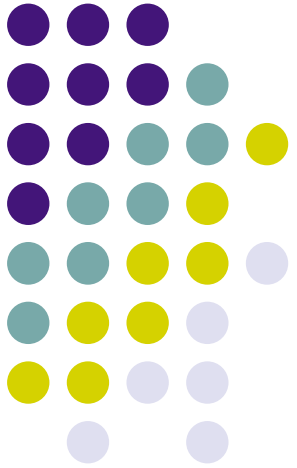
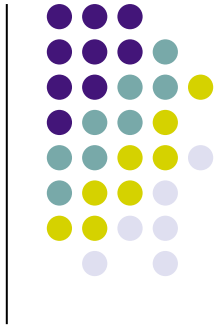
Web services: implémentation en J2EE: Axis 2 / JAX-WS

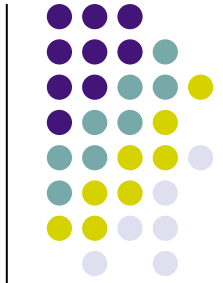


J2EE ?



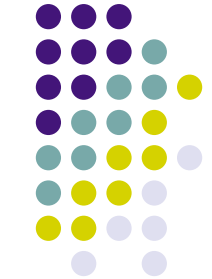
- Plateforme Java
- Serveur d'applications
  - Ensemble d'APIs/Frameworks dédiées:
    - JAX-WS, JSP/JSF, Java Beans, JPA, JMS …
- Programmation côté serveur
  - Logique applicative
  - Gestion de l'interfaçage avec les sources de données
  - Génération dynamique de contenu (pages webs, documents xml, images, …)

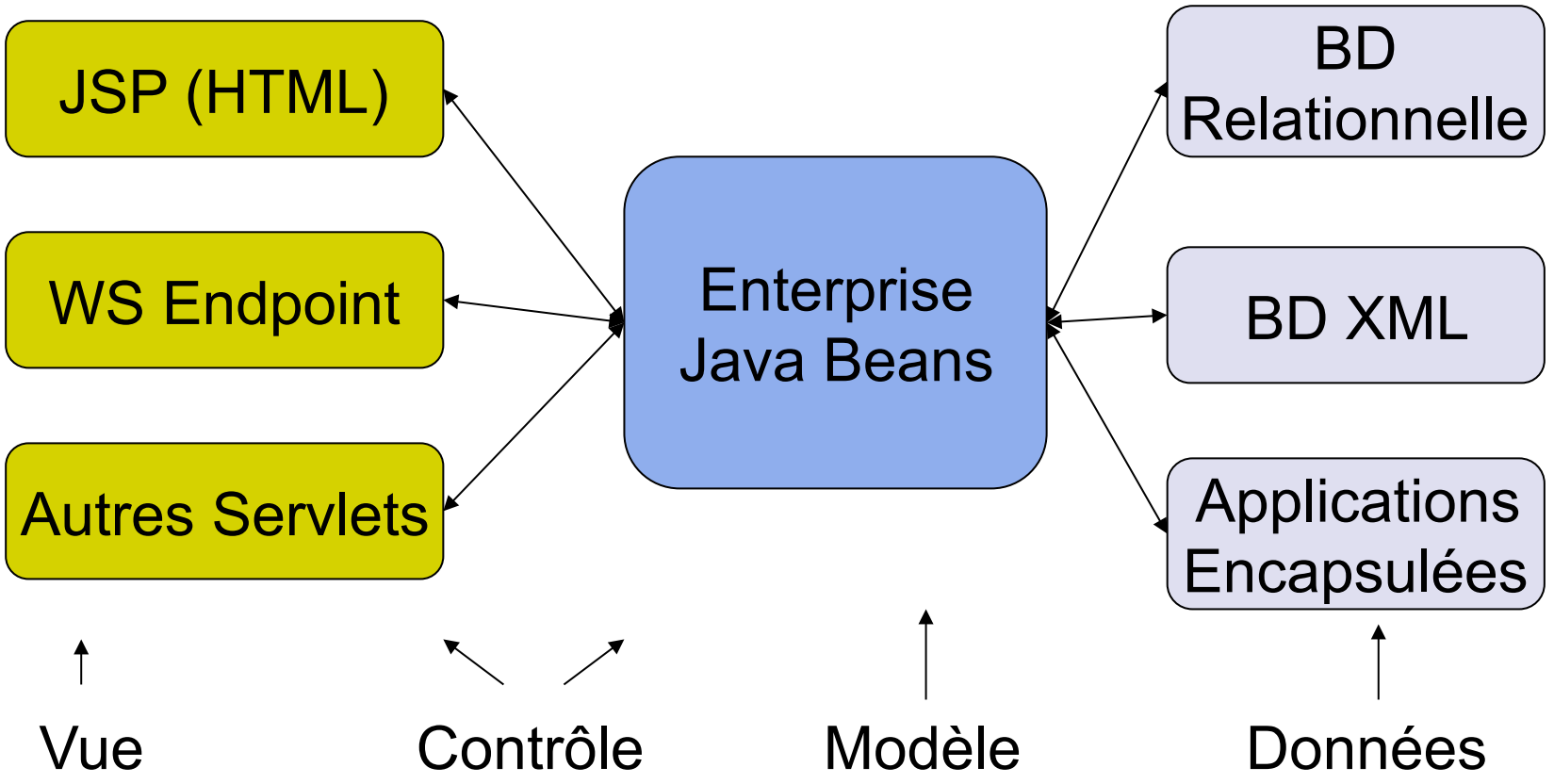
Quelques serveurs J2EE



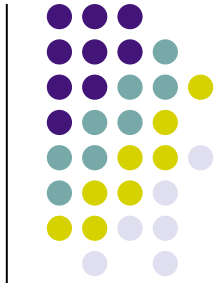
- Glassfish (SUN)
  - Integre Metro (implementation de JAX-WS)
  - Implementation de référence J2EE
- JBoss (JBoss.org)
- Jonas (ObjectWeb)
- WebSphere (IBM)
- Oracle Application Server
- Geronimo (apache.org)
- Tomcat (apache.org) + des bibliothèques

Modèle-vue-contrôleur: version J2EE



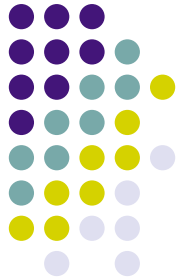


Serveur J2EE



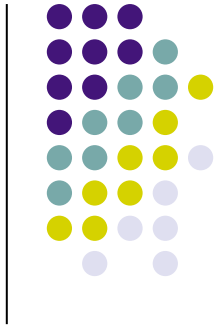
- Conteneur: serveur « contenant une machine virtuelle »
- Orientée composants
  - Chaque composant a un rôle précis
    - Permet de simplifier chacun des composants
  - De nombreux composants sont fournis de manière standard
    - Soit directement
    - Soit générés à partir de spécifications de haut niveau
- APIs destinées à simplifier (?) la programmation des différents composants de l'application

Servlets



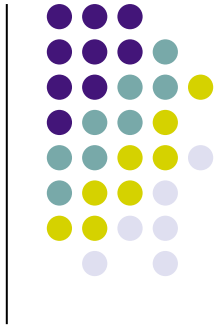
- Point d'entrée d'un client sur un serveur web J2EE
  - Pages dynamiques (JSP)
    - Similaires à des pages PHP, mais en Java
  - Web Services
    - En particulier si on utilise HTTP pour le transport

Servlets: principe



- Une classe Java
  - Qui implémente l'interface javax.servlet.Servlet
  - En général en étendant la classe javax.servlet.http.HttpServlet
- Un morceau dans un fichier de configuration (XML) du conteneur
  - Relie la classe à un emplacement (i.e. une URL) du serveur

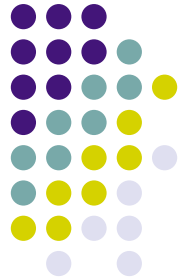
Servlets HTTP: du côté classes Java



- Méthodes Java appelées lors d'une requête d'un client:
  - doGet, doPost, doPut, doDelete
    - En fonction de la méthode HTTP utilisée
    - Arguments représentant la requête et la réponse
  - Init et destroy
    - Permettent de réserver et de libérer les ressources nécessaires à la servlet
  - Attention à l'aspect concurrent:
    - Une servlet peut être amenée à gérer plusieurs requêtes simultanément

Exemple …

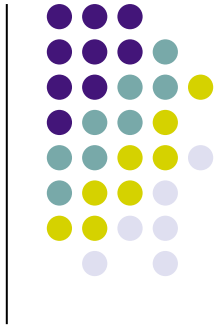
Servlets: du côté de la configuration du conteneur



- Fichier WEB-INF/web.xml:

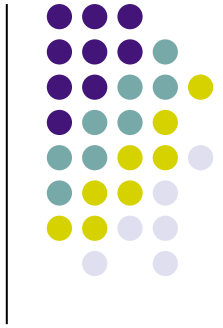
```
…
<servlet>
  <display-name>DatePage</display-name>
  <servlet-name>DatePage</servlet-name>
  <servlet-class>fr.univlyon1.ecoquery.cours.DatePage</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>DatePage</servlet-name>
  <url-pattern>/DatePage</url-pattern>
</servlet-mapping>
…
```

Apache Tomcat



- Conteneur de servlet uniquement
  - et non pas tout J2EE !
- Relativement léger
  - Installation de base ~12 Mo
    - > 100 Mo pour un serveur J2EE complet
- Possibilité d'ajouter des modules au besoin (ex: Axis 2 ou Metro)

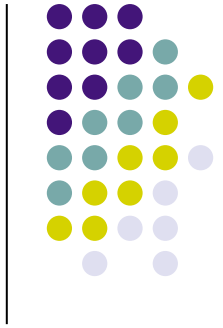
Servlets: suffisant pour des services Web basiques



- Un point d'accès à un service:
  - Reçoit une requête HTTP
  - Extrait de la requête un document SOAP
  - Effectue un traitement
  - Créé un document SOAP résultat
  - Renvoie ce document comme réponse

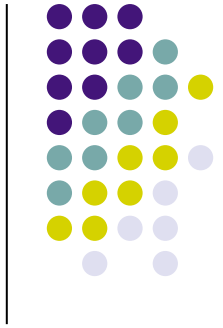
- Réalisable par une servlet en utilisant
  - les APIs XML Java
  - SAAJ

Correspondance objet <--> XML



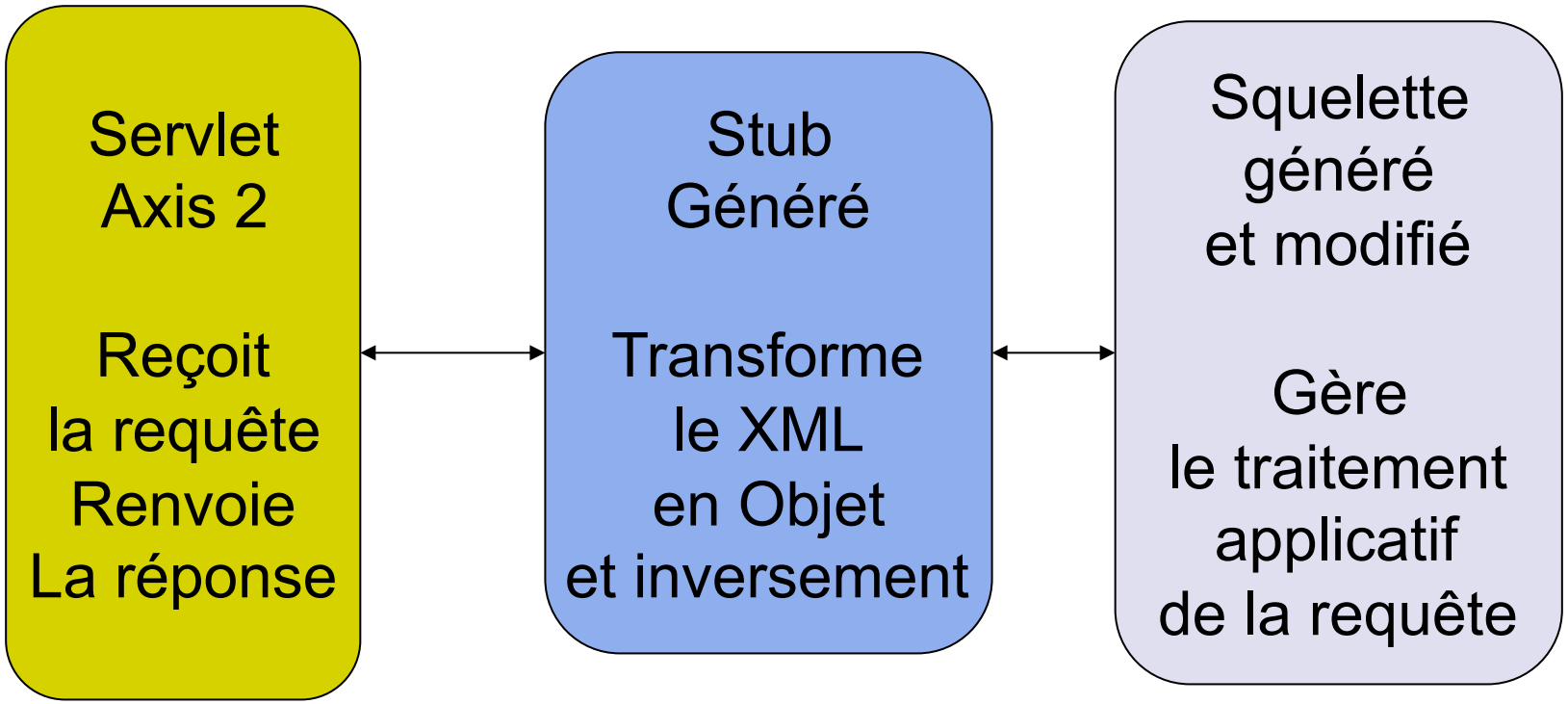
- Etant donné un modèle de document XML (i.e. un schéma XML ou une DTD) en faire une représentation sous forme d'objet Java
- Représenter les objets d'une certaine classe par un document XML
  - Sérialisation XML

Axis 2: WSDL2Java

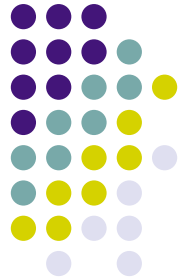


- Conversion d'un fichier WSDL en un ensemble de classes Java
  - Pour représenter le contenu des messages
  - Une classe squelette à compléter pour implémenter les différentes opérations
  - Éventuellement des classes intermédiaires qui vont s'occuper de la gestion du message SOAP (stubs)

Fonctionnement: runtime

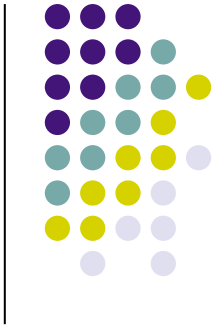


Axis2 Databinding Framework ADB



- Conversion la plus simple de XML en objet
  - Expanded mode (mode expansé)
    - Une classe pour chaque element externe
      - Non imbriqué dans un complexType
    - Une classe pour chaque complexType nommé
    - Mode systématiquement utilisé par le compilateur en ligne de commande
  - Wrapped mode (mode « intégré »)
    - Une classe contenant toutes les classes représentant les documents XML à traiter
- Limitations:
  - Difficulté à gérer les schémas complexes
    - Gère uniquement <sequence> et <all>
    - Pas de restriction ou d'extension de type complexes

ADB: Lien XML - Java



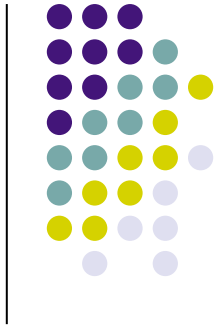
- Une classe par type complexe
  - Les attributs et les éléments sont transformés en variables d'instances
- Une classe par élément hors type complexe
- Les restrictions sur les types simples sont converties dans le type primitif Java le plus proche
  - En général, cela revient à utiliser le type de base qui a servi à la restriction

Exemple (tutoriel Axis 2)

- Fichier StockQuoteService.wsdl
- Génère:

adb/StockQuoteServiceMessageReceiverInOnly.java
adb/StockQuoteServiceMessageReceiverInOut.java
adb/StockQuoteServiceSkeleton.java
adb/StockQuoteServiceSkeletonInterface.java
pojo/ExtensionMapper.java
pojo/GetPrice.java
pojo/GetPriceResponse.java
pojo/Update.java

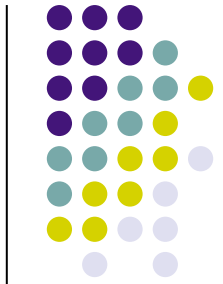
Génération de clients



- WSDL2Java peut être utilisé pour générer un client
- Plus précisement:
  - L'ensemble des classes pour (dé)sérialiser le XML
  - Une classe « stub » pour faciliter l'accès au service
  - Une classe client exemple

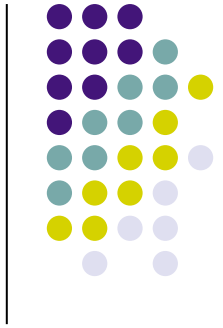
- Exemple: ADBClient.java

JAXB: ADB version JAX-WS



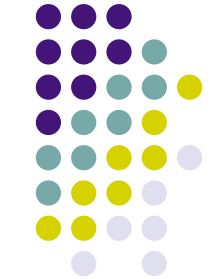
- Gère complètement XML Schema
- Bi directionnel:
  - Classes Java -> XML Schema
  - XML Schema -> classes Java
- Mécanisme d'annotations
  - Permet de personnaliser la liaison XML/Java

XMLBeans



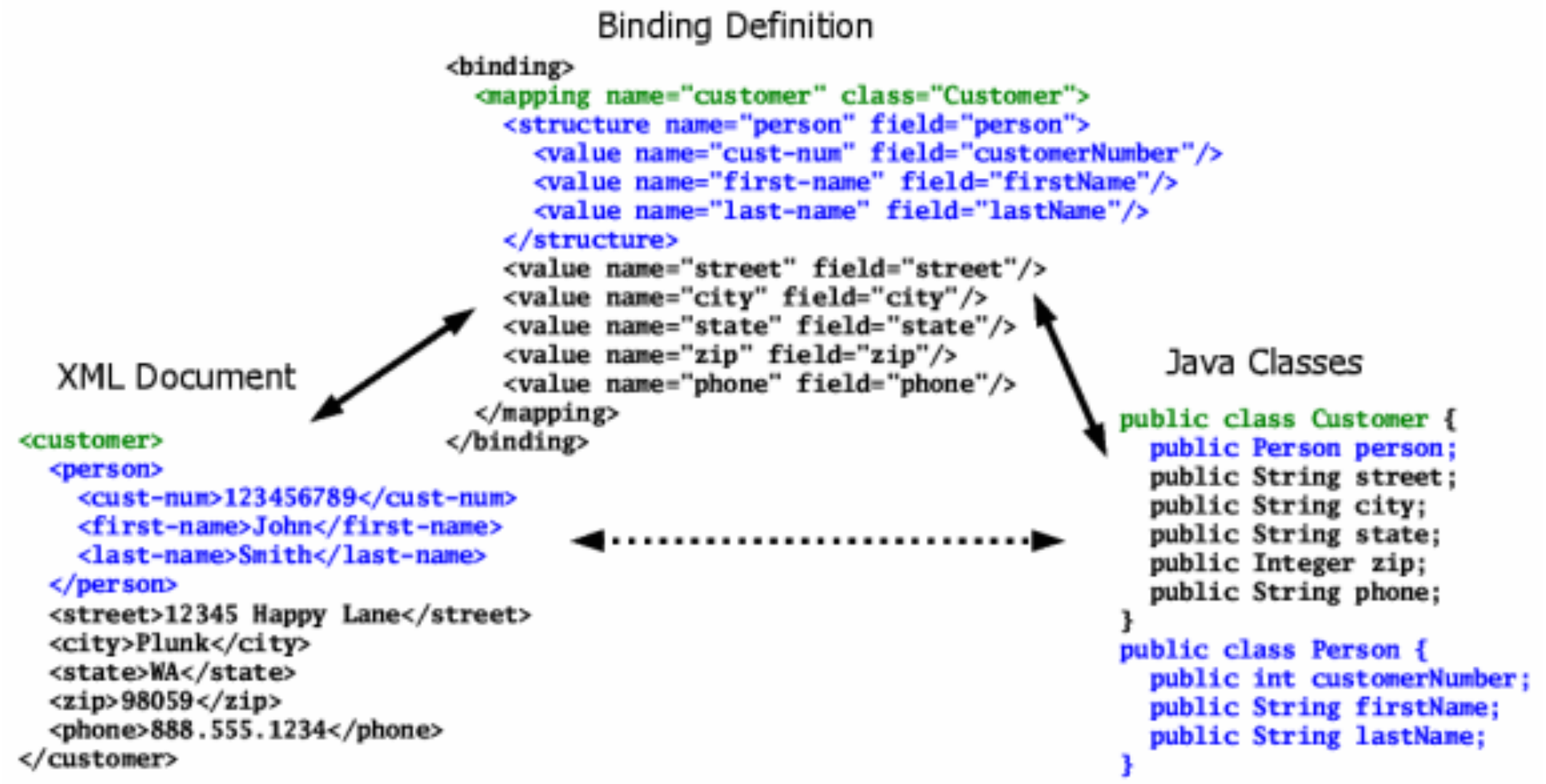
- Plus complet que ADB
- Conserve un lien fort avec le XML:
  - XmlCursor
    - Représente une position dans le document XML correspondant à l'objet
    - Accès bas niveau
  - Modèle objet proche du schéma XML de départ

JiBX: Binding XML to Java Code



- C'est le programmeur qui donne le lien entre le schéma XML et ses propres classes Java
  - Gain en flexibilité
  - Pratique lorqu'on utilise des classes déjà existantes
    - Dans le cas de l'exposition d'une application préexistante comme un service web
  - Un peu plus lourd à mettre en place

Exemple de binding simple



Source: tutoriel JiBX

Exemple JiBX - 2

Binding Definition

```
<binding>
  <mapping name="customer" class="Customer">
    <structure name="person" field="person"
        value-style="attribute">
      <value name="cust-num" get-method="getNumber"
        set-method="setNumber"/>
      <value name="first-name" field="firstName"/>
      <value style="text" field="lastName"/>
    </structure>
    <value name="street" field="street"/>
    <value name="city" field="city"/>
    <value name="state" field="state"/>
    <value name="zip" usage="optional"
      field="zip" type="java.lang.Integer"/>
    <value name="phone" field="phone"
      set-method="setPhone"/>
  </mapping>
</binding>
```

XML Document

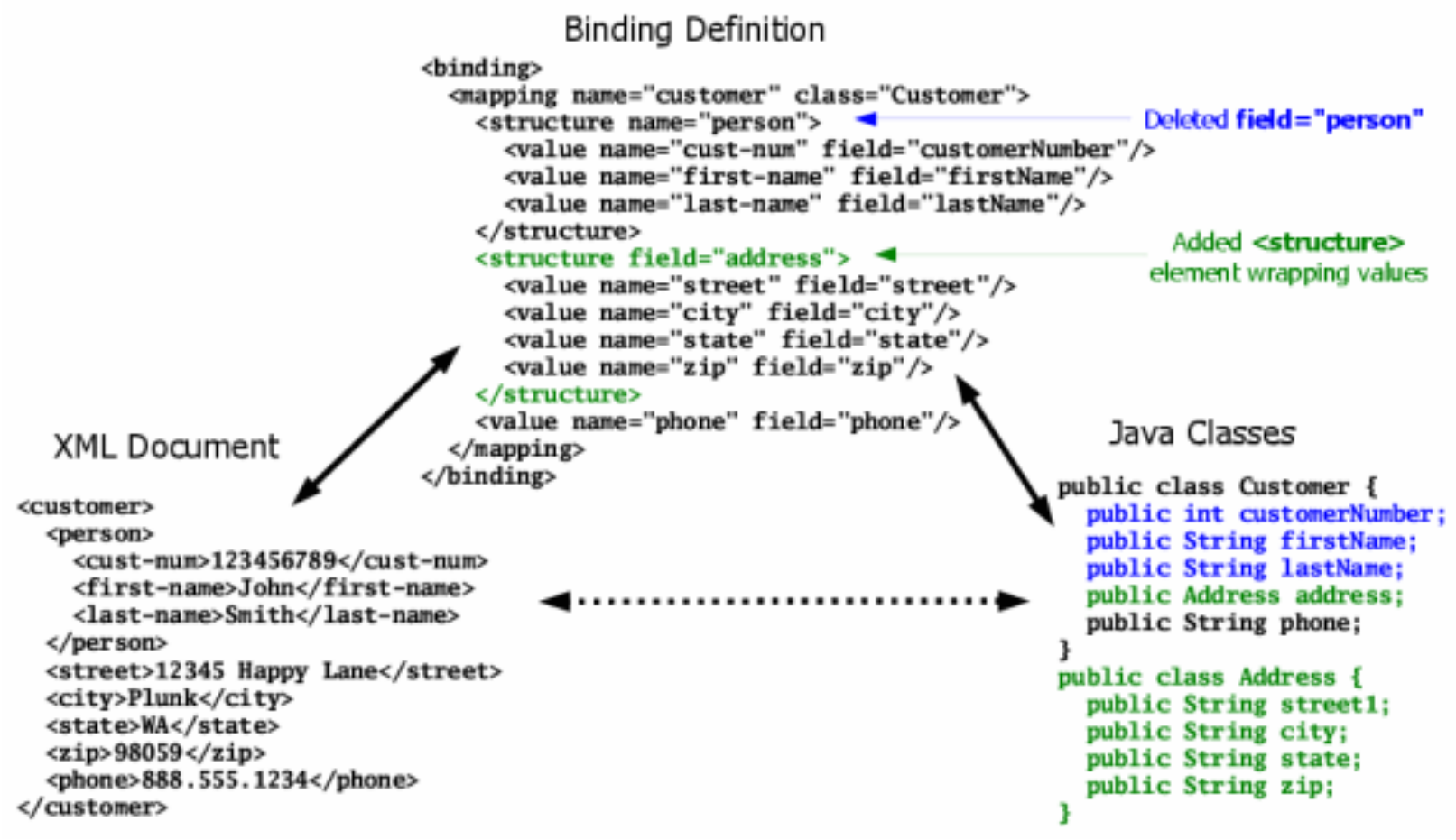
```
<customer>
  <person cust-num="123456789"
    first-name="John">Smith</person>
  <street>12345 Happy Lane</street>
  <city>Plunk</city>
  <state>WA</state>
  <phone>888.555.1234</phone>
</customer>
```

Java Classes

```
public class Customer {
  private Person person;
  private String street;
  private String city;
  private String state;
  private Object zip;
  private String phone;
  public void setPhone(String phone) {
    this.phone = phone;
  }
}
public class Person {
  private int customerNumber;
  private String firstName;
  private String lastName;
  protected int getNumber() {
    return customerNumber;
  }
  protected void setNumber(int num) {
    customerNumber = num;
  }
}
```

Source: Tutoriel JiBX

Exemple JiBX: variations de structures



Source: tutoriel JiBX

Exemple JiBX: collections

Binding Definition

```
<binding>
  <mapping name="timetable" class="TimeTable">
    <collection field="carriers" item-type="Carrier"/>
    <collection field="airports"/>
  </mapping>
  <mapping name="carrier" class="Carrier">
    <value style="attribute" name="code" field="code"/>
    <value style="attribute" name="rating" field="rating"/>
    <value name="url" field="url"/>
    <value name="name" field="name"/>
  </mapping>
  <mapping name="airport" class="Airport">
    <value style="attribute" name="code" field="code"/>
    <value name="location" field="location"/>
    <value name="name" field="name"/>
  </mapping>
</binding>
```

XML Document

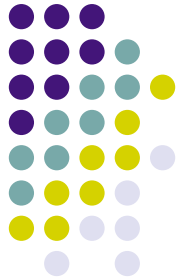
```
<timetable>
  <carrier code="NL" rating="4">
    <url>http://www.northleft.com</url>
    <name>Northleft Airlines</name>
  </carrier>
  ...
  <airport code="BOS">
    <location>Boston, MA</location>
    <name>Logan International</name>
  </airport>
  ...
</timetable>
```

Java Classes

```
public class TimeTable {
  private ArrayList carriers;
  private Object[] airports;
}
public class Airport {
  ...
}
public class Carrier {
  ...
}
```

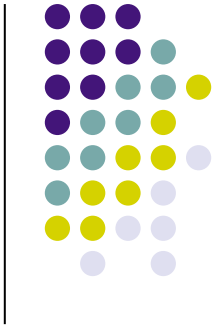
Source: tutoriel JiBX

Quelques options



- L'attribut ordered="false" indique que les valeurs peuvent ne pas être ordonnées comme dans la specification
- L'attribut flexible="true" indique que l'on ignore les valeurs qui n'apparaissent pas dans le binding

Déploiement



- Structure:

META-INF/services.xml

META-INF/monservice.wsdl

mon/nom/de/package/maclasse1.class

mon/nom/de/package/maclasse2.class

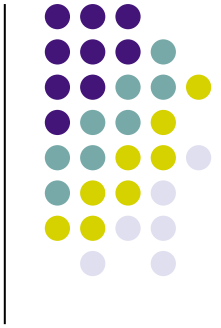
mon/nom/de/package/classegeneree.class

…

Description du service pour le runtime Axis
Généré par WSDL2Java

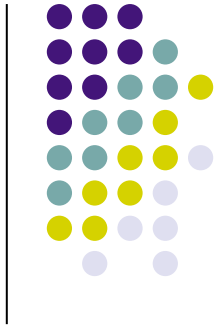
- Archive .aar (i.e. jar) à placer dans le repertoire de tomcat:
  - webapps/axis2/WEB-INF/services/

java2wsdl



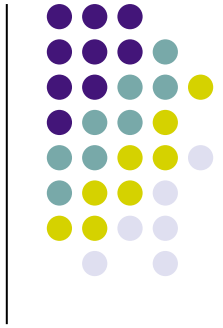
- Crée un fichier WSDL à partir d'une classe Java
- Bien pour:
  - Exposer facilement une application existante (vue à travers une classe) comme un service Web
  - Faire du prototypage
    - Plus rapide que le développement d'un fichier WSDL à la main
- Cette approche est cependant déconseillée en production: elle ne permet pas un bon contrôle du WSDL généré

JAX-WS



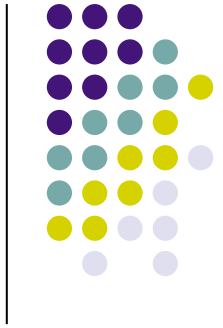
- Ensemble d'APIs pour la programmation de services web sur J2EE
  - Inclus SAAJ, JAXB
- Utilisation simplifiée par des annotations dans les classes Java
  - Spécification de la correspondance XML - Objets
  - Facilitation de la programmation des points d'entrée de services

Annotations JAXB



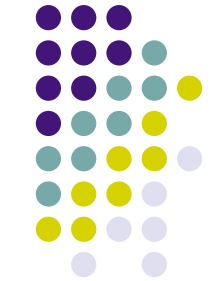
- Package java.xml.bind.annotation
- @XmlRootElement: classe pouvant être (dé) sérialisée dans un document
- @XmlElement: propriété codée sous forme d'élément
- @XmlAttribute: propriété codée sous forme d'attribut (attention au type -> type simple)
- @XmlTransient: non sérialisé
- @XmlValue: texte dans l'élément (pour les types complexes XML Schema basés sur des simpleContent)

Exemple: classe CD



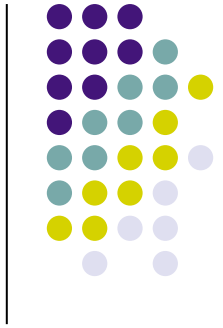
- Cd.java
- TestCd.java
- cd.xml

Annotations pour les points d'accès



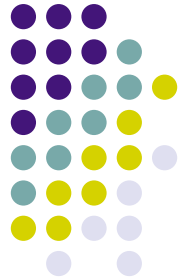
- Classe ou interface annotée avec @WebService
- Méthodes codant les opérations annotées avec @WebMethod
  - @WebParam peut permettre de spécifier des infos sur les paramètres (ex: quelle partie du message, ce paramètre provient-il du header, etc)
  - Les paramètres sont des objets extraits du XML via JAXB
  - @WebResult similaire à @WebParam
    - Utilisation de la classe javax.xml.ws.Holder<T>
- Annotation compatibles avec les beans (en particulier ajout possible de @Stateless)
- Déploiement: Mapping URL <-> service dans un fichier de configuration (sun-jaxws.xml pour Metro)

Génération de code



- A partir du Java:
  - Génération automatique de WSDL+XML Schema
- A partir du WSDL
  - Génération de classes annotées pour représenter les documents spécifiés par le schéma
  - Génération d'une interface annotée avec @WebService.
    - Il reste à l'implémenter

Travailler directement avec le XML



- Classe implémentant Provider<Source> ou Provider<SOAPMessage>
- @WebServiceProvider
- @ServiceMode
  - value=Service.Mode.MESSAGE ou PAYLOAD pour les Provider<Source>
- Plus léger que l'implémentation complète via une servlet, mais permet de gérer directement le XML.
- Se déploie similairement à une implémentation de service via une correspondance XML - objet