Java - XML Binding

Didier DONSEZ

Université Joseph Fourier (Grenoble 1)

IMA – IMAG/LSR/ADELE

Didier.Donsez@imag.fr

Sommaire

- Rappel XML
- Objectif
- Outils
  - JAXB
  - java.beans.XMLDecoder/XMLEncoder
  - Castor
  - XMLBeans
  - Zeus
  - JDOM

XML - Extensible Markup Language

- Document semi-structuré
  - Bien formé
  - Valide (respecte une structure DTD ou XML Schema)
- Exemple

```
<?xml version="1.0"?>
<USAddress>
    <name>Alice Smith</name>
    <street>333 Huntington Ave</street>
    <city>Boston</city>
    <state>MA</state>
    <zip>02115</zip>
</USAddress>
```

DTD - document type definition

- Héritage de SGML
- Exemple

```
<!DOCTYPE USAddress [
    <!ELEMENT USAddress(name, street, city, state, zip) >
    <!ELEMENT name    (#PCDATA) >
    <!ELEMENT street  (#PCDATA) >
    <!ELEMENT city    (#PCDATA) >
    <!ELEMENT state   (#PCDATA) >
    <!ELEMENT zip     (#PCDATA) >
]>
```

- Critiques
  - pas de typage, pas de cardinalité, pas d'imbrication, …

XML Schema (W3C)

- Typage, Cardinalité, Imbrication, Modularité
  - XSI : types simples
  - XSD: types complexes (structuré)

XML Schema (W3C)

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.mycomp.com/XMLSchema/mytypes" >
 <xsd:element name="zip-code">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:pattern value="[0-9]{5}(-[0-9]{4})?"/>
       </xsd:restriction>
    </xsd:simpleType>
 </xsd:element>
 <xsd:element name="USAddress">
<xsd:complexType>
 <xsd:sequence>
   <xsd:element name="name"        type="xsd:string"/>
   <xsd:element name="street"      type="xsd:string"/>
   <xsd:element name="city"        type="xsd:string"/>
   <xsd:element name="state"       type="xsd:string"/>
   <xsd:element name="zip"         type="zip-code"/>
 </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

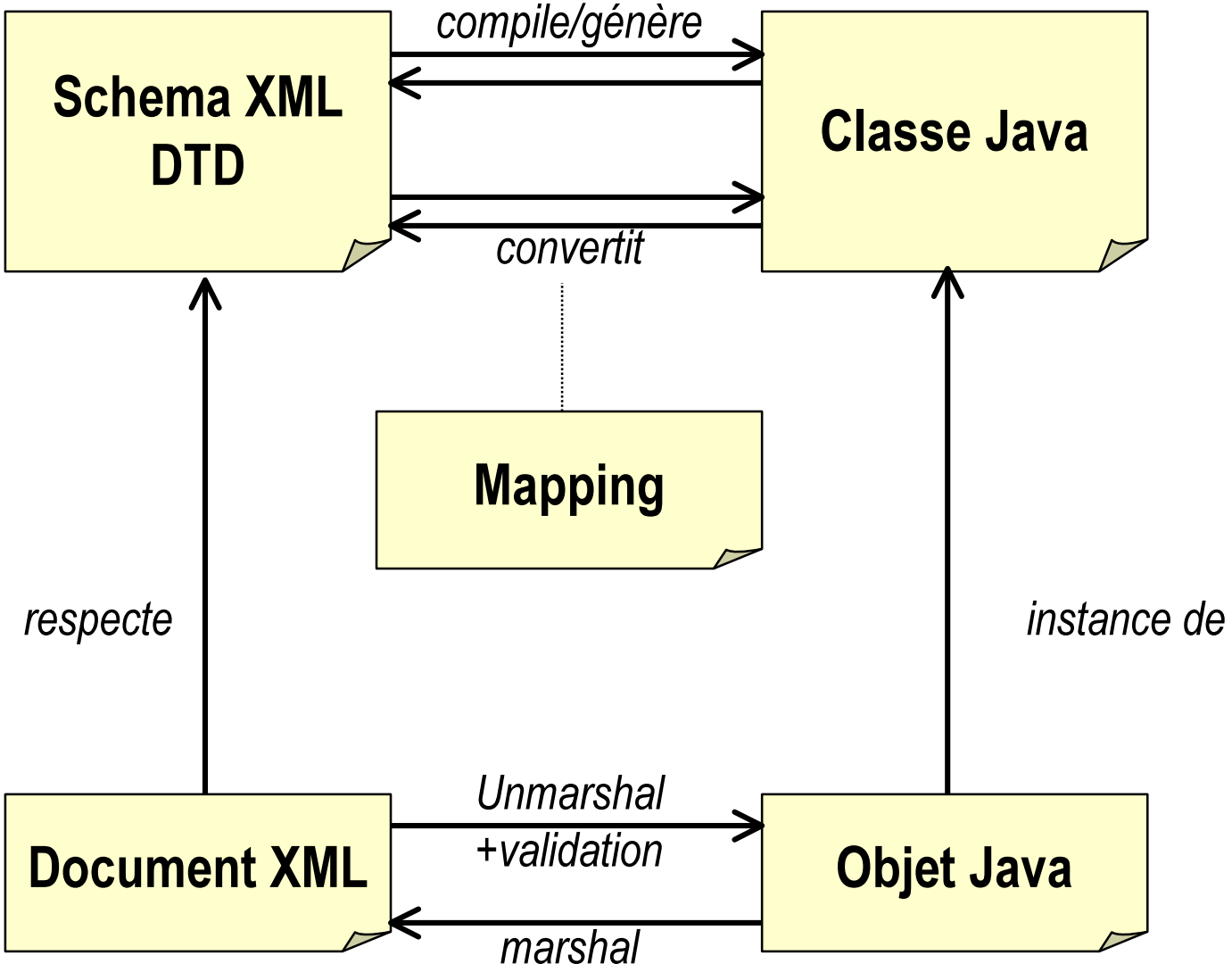
Remarques sur XML

- Représentation de documents semi-structurés
  - Donc très polymorphe (nombreuses alternatives de représentation)
- Graphe de données (en DTD)
  - Attribut de type ID :
    - L'élément est identifié de manière unique (portée : le document)
  - Attribut de type IDREF, IDREFS :
    - Référence un élément identifié (ID)
  - Pas de typage des références
- Références (intra-extra document)
  - Xlink, Xpointer

Manipulation XML depuis Java

- SAX - Simple API for XML
  - Interface événementielle
- DOM - Document Object Model
  - Interface navigationnelle dans un arbre
- XML Data-Binding (JSR-031)

Objectif du Binding XML - Java



Usage

- Fichiers de configuration, de déploiement, …
  - Web apps, EJB, Manifest XML …
- Persistance Objet Java
- …
- Web Services
  - Rappel SOAP
    - Les paramêtres et résultats des requêtes sont encodés en XML

      XML Schema, XMI, …
  - Rappel WSDL
    - Les champs de messages sont des types XML Schema
  - Désérialisation/Sérialisation
    - Java vers/depuis les paramètres des messages SOAP

Exemple de classe Java générée à partir d'un schéma ou d'une DTD

- Exemple

```
public class USAddress {
    public USAddress ( String name, String street, String city,
                                    String state, String zip) { ... }
    public String getName() { ... }
    public void setName(String name) { ... }
    public String getStreet() { ... }
    public void setStreet(String street) { ... }
            ......
    public void marshal(OutputStream out) throws IOException { ... }
    public static USAddress unmarshal(InputStream in)
                                throws IOException { ... }
    ... // SAX, DOM, ...
}
```

Exemple d'usage

```
File f = new File("address.xml");
FileInputStream fis = new FileInputStream(f);
AddressUS a=USAddress.unmarshal(fis);
fis.close();

a.setName="Alice Dow";

FileOutputStream fos = new FileOutputStream(f);
a.marshal(fos));
fis.close();
```

Outils

- Type
  - Génération à la conception
    - Possibilité de configurer les champs à lier, …
  - Génération à la Exécution
    - Pas de configuration
- Procédé
  - Automatique (conception, exécution)
    - Se base sur l'introspection Java, JavaBeans (setter/getter)
  - Manuelle (conception)
    - GUI, …
- Cibles
  - Tout objet Java
  - Sous typage d'une classe racine (EJB, …)
  - JavaBeans seulement

Outils

Type (2)

- Centré Document
  - Générer des classes Java à partir de tout type de document

    DOM +/- API propriétaire
- Centré Objet
  - Générer des schémas XML et DTD à partir de toute classe Java

    DTDs et XML Schémas propriétaires
- Centré Données
  - Représente une information en XML Schéma et par une/des classes Java

Outils

- Conception
  - VB, C#, C++ (Microsoft COM ou .NET)
    - .NET, SchemaCoder
  - Java
    - Exolab's Castor http://castor.exolab.org/
    - Sun's JAXB
    - ObjectWeb JORM ??
    - ZOPE, Zeus (enhydra), Schema2Java, Oracle XML, Informix, …
- Exécution
  - J2SE1.4 java.beans.XMLEncoder/XMLDecoder
  - XMLConverter, JSX, …

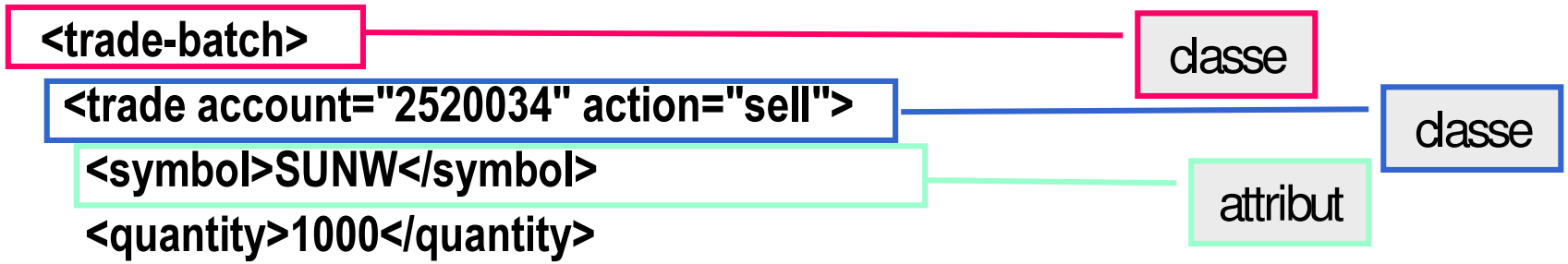
JAXB (SUN)

- Génération de classes Java à partir
  - de la DTD
  - d'un fichier de correspondance (binding schema) (`.xjs`)

- Remarque
  - Early Access
  - Succeptible de bouger et d'utiliser XML Schema plutôt que la DTD !

Exemple de fichier XML à unmarshaller

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE trade-batch SYSTEM "trade-batch.dtd">
<trade-batch>
  <trade account="2520034" action="sell">
    <symbol>SUNW</symbol>
    <quantity>1000</quantity>
    <limit>115</limit>
    <date>2001-05-05</date>
  </trade>
  <trade account="9240196" action="buy">
    <symbol>CSCO</symbol>
    <quantity>500</quantity>
    <date>2001-05-05</date>
  </trade>
</trade-batch>
```



Les DTDs

```
<!ELEMENT trade ( symbol, quantity, limit?, date ) >

<!ATTLIST trade
        account CDATA #REQUIRED
        action ( buy | buy-to-cover | sell | sell-short ) #REQUIRED
>

<!ELEMENT symbol (#PCDATA) >
<!ELEMENT quantity (#PCDATA) >
<!ELEMENT limit (#PCDATA) >
<!ELEMENT date (#PCDATA) >



<!ENTITY % trade SYSTEM "trade.dtd">
%trade;

<!ELEMENT trade-batch (trade+) >
```

Le fichier de binding (i)

```
<xml-java-binding-schema version="1.0-ea">

<!– Convertisseur de type Java -->
  <conversion name="price" type="java.math.BigDecimal"/>
  <conversion name="date" type="java.util.Date"
              parse="Cnv.parseDate" print="Cnv.printDate"
  />

<!-- Attributs Java -->
  <element name="symbol" type="value"/>
  <element name="quantity" type="value" convert="int"/>
  <element name="limit" type="value" convert="price"/>
  <element name="date" type="value" convert="date"/>
  …
```

Le fichier de binding (ii)

```
<!-- Les classes Java et leurs attributs -->
 <element name="trade" type="class" root="true">
  <content>
   <element-ref name="symbol"/>
   <element-ref name="quantity"/>
   <element-ref name="limit" property="limit-price"/>
   <element-ref name="date"/>
  </content>
  <enumeration name="Action" members="buy buy-to-cover sell sell-short"/>
  <attribute name="account" convert="int"/>
  <attribute name="action"/>
 </element>

 <element name="trade-batch" type="class" root="true">
  <content property="Trades"/>
 </element>
</xml-java-binding-schema>
```

Classe de conversion

```
public class Cnv {
    private static java.text.SimpleDateFormat df
            = new java.text.SimpleDateFormat("yyyy-MM-dd");
    public static java.util.Date parseDate(String s)
                        throws java.text.ParseException {
        return df.parse(s);
    }
    public static String printDate(java.util.Date d) {
        return df.format(d);
    }
}
```

Utilisation

- Génération des classes Trade et Trades (+ sous classes)
  - $JAXBHOME/bin/xjc trade-batch.dtd trade-batch.xjs
- Appel au marshaller/unmarshaller

```
import java.io.*;
import javax.xml.bind.*;

public class TradeLister {
    public static void main(String[] args) throws Exception {
     Dispatcher d = Trade.newDispatcher();
     Object ob = d.unmarshal(new FileInputStream(new File(args[0])));
     System.out.println(ob);
    }
}
```

java.beans.XMLDecoder/XMLEncoder

- Codage/Décodage automatique d'objets Java
  - Depuis J2SE 1.4
- Exemple codage

```
JFrame jf=new JFrame("Sample XML Encoder");
...
jf.add(new JButton("Hello, world"));
XMLEncoder e = new XMLEncoder(new FileOutputStream("Test.xml"));
e.writeObject(jf); e.close();
```

- Exemple codage

```
XMLDecoder d = new XMLDecoder( new FileInputStream("Test.xml"));
Object result = d.readObject(); d.close();
JFrame jf=(JFrame)result;
```

java.beans.XMLDecoder/XMLEncoder

- Fiche Test.xml

```
<?xml version="1.0" encoding="UTF-8"?>
 <java version="1.0" class="java.beans.XMLDecoder">
 <object class="javax.swing.JFrame">
  <void property="name"><string>Sample XML Encoder</string></void>
  <void property="bounds">
   <object class="java.awt.Rectangle">
     <int>0</int><int>0</int><int>200</int><int>200</int>
   </object>
  </void>
  <void property="contentPane">
   <void method="add">
    <object class="javax.swing.JButton">
     <void property="label"><string>Hello World !</string></void>
    </object>
   </void>
  </void>
  <void property="visible"><boolean>true</boolean></void>
 </object></java>
```

Castor (Exolab)

- Générateur de classes Java à partir
  - de schémas XML (XSD)
- Mapping (format XML)
  - Définit une correspondance (conversion)
    entre une structure XML et une classe Java **existante**.
    - Patrimonial / Legacy
- Également JDO, DSML, …

Marshal/Unmarshal dans Castor

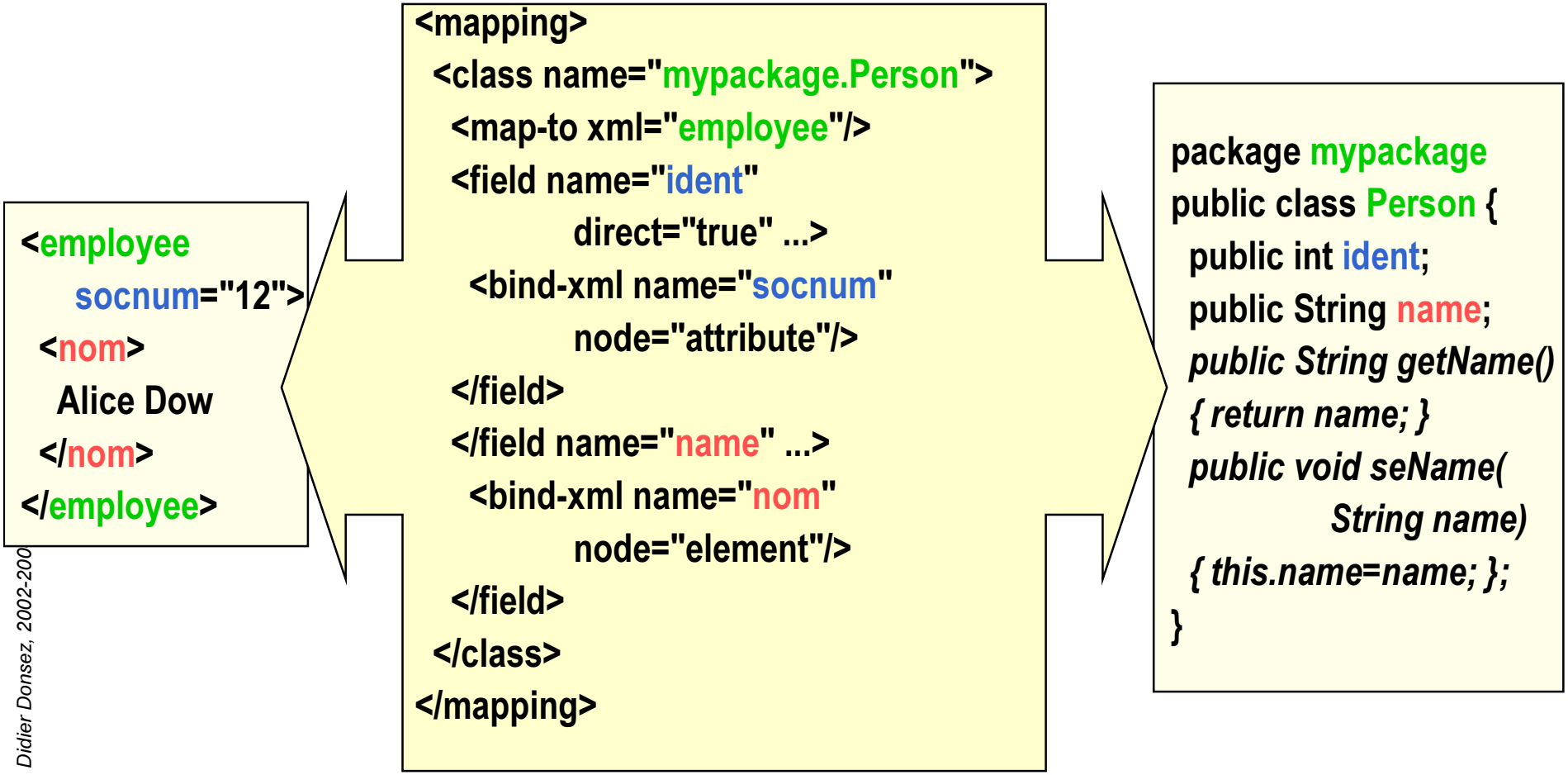
```
class org.exolab.castor.xml.Marshaller …{
    static void marshal(Object obj, DocumentHandler handler); // SAX
    static void marshal(Object obj, Node node); // DOM
    static void marshal(Object obj, Writer writer); // File
    …
}
class org.exolab.castor.xml.Unmarshaller …{
    Object unmarshall(EventProducer events); // SAX
    static Object unmarshall(Class cls, Node node); // DOM
    static Object unmarshall(Class cls, Reader reader) // File
    static Object unmarshall(Class cls, InputSource) // File
    …
}
```

Castor

```
// Create a new Person
Person person = new Person("Alice Smith");
person.setDateOfBirth(new Date(1955, 8, 15));

// Create a File to marshal to
writer = new FileWriter("person.xml");

// Marshal the person object
Marshaller.marshal(person, writer);

// Create a Reader to the file to unmarshal from
reader = new FileReader("person.xml");

// Marshal the person object
Person person = (Person)Unmarshaller.unmarshal(Person.class, reader);
person.setName("Alice Dow");
```

Castor - Mapping

- Définit une correspondance
  entre une structure XML et une classe Java existante.



Castor - Mapping

```
// Load Mapping
Mapping mapping = new Mapping();
mapping.loadMapping("mapping.xml");

// Create a Reader to the file to unmarshal from
reader = new FileReader("employee.xml");

// Create a new Unmarshaller
Unmarshaller unmarshaller = new Unmarshaller(Person.class);
unmarshaller.setMapping(mapping);
// Unmarshal the person object
Person person = (Person)unmarshaller.unmarshal(reader);
```

Apache XMLBeans
(http://xmlbeans.apache.org/)

- Basé sur XML Schema
  - Full XML Schema support.
  - Full XML Infoset fidelity.
- 3 API
  - XmlObject
    - The java classes that are generated from an XML Schema are all derived from XmlObject. These provide strongly typed getters and setters for each of the elements within the defined XML. Complex types are in turn XmlObjects.
  - XmlCursor
    - From any XmlObject you can get an XmlCursor. This provides efficient, low level access to the XML Infoset. A cursor represents a position in the XML instance.
  - SchemaType
    - XMLBeans provides a full XML Schema object model that you can use to reflect on the underlying schema meta information.

ObjectWeb Zeus (http://zeus.objectweb.org/)

- DTD et XML Schema
- Pluggable constraints

JDOM (http://www.jdom.org/)

- TODO

Ressources

- http://www.rpbourret.com/xml/XMLDataBinding.htm
- Chapitre 15
  - http://www.wrox.com/Support/PDF/SampleChapter_5059.pdf