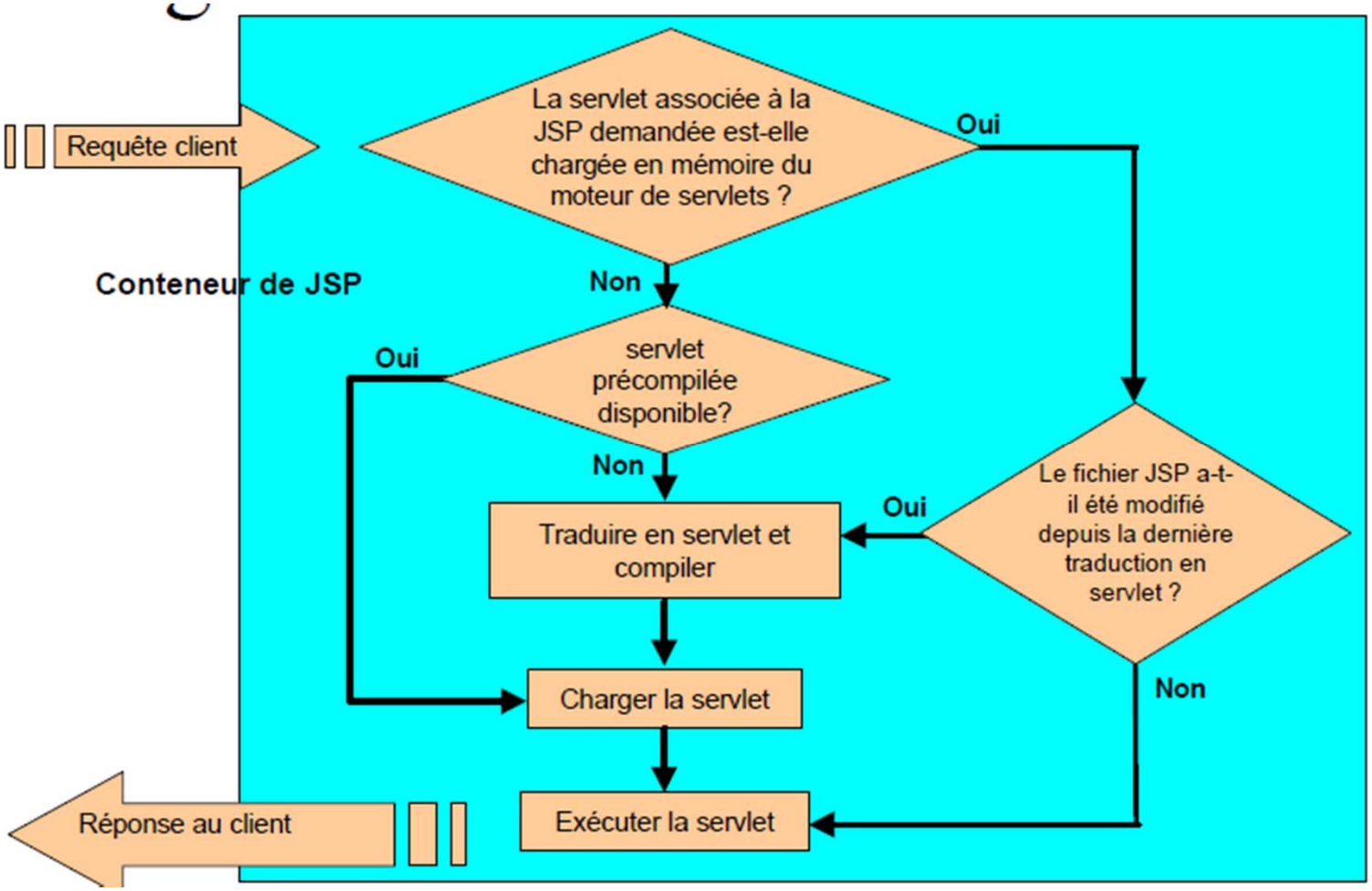
Technologie JSP version 2.1

Algorithme d'exécution d'une servlet



Composants d'une page JSP

- Deux styles d'écriture de page jsp
  - Style standard
  - Style XML
- Types de blocs dans une page JSP
  - Directives
  - Scripts
  - Actions
- Commentaires: une page jsp peut contenir des commentaires html ou des commentaires jsp
  - <!– commentaire html-->
  - <%-- commentaire- jsp -%>

Directives

- Une directive fournit au conteneur des informations sur la manière dont il doit traiter la page jsp.
- Directives
  - page
  - include
  - tag
  - taglib

Directive page

- Styles
  - Style standard: <% @page « attributs »%>
  - Style XML: <jsp:directive.page « attributs »/>
- Attributs
  - import : import de packages
  - session : précise si la page prend en considération les variables session ou non (par défaut =false)
  - errorPage: url de la page à afficher une exception est levée.
  - isErrorPage: précise si c'est une page d'erreur (par défaut=false)
  - contentType
  - pageEncoding
  - extends: définit la classe parente qui doit être de type javax.servlet.GenericServlet
- Exemple:
  - Style standard:
    <%@ page errorPage="/WEB-INF/errorPage.jsp"
    import="java.util.Iterator,jsp.exemple1.Categories" %>
  - Style xml:
    <jsp:directive.page errorPage="/WEB-INF/errorPage.jsp"
    import="java.util.Iterator,jsp.exemple1.Categories" />

Directive include

- Inclure un fichier
- Style standard:
  - <%@ include file=" *cheminrelatif " %>*
- Style xml:
  - <jsp:directive.include file="*cheminrelatif" />*
- Exemple
  - <%@ include file="/WEB-INF/footer.jspf" %>

Directive tag

- Similaire à la directive page mais s'applique aux fichiers de tags.
- Syntaxe:
  - Style standard

    <%@ tag attributs%>

  - Style xml

    <jsp:directive.tag *attributs />*

Les scripts

- 3 types de scripts jsp
  - Déclarations
  - Scriptlets
  - Expressions

Déclarations

- Ce type de bloc est utilisé pour déclarer des variables (qui peuvent aussi être initialisées) et pour définir des méthodes.
- Syntaxe:
  - Style standard: <%! *déclarations %>*
  - Style XML: <jsp:declaration>*déclarations</jsp:declaration>.*
- Exemple:
  - <%! Categories themes = new Categories(); %>
  - <jsp:declaration>Categories themes = new Categories(); *</jsp:declaration>*
- *Exemple 2:*

```
<%!
   int entier() {
      return (int) (Math.random() * 100);
   }
      %>
```

- Remarque:
  - Une variable ou une méthode déclarée dans ce bloc devient un membre d'instance dans la classe qui implémente la page jsp.
  - Les déclarations Seront insérées comme des membres de la servlet
  - Ce type de bloc permet de définir des méthodes ou des données membres

Scriptlets

- Ce type de bloc contient des instructions Java
- Syntaxe:
  - Style standard: <% *code %>*
  - Style XML: <jsp:scriptlet>*code </jsp:scriptlet>*

- Exemple 1:

```
<% for (int i = 1; i < 10; i++) {
%>
Bonjour, Monde!
<%
}
%>
```

- Exemple 2

```
String[] langages = {"Java", "C++", "Smalltalk", "Simula"};
out.println("<h3>Principaux langages orientés objets : </h3>");
for (int i=0; i < langages.length; i++) {
out.println("<p>" + langages[i] + "</p>");
}
```

- Remarque
  - Les variables déclarées dans ce type de bloc ont une portée locale.
  - Les méthodes ne peuvent pas être définies dans ce type de bloc.
  - inséré dans _jspservice() de la servlet, donc peut utiliser out, request, response, etc.

Expressions

- Une expression est utilisée pour généré une valeur
- Syntaxe:
  - Style standard: <%= *expression %>*
  - Style XML: <jsp:expression>*expression</jsp:expression>*
- Exemple:
  - Style standard: <%= remplacerUnderscore(categorie) %>
  - Style XML: <jsp:expression> remplacerUnderscore(categorie) *</jsp:expression>*

Les actions

- Les actions sont des balises qui sont utilisées dans une page jsp pour exécuter certaines actions
- La spécification jsp définit un certain nombre d'actions dites actions standards.

Actions standards

- <jsp:useBean>
- <jsp:setProperty>
- <jsp:getProperty>
- <jsp:param>
- <jsp:include>
- <jsp:forward>
- <jsp:plugin>
- <jsp:params>

Action useBean

- Localisation ou instanciation d'un JavaBean
- Un JavaBean est une classe qui respecte les contraintes suivantes
  - La classe ne doit pas avoir des constructeurs avec argument
  - Pour chaque attribut qui doit être accessible au client du JavaBean on doit créer les accesseurs get et ou set.
  - Le nom des accesseurs possède le format suivant:
    - public type getNomAttribut { return nomAttribut;}
    - public Boolean isNomAttribut { return nomAttribut;}
    - public void setNomAttribut(type val){nomAttribut=val;}

Action useBean

- Syntaxe: <jsp:useBean « attributs »>
- Attributs
  - id:nom de la variable qui référence l'instance du bean
  - scope: définit la portée du bean, valeurs possibles: page (par défaut), request, session ou application
  - class: nom complètement qualifié de la classe du bean
  - type: le type à utiliser pour l'instance qui référence le bean, valeurs possibles: la classe du bean, une classe parente, une interface implémentée par la classe du bean ou une classe parente.
- Remarque:
  - Cette action doit précéder les actions setProperty et getProperty.

<jsp:setProperty>

- Définit les propriétés d’ un JavaBean
- Attributs:
  - name: id du bean
  - property: nom de la propriété du bean à définir (ou *)
  - param: nom du paramètre de l’objet request dont la valeur sera affectée à la propriété (à utiliser dans le cas où le nom de la propriété et celui du paramètre sont différents).
  - value: valeur à affecter dans la propriété.

<jsp:getProperty>

- Récupère la valeur d'une propriété
- Attributs
  - name : id du bean
  - property: nom de la propriété

Action forward

<jsp:forward page="URL">

<jsp:param name=« par1" value="val"/>

</jsp:forward>

- Remarque
  - L’action forward doit être appelée avant l’envoi vers OutPutSream

Action include

- Exécution d’une page jsp

  <jsp:include page="URL" >

  <jsp:param name="param1 " value=« val"/>

  </jsp:include>

- Remarque
  - Le fichier appelé ne doit pas modifié l’entête http ou définir des cookies
  - La page appelée peut récupérer les paramètres à l’aide des méthodes getParameter() getParameterValues() de l’objet request

Objets implicites

- request
- response
- out
- session
- config
- exception
- Application
- page
- pageContext

Objet request

- Encapsule une requête http
- Méthodes

  Récupération des paramètres

  - String request.getParameter(String nom)
  - String[] request.getParameterValues(String nom)
  - Enumeration request.getParameterNames()
    - Une énumération possède les méthodes suivantes:
      - » e.hasMoreElements();
      - » e.nextElement();
  - Map getParameterMap()
    - Un objet Map possède les éléments suivants:
      - » P.get(clé)

  Récupération des cookies

  - Cookie [] request.getCookies()
- Remarque:
  - Request est une instance de javax.servlet.HttpServletRequest
  - Un objet request a la portée request

Objet response

- Encapsule une réponse envoyée à un client web
- Méthodes
  - public void addHeader(String nom, String valeur)
  - public void addCookie(Cookie cookie)
  - public void sendRedirect(String url)
- Remarque
  - response est une instance de javax.servlet.HttpServletResponse
  - Un objet respnose à la portée page

Objet out

- Exemple:

```
<%
Iterator categories = themes.getAllCategories();
while (cate00000000000gories.hasNext()) {
String categorie = (String)categories.next();
out.println("<p><a href=\"" + replaceUnderscore(categorie) + "\">"
    +
categorie+ "</a></p>");
}
%>
```

- Remarque
  - out est une instance de la classe javax.jsp.JspWriter
  - La portée de l'objet out est page

Objet session

- Méthodes
  - Object setAttribute(String nom, Object valeur)
  - Object getAttribute(String nom)
  - Enumeration getAttributeNames()
  - void removeAttribute(String nom)
- Remarque
  - session est une instance de la classe javax.servlet.http.HttpSession
  - L'objet session a la portée session
  - Les sessions ne sont pas thread -safe

Objet config

- L'objet config est utilisé pour récupérer les paramètres d'initialisation d'une page jsp définis dans le DD web.xml

```
<servlet>
…
<param-name>nom</param-name>
<param-value>valeur</param-value>
</init-param>
</servlet>
```

- Méthodes
  - config.getInitParameter(String nom)
  - Enumeration config.getInitParameterName()
  - String config.getServletName()
- Remarque
  - config a la portée page
  - config est une instance de javax.servlet.ServletConfig

Objets exception et application

Objet exception

- Accessible uniquement dans les pages d'erreurs.
- L'objet exception est une référence à l'objet de type java.lang.Throwable créé par le serveur suite à une erreur

Objet application

Utilisé pour récupérer des paramètres

- application.getInitParameter(String name)
- Enumeration application.getInitParameterNames()

```
<web-app>
<context-param>
<param-name>nom</param-name>
<param-value>value</param-value>
</context-param>
</web-app>
```

- Portée application
- application est une instance de javax.servlet.ServletContext

page

pageContext

Initialisation et destruction d'une page JSP

- Initialisation

```
<%!
public void jspInit() {
// Ajouter le code qui doit être exécuté une fois par page et non par
   requête
}
%>
```

- Destruction

```
<%!
public void jspDestroy() {
// libération des ressources
}
%>
```

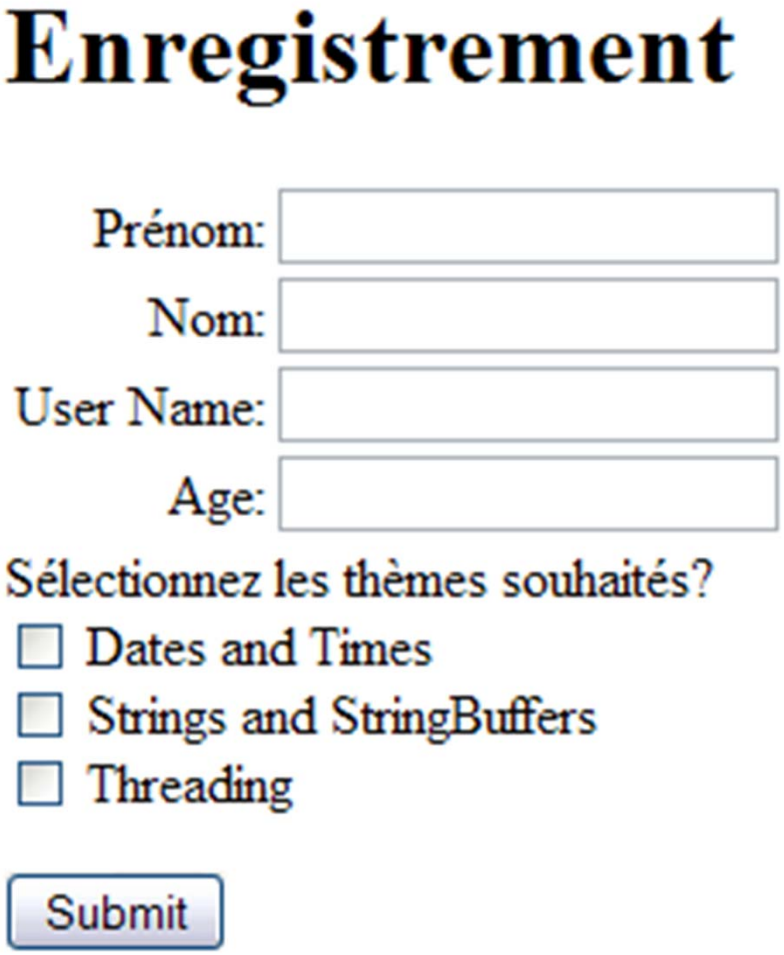
Exemple 2

- Les éléments suivants doivent être ajoutés dans l'application:
  - Un JavaBean User.java
  - Enregistrement .html: page qui contient un formulaire de saisie des information
  - Enregistrement.jsp: page de traitement des informations envoyées par le formulaire

1- Ajouter une classe JavaBean User.java dans le package jsp.exemple2 ayant les attributs suivants:

- prenom, nom, userName et age.

Exemple 2: enregistrement.html



2- Ajouter la page enregistrement.html

- Les informations du formulaire doivent être traitées par la page enregistrement.jsp

Exemple 2: Enregistement.jsp

Informations

Bienvenue nouvel utilisateur, Informations enregistrées:

Prénom:Said.

Nom: El Fehri.

User Name: said.

age: 34.

Thèmes sélectionnés:
Strings et StringBuffers
Threads

Liste des thèmes

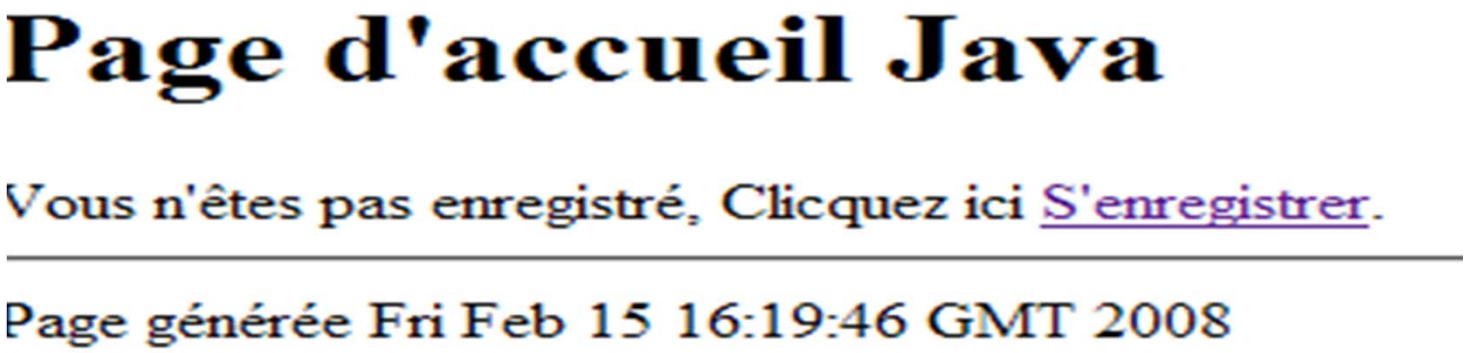
Page générée Fri Feb 15 19:34:19 GMT 2008

Ajouter la page Enregistrement.jsp

- Les informations du formulaire doivent être stockées dans un javaBean (user)
- Le lien **Liste des thèmes** envoie vers la page index.jsp

Exemple 2: modification de la page index.jsp

1. Si l'utilisateur n'est pas encore enregistré alors , on affiche un lien vers la page enregistrement.html, sinon on affiche la lite des thèmes



Exemple 2: Exercices

  - Exercice 1
  - Modifier index.jsp de sorte à ce que uniquement les thèmes sélectionnés par l'utilisateur soit affichées.
  - Si l'utilisateur n'est pas enregistré, il doit être automatiquement redirigé vers la page enregistrement.html.
- Exercice 2
  - Ajouter des pages pour que l'utilisateur puisse ajouter une question ou répondre à une question.
- Exercice 3
  - Ecrire une application web de gestion de QCM

Gestion des erreurs et des exceptions dans le descripteur de déploiement

- Gestion des exceptions java

```
<error-page>
<exception-
   type>java.lang.NumberFormatException</exception-type>
<location>/WEB-INF/FormatNombre.jsp</location>
</error-page>
```

- Erreurs http

```
<error-page>
<error-code>404</error-code>
<location>/WEB-INF/pageintrouvable.jsp</location>
</error-page>
```

Exercices

- Ecrire une page qui affiche une liste de fruits dans des cases à cocher et affiche la sélection de l'utilisateur.
  - Utiliser un javabean et l'objet request
- Calcul du coût de revient d'une voiture:
  - //Définition des propriétés
  - private float coutRoue ;
  - private float coutCarrosserie ;
  - private float coutMoteur ;
  - private float coutVolant ;
  - private float coutRevient ; // En lecture seule