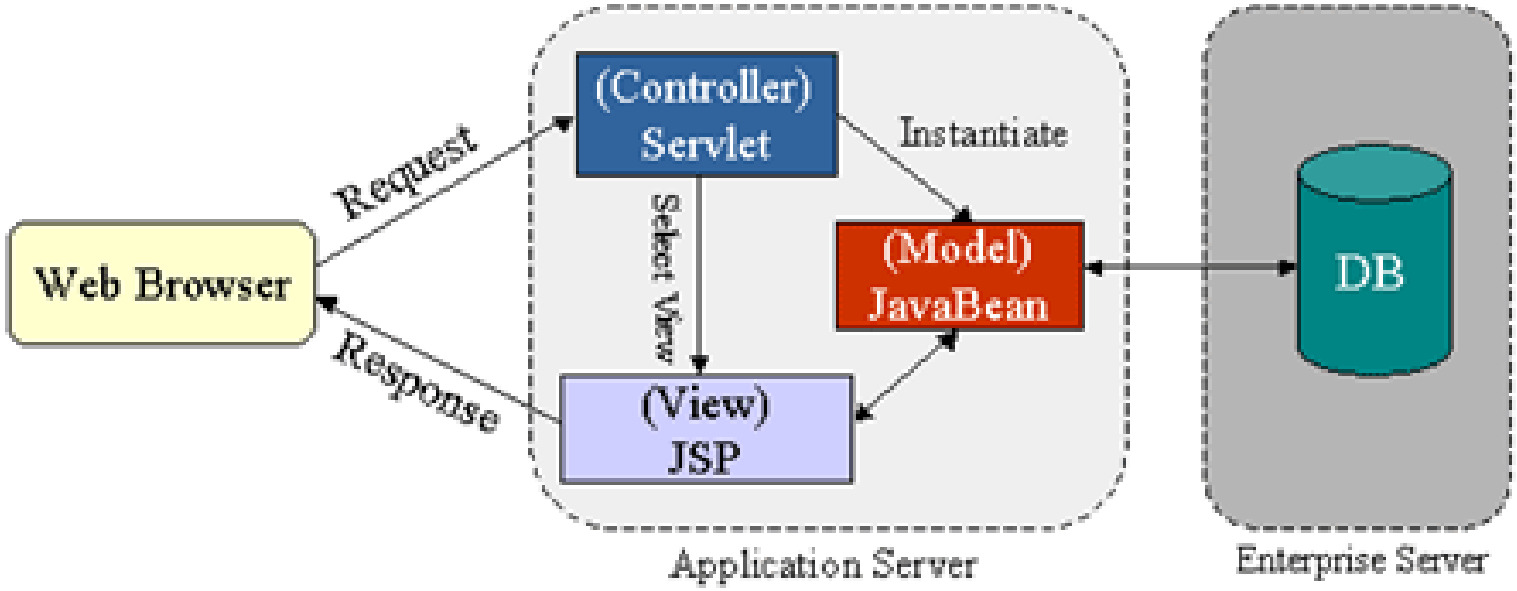


Best practices

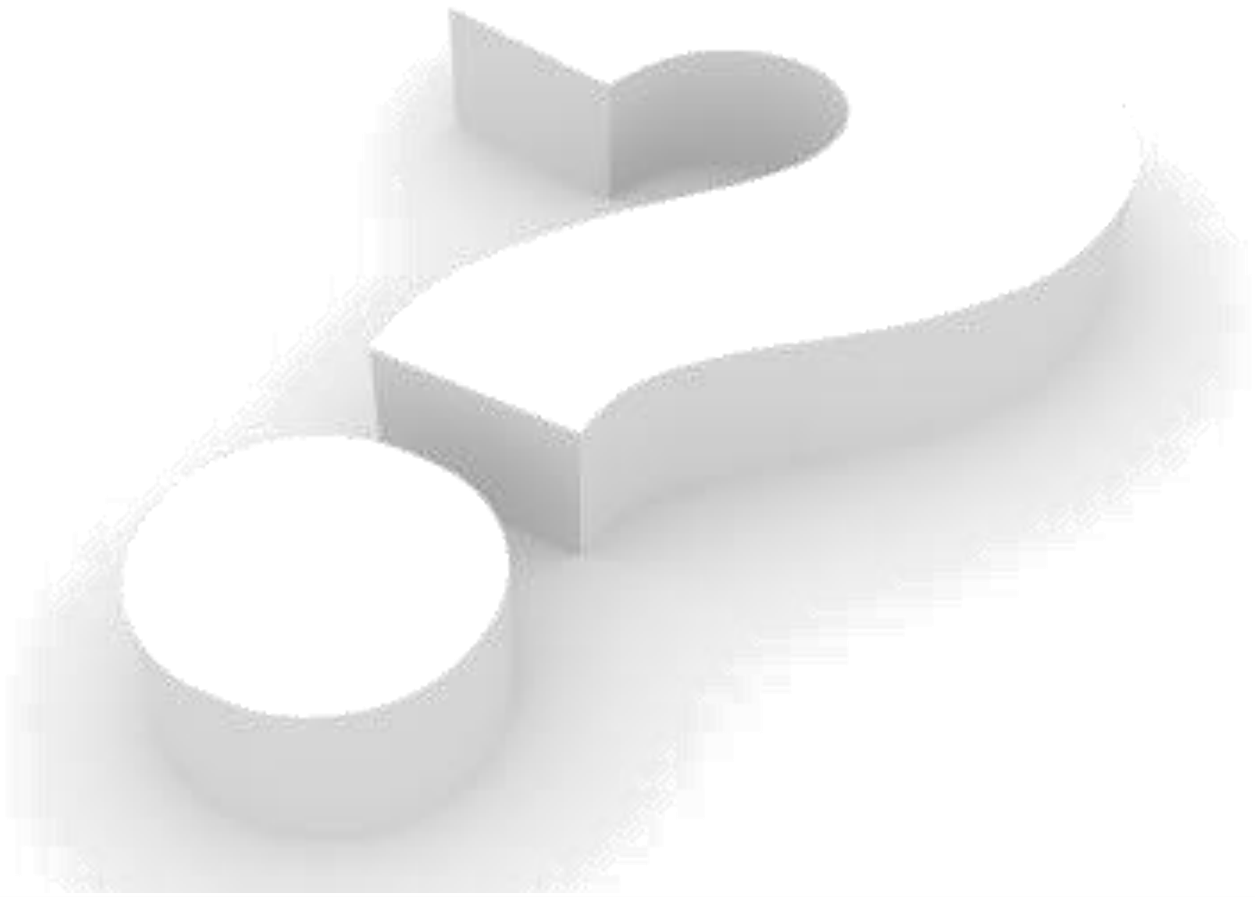
- Les JSP et les servlets fonctionnent de manière complémentaire
  - Pages JSP utilisées pour générer une réponse HTML
  - Servlet pour le traitement des tâches
- Servlet agit en tant que contrôleur
  - Traiter la requête, créer des beans, …
  - Décider à quelle page JSP transférer la demande
- JSP agit comme une vue
  - Récupère les objets créés par le contrôleur
  - Utilisez-les pour construire la réponse HTML

JSP Model 2 Architecture

- Model View Controller (MVC) design pattern !



Questions ?



Exercices (1/2)

- Vous comprenez maintenant comment Servlets et JSP sont complémentaires :
  - Refactoriser **ListProductServlet** et **listProduct.jsp** pour implémenter l'architecture JSP modèle 2
    - Mettre la liste de produits dans les attributs de la requête
    - Transférer la demande au JSP
    - Récupérer la liste dans la page JSP
  - Refactoriser **AddProductServlet** et **addProduct.jsp**
    - Remplacer la méthode **doGet(…)**
    - Transférer la demande à la page JSP

Exercices (2/2)

- Vous comprenez maintenant comment Servlets et JSP sont complémentaires :
  - Refactoriser **ShowProductServlet** et **showProduct.jsp**
  - Refactoriser **LoginServlet** et **login.jsp**
  - Remplacez tous les éléments de script de votre page JSP par JSTL + EL !

Fin

Merci de votre attention