Java côté serveur

Eugen Dedu
Eugen.Dedu@pu-pm.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~dedu/

UFC, IUP 3ème année
Montbéliard
septembre 2004

Plan

- Pages avec contenu mixte
- JSP (Java Server Pages)
  - beans
- Intégration servlets / JSP
- Accès aux bases de données

Pages avec contenu mixte - méthodologie -

- Retour d'un contenu HTML et les objets associés (e.g. images)
- Cas classique (HTML) :
  - plusieurs requêtes à des fichiers distincts : A.html, 1.jpg, 2.png, ...
- Cas des servlets :
  - soit image statique :
    - out.println ("<IMG SRC=... >");
  - soit image dynamique
    - voir transparent suivant

Pages avec contenu mixte - méthodologie -

- Cas des servlets, images dynamiques :
  - la même servlet répond à toutes les requêtes (les images renvoient à la servlet aussi)
  - retourner l'URL courant sans paramètres :
    - StringBuffer req.getRequestURL ()

Pages avec contenu mixte - méthode de la récriture de l'URL -

- http://.../servlets/test : retourne le fichier HTML
  - "<img src=\"" + req.getRequestURL() + "?img=1\" ...>"
- http://.../servlets/test?img=1 : renvoie l'image
  - if (req.getParameter ("img") != null)
  - comme HTML, mais avec des paramètres
- Ne marche pas avec des images dynamiques (dont les données changent dans le temps) :
  - entre la requête HTML et la requête de l'image il passe un certain temps
  - => HTML et image peuvent être incohérents
  - ex. : “L'heure à Paris est [image] et à Londres est [image]”.

Pages avec contenu mixte - méthode de la session -

- Stocker les données (ou les images-mêmes) dans HttpSession
  - exemple : dans les attributs "image1" et "image2"
- http://.../servlets/test : retourne le fichier HTML
- http://.../servlets/test?img=image1 : renvoie l'image stockée dans l'attribut image1
- http://.../servlets/test?img=image2 : renvoie l'image stockée dans l'attribut image2

Pages avec contenu mixte - méthode de la session -

- Marche avec des images dynamiques
- Pratique si les sessions sont de toute façon utilisées dans le site
- On peut utiliser un mécanisme de “session” distinct, manuel, e.g. par récriture de l'URL
  - tâche complexe cependant

Pages avec contenu mixte - méthode du stockage côté client -

- Stocker les données côté client, dans l'URL
  - .../?num=3&v0=Apache|70&v1=IIS|22&v2=Autres|8
  - NB : l'ordre des paramètres est aléatoire !
- Il faut deux méthodes :
  - Image -> StringBuffer
  - String -> Image
- Les données sont transférées deux fois (serveur->client et client->serveur)
- Pratique pour des images à très faible volume de données (e.g. diagrammes)

Pages avec contenu mixte - création d'une image -

- Graphics.fillArc ()
- Utilisation des bibliothèques non inclues en Java SDK
  - création d'images PNG, JPG, ...

Pages avec contenu mixte - envoi de données binaires -

- But : envoi d'images, d'exécutables (bouton Download), ...
- Méthode : .getOutputStream ()
- Utiliser des chemins complets pour les fichiers !
- Fichiers binaires :
  - FileInputStream fis = new FileInputStream ("chemin");
  - while ((int c = fis.read ()) != -1)
  - req.getOutputStream().print ((char)c);
- Fichiers texte :
  - BufferedReader br = new BufferedReader (new FileReader ("/chemin/complet"));

Pages avec contenu mixte - exemple complet -

- Exemple complet d'une servlet envoyant une page HTML avec :
  - du texte
  - une image “paysage.png” par récriture de l'URL

Pages avec contenu mixte - données binaires : en-têtes utiles -

- gzip :
  - String s = req.getHeader ("Accept-Encoding");
    - if (s.indexOf ("gzip") != -1)
  - res.setHeader ("Content-Encoding", "gzip");
  - PrintWriter out = new PrintWriter (new GZIPOutputStream (res.getOutputStream ()));
- Image : en-tête envoyé par le client :
  - Accept: image/png, image/jpg, */*

JSP (Java Server Pages)
- introduction -

- Insérer, “à la PHP”, du code servlet dans des fichiers HTML
- Tomcat = serveur apache + servlets + moteur JSP
- Raison :
  - pas pratique d'envoyer dans des servlets du code HTML avec out.println ()
  - permet d'utiliser les éditeurs HTML
  - sépare la présentation du contenu (traitement)

JSP
- exemple complet -

- HelloWWW.jsp :
  - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
  - <html><head>
  - <title>Hello WWW</title>
  - <meta http-equiv="content-type" ... >
  - </head><%-- commentaire --%>
  - <body>
  - Hello <%= request.getParameter ("title") %>.
  - </body></html>
- Compilation, en servlet :
  - <H2>Titre</H2> -> out.println ("<H2>Titre</H2>");

JSP
- fonctionnement interne -

- Le serveur HTML demande au moteur JSP la page .jsp
  - pas besoin de compilation ou de répertoire spécial
- Si la page .jsp n'est pas compilée (en servlet) ou si elle est plus récente, le moteur JSP :
  - la compile (en la passant à javac)
    - répertoire où les pages compilées sont stockées : /var/cache/tomcat4/
    - à regarder les pages compilées !
  - crée une instance A et exécute A.init()
- Sinon, il utilise l'instance A créée auparavant !
- Le moteur JSP exécute A.service()
- Quand le moteur JSP finit ou temps d'objet dépassé, il exécute A.destroy()

JSP
- types d'éléments -

- Expressions : chaînes affichées dans la page
  - <%= expression %>
- Scriptlets : code servlet
  - <% code %>
- Déclarations : méthodes/variables propres à la page
  - <%! déclaration %>
- Directives : informations globales
  - <%@ directive attribut="valeur" %>
- Autres directives (forward, inclusion, beans)
  - <jsp:XXX ...="..." />

JSP
- expressions <%=...%> -

- Évaluation, conversion, affichage
- Placées à l'intérieur de out.println ();
  - => pas de point-virgule à la fin !
- Exemple :
  - ...<BODY>
  - Le paramètre title de la page est :
  - <%= request.getParameter ("title") %>.
  - </BODY></HTML>

JSP
- scriptlets <%...%> -

- Du code servlet
- Peut accéder aux variables et composants déclarés
- Insérés tels quels dans la méthode _jspService de la servlet (get et post)
- Pas nécessairement des expressions complètes :
  - <% for (int i=0 ; i<4 ; i++){ %>
  - La valeur du compteur est <%= i %>.
  -
  - <% } %>

JSP
- scriptlets : variables prédéfinies -

- request
- response
- session
- application – getServletConfig().getContext ()
- out – type JspWriter (PrintWriter bufferisé)
- config – l'objet ServletConfig
- ...

JSP
- déclarations <%!...%> -

- Inserées telles quelles (variables / méthodes) dans la déclaration de la classe
- Exemples :
  - <%! private int age = 25; %>
  - <%! public void getAge () {...}; %>
- Les champs persistent entre appels !
  - <%! private int nbConn = 0; %>
  - Number of total connexions : <%= ++nbConn %>
  - (voir servlets aussi)

JSP
- déclarations : méthodes -

- out, request etc. sont locales à jspService => pour les utiliser, passer-les en paramètre aux méthodes
  - pourquoi pas champs d'instance ? (partage)
  - <%! void printTitle (HttpServletRequest req, JspWriter o) throws java.io.IOException {
  - o.println (req.getParameter ("title"));
  - } %>
  - ...
  - <% printTitle (request, out); %>
- jspInit, jspDestroy – fonctions vides par défaut appelées par init et destroy de la servlet générée

JSP
- directives <%@...%> -

- Informations globales à la page
- <%@ directive attribute="value" ...%>
- Directives :
  - include – fichiers à inclure littéralement pendant la compilation, à l'intérieur de out.println
    - pour l'inclusion de la sortie d'un JSP pendant l'exécution : voir plus tard
  - page – informations relatives à la page
- Directive include :
  - <%@ include file="..." %>
  - incohérence possible si le fichier à inclure est modifié !

JSP
- directive page -

- import="java.io.*,java.awt.*"
- session="true|false" (true par défaut)
- buffer="none|Nkb" (8 kb par défaut) – la taille du buffer de sortie
- autoFlush="true|false" (false par défaut) – si buffer plein, vidange du buffer ou exception
- isThreadSafe="true|false" (true par défaut)
- info="text" – accessible par Servlet.getServletInfo ()
- contentType="text/html; charset=iso-8859-1"
- errorPage="..." – page appelée si exception non gérée

JSP
- directive page : explications -

- buffer : il est possible d'écrire dans les en-têtes HTML tant qu'aucune donnée n'ait pas été écrite du buffer à la sortie
- errorPage : dans la page gérant l'erreur, la variable “exception” représente l'exception générée
  - exceptions pouvant apparaître : IOException, ServletException et toute exception non vérifiable
  - <%@ isErrorPage="true" %> dans la page gérant l'erreur (nécessaire pour générer le code d'initialisation de la variable exception ci-dessus)

JSP
- accès concurrent : problème -

- Page concurrent.jsp :
  - <%! private int i = 0; %>
  - <%= i++ %>
- i est un champ d'instance
- Une même instance peut être exécutée en parallèle (si plusieurs requêtes client) par le moteur JSP :
  - Servlet serv = new MaServlet (req, res);
  - while (requête)
  - serv.service (...);
- => i est une variable / zone partagée

JSP
- accès concurrent : problème -

- i++ (ou i = i+1) est composée de deux opérations :
  - calcul de i+1
  - affectation du résultat à i
- Les deux threads peuvent se trouver en même temps entre les deux opérations
- => *race condition* à i++ !

JSP
- accès concurrent : solution -

- Utiliser :
  - *synchronized* :
    - <% synchronized (this){
    - i++;
    - } %>
    - (voir cours de parallélisme)
  - isThreadSafe=false
    - performance dégradée
    - utiliser seulement si strictement nécessaire

JSP
- transfert de contrôle -

- <jsp:forward page="..." /> – chemin relatif à la servlet
- Efface la sortie actuelle
- Exemple :
  - <% if (request.getParameter ("db") == null) { %>
  - <jsp:forward page="help.jsp" />
  - <% } %>

JSP
- inclusion de fichiers -

- <jsp:include page="..." />
- Inclut la sortie du fichier pendant l'exécution
  - => le fichier est exécuté
  - => fichier non modifié si le fichier inclu est modifié

JSP
- beans : définition -

- Java Bean = composant (classe) obéissant à certaines conventions :
  - classe publique
  - constructeur publique sans argument
    - écrire un tel constructeur, s'il n'existe pas
  - pas de champ d'instance publique
    - remplacer les champs publiques par des champs privés (propriétés)
  - méthodes get/is et set pour accéder aux propriétés
    - propriété “color” : type getColor (), void setColor (type)
    - propriété “done” : boolean isDone (), void setDone (boolean)

JSP
- beans : exemple -

- Dans ~/public_html/WEB-INF/classes/exemple :
  - package exemple;
  - public class SBean{
  - private int size = 162;
  - public int getSize (){
  - return size;
  - }
  - public void setSize (int newSize){
  - size = newSize;
  - }
  - }

JSP
- beans prédéfinis -

- Les composants de Swing :
  - JButton, JDialog, JFrame, JTextField, ...
- Diverses classes :
  - Calendar, ...

JSP
- beans : avantages convention -

- Rajout de contraintes sur les valeurs :
  - public void setSize (int newSize){
  - if (newSize < 0)
  - sendErrorMessage (...);
  - else
  - size = newSize;
  - }

JSP
- beans : avantages convention -

- Changement du code interne :
  - public void setSize (int newSize){
  - size = meter2inch (newSize);
  - }
- Effets secondaires :
  - public void setSize (int newSize){
  - size = newSize;
  - updateSizeLabel ();
  - }

JSP
- beans : intérêt pour JSP -

- Le nom des méthodes est trouvé, par réflexion, à partir du nom des propriétés
  - => programmation visuelle possible
- Avec JSP :
  - instanciation simple des classes Java, sans écrire du code complet Java
  - réutilisation de code

JSP
- beans : exemple d'utilisation -

- <!DOCTYPE...>
- <html>
- <head>
- ...
- </head>
- <body>
- <jsp:useBean id="bean" class="exemple.SBean" />
- Value of bean : <jsp:getProperty name="bean" property="size" />
- </body>
- </html>

JSP
- beans : instanciation -

- <jsp:useBean attribut="valeur" />
- Exemples d'attributs :
  - id="name", permet de référencer plus tard cette instance de bean
    - si une instance de bean avec mêmes nom et portée existe, utilise celle-ci
      - utile surtout pour portées session et application
    - sinon, crée une instance
      - erreur si un bean de même nom et portée différente existe
  - class="pacquetage.class", nom complet du bean
    - un bean doit toujours se trouver dans un paquetage
  - type="pacquetage.class", synonime de class

JSP
- beans : instanciation -

- Exemples d'attributs :
  - scope="page|request|session|application", portée du bean
    - page (valeur par défaut) : disponible seulement dans cette page
    - request : disponible pendant la requête, avec ServletRequest (détruit à la fin de la requête)
    - session : disponible pendant toute la session, avec HttpSession.getAttribute (nom)
    - application : existe pendant toute l'application
- Un bean en vie signifie que la valeur de ses champs est gardée

JSP
- beans : utilisation -

- <jsp:useBeans id="bean" class="exemple.SBean" />
  - <% exemple.Bean bean = new exemple.Bean (); %>
- <jsp:setProperty name="bean" property="size" value="173" />
  - <% bean.setSize (173); %> conversion automatique !
- <jsp:getProperty name="bean" property="size" />
  - <%= bean.getSize () %>
- Imbrications JSP dans les valeurs possible :
  - <jsp:setProperty name="..." property="title" value="<%= request.getParameter (\"title\") %>" />
    - pour les types autres que String, utiliser e.g. Integer.parseInt

JSP
- beans : avantages -

- Les beans peuvent être remplacés par des scriptlets
- Mais :
  - beans peuvent être partagés entre pages / servlets
    - voir <jsp:useBean scope="..." />
  - beans plus faciles pour lecture des paramètres

JSP
- beans : utilisation des paramètres -

- Association d'une propriété à un paramètre du même nom :
  - <jsp:setProperty name="bean" property="size" />
- Association d'une propriété à un paramètre :
  - <jsp:setProperty name="bean" property="size" param="taille" />
- Association de toutes les propriétés aux paramètres :
  - <jsp:setProperty name="bean" property="*" />
  - propriété non initialisée si pas de paramètre du même nom

JSP
- beans : exemple complet -

- Exemple complet avec :
  - un bean, avec :
    - propriétés i et j
    - méthode String generate () retournant tous les nombres compris entre i et j
  - une page JSP recevant comme paramètres les deux propriétés du bean, qui exécute generate ()
  - voir le résultat si deux appels :
    - le 1er avec les deux paramètres
    - le 2ème avec un seul paramètre (paramètre retenu du 1er)
      - voir ses différentes portées

JSP
- beans : opérations conditionnelles -

- Instancie un bean et initialise des propriétés seulement lorsque le bean est instancié
- <jsp:useBean ...> xyz </jsp:useBean>
  - instancie un bean s'il n'est pas instancié
  - xyz (éléments setProperty) seront exécutées seulement si un bean nouveau est créé
  - pas de / à la fin de l'ouverture de la balise !

Intégration servlets / JSP - introduction -

- JSP meilleur :
  - génération de contenu HTML
- Servlet meilleure :
  - traitement complexe
  - formatage HTML complexe
  - sortie binaire (e.g. image)
- => combiner servlets et JSP pour des pages complexes

Intégration servlets / JSP - méthodologie -

- Requête originale affectée à une servlet
  - éventuellement, elle dispatche à la servlet appropriée
- Servlet fait le traitement
- Les résultats sont mis dans des beans
- Requête transférée au JSP approprié pour affichage

Intégration servlets / JSP - côté servlet -

- Écrire / extraire des données d'un bean
  - request.setAttribute ("key", new Expl ());
  - request.getAttribute ("key");
- Transférer le contrôle à une servlet / page JSP
  - getServletContext().getRequestDispatcher("path").forward (request, response);
  - "path" commence par / et est relatif à l'*application*
    - exemple : "/servlet/Achat"
  - la sortie générée est effacée
  - => il faut qu'aucune sortie n'ait pas été envoyée au client

Intégration servlets / JSP - côté JSP -

- Récupérer un bean
  - <jsp:useBean id="key" class="pkg.Expl" scope="request" />
    - le id est le même que la clé de setAttribute !
  - utiliser <jsp:setProperty ... /> et les autres
- Transférer le contrôle à une servlet / page JSP
  - <jsp:forward ... />
  - "path" commence par / et est relatif à l'*application*

Intégration servlets / JSP - partage des beans -

- scope=page rien
- scope=request request.setAttribute
- scope=session HttpSession.setAttribute
- scope=application getServletContext.setAttribute ?

JDBC

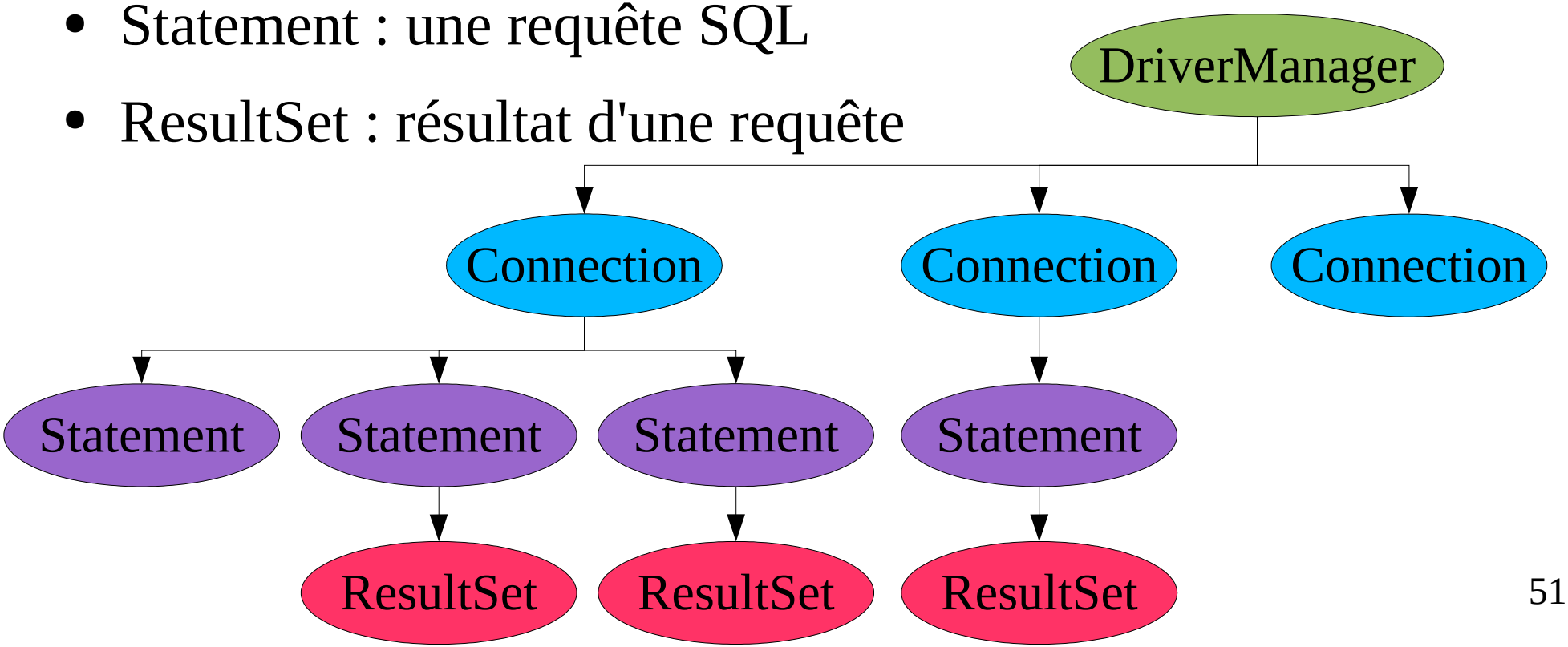
- Une SGBD utilise une certaine API (interface)
- Plusieurs APIs existent
- JDBC (Java Database Connectivity) = API SQL de bas niveau
  - bibliothèque standard pour accéder à des BDs
    - utilisable par toutes les SGBD
- import java.sql.*
- Voir la documentation

JDBC
- fonctionnement -

- établir connexion
- effectuer des requêtes
  - erreur si la BD ne reconnaît pas la requête
- utiliser les données obtenues (e.g. affichage)
  - besoin de types de variables conformes à SQL, e.g. dates et valeurs monétaires
- mettre à jour les informations
- terminer connexion

JDBC
- API -

- DriverManager : chargement de pilotes et connexion à la BD
- Connection : une connexion
- Statement : une requête SQL
- ResultSet : résultat d'une requête



JDBC
- exemple -

- URL url = new URL ("jdbc:odbc:Annuaire");
- Connection c = DriverManager.getConnection (url, "", "");
- Statement s = c.createStatement ();
- ResultSet rs = s.executeQuery ("select * from phone");
- while (rs.next ()){
  - String nom = rs.getString ("nom"); ...
- }
- rs.executeUpdate ("delete from phone where nom='...'");
- .close () sur tous les objets
- Voir ... pour un exemple complet