

Les services Web Java EE

François Robert

francois.robert@shipstone.org



De quoi parle t'on ?

- Java EE
- Service Web
- SOAP, REST & RESTful
- Web Oriented Architecture

Le plan !

- Java EE & REST, définition
- JAXB
- SOAP
- REST
- Services REST vs SOAP ?
- Introduction WOA

Les outils du cours

- VM Linux (xubuntu)
- Java 8
- IDEs
 - IntelliJ
 - Netbeans
- Maven
- Serveur d'application
 - Glassfish
 - Wildfly
 - Jetty (container de servlet)
- Git
- shell



maven

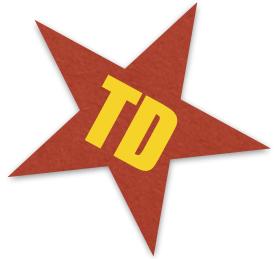


ji





Initialisation

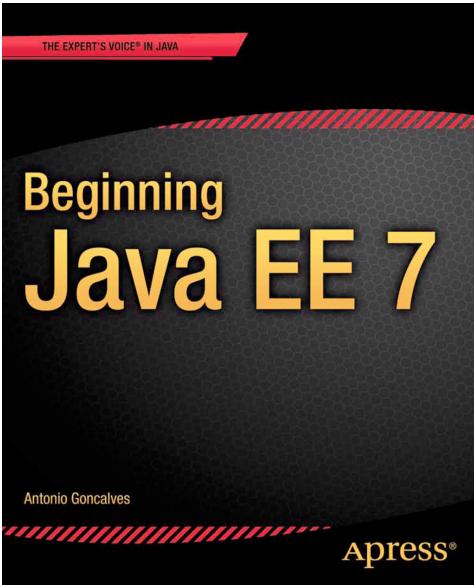


- Lancer la VM
 - login : siad - password : javeeesiad
- Ouvrez une console et rendez vous dans le repertoire de dev
 - Créez un repertoire workspaces et placez vous dedans
 - Astuce :mkdir workspaces && cd \$_
- Il s'agit maintenant de cloner le repository de source (hébergé par github)
 - Exécutez git clone https://github.com/ptitbob/siad_m2_ws.git
 - Placez vous dans le répertoire siad_m2_ws
 - Exécutez mvn clean compile (il devrait télécharger le web)

Pour faire les mise à jour pour chaque cours (si la VM n'a pas été réinitialisée) :

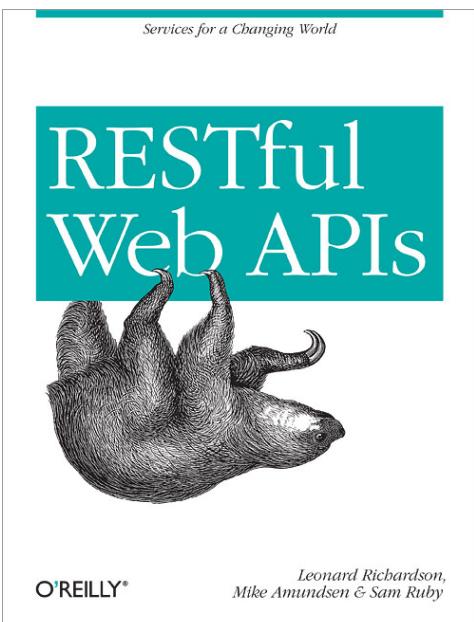
- git pull origin
- mvn clean compile

Références

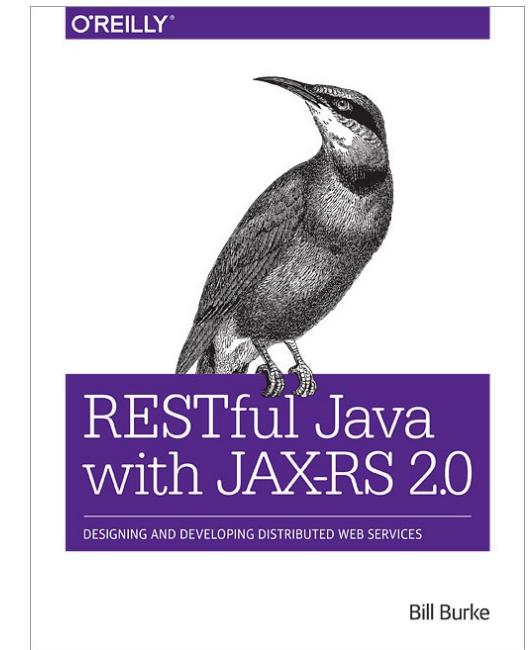


Beginning Java EE 7
Antonio Goncalves
aPress 2013

RESTful Java with Jax-RS 2.0
Bill Burke
O'Reilly 2013



RESTful Web APIs
Leonard Richardson, Mike Amundsen, Sam
Ruby
O'Reilly 2013



Java EE 7

Un standard

C'est un ensemble de plus de 30 spécifications

Coiffées par la JSR 342

Et réparties dans 5 domaines

Web Application Technologies

Enterprise Application Technologies

Web Services Technologies

Management and Security Technologies

Java EE-related Specs in Java SE

Java EE 7

Les principaux composants

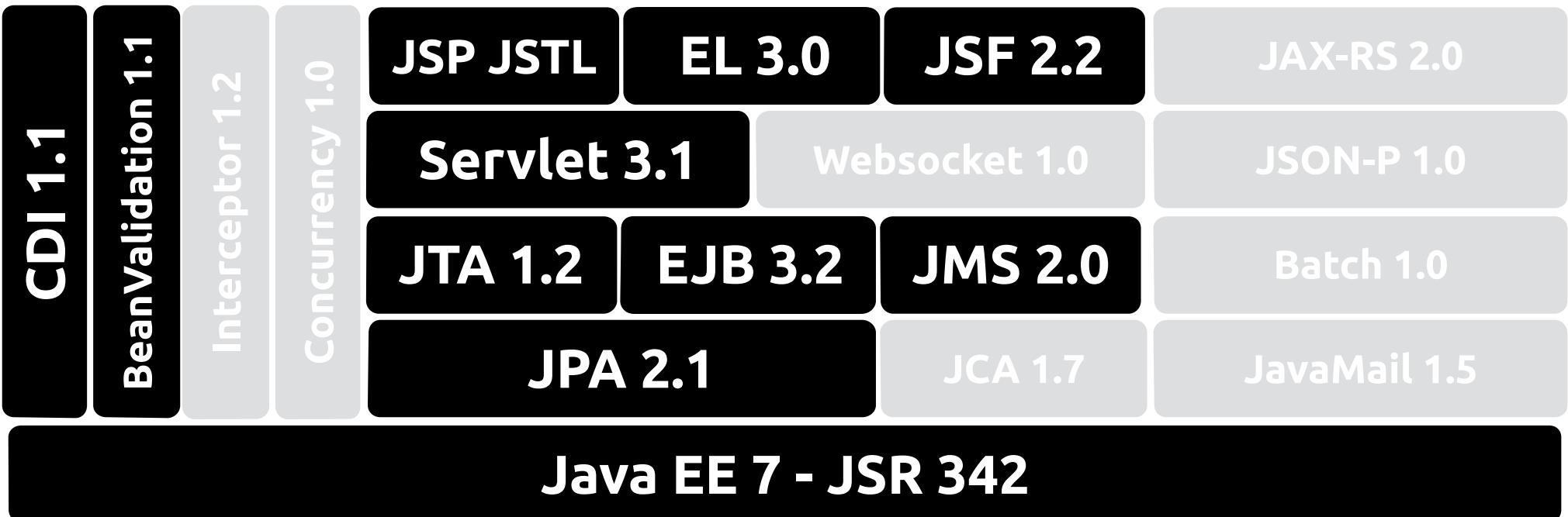
CDI 1.1	BeanValidation 1.1	Interceptor 1.2	Concurrency 1.0	JSP JSTL	EL 3.0	JSF 2.2	JAX-RS 2.0
				Servlet 3.1	Websocket 1.0		JSON-P 1.0
				JTA 1.2	EJB 3.2	JMS 2.0	Batch 1.0
				JPA 2.1		JCA 1.7	JavaMail 1.5

Java EE 7 - JSR 342

Java EE 7

Les principaux composants

Vu en MI



Java EE 7

Les principaux composants

Les web Services

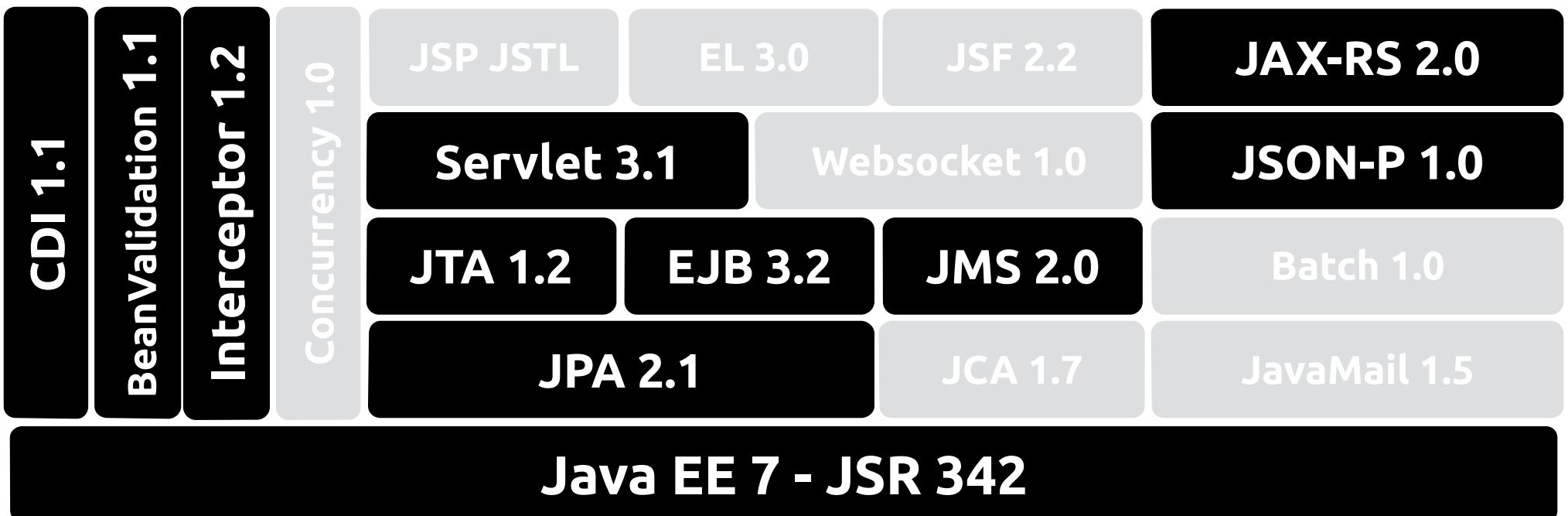


Java EE 7

Les principaux composants

Les briques d'une application d'entreprise basé sur les WS

C'est évidemment une architecture presque minimale



Définitions

- Echange de **services** de façon **faiblement couplée** entre un **fournisseur** et un **demandeur**
- **Service** : fonction métier
- **Fournisseur** : celui qui offre le service
- **Demandeur** : celui qui consomme le service
- **Faiblement couplée** : Fournisseur et Demandeur ignorent les détails de l'implémentation du partenaire

Type de service Web

- WS-* :Web Services - * : repose sur les standards SOAP et WSDL
- REST : Representational State Tranfer : repose sur les standards HTTP et URI

XML & JAXB

JAXB

- **Java Architecture for XML Binding (Java SE)**
- Présent dans JavaEE (7) et JavaSE (1.8)
- JSR 222 - révision 2 - version 2.2.3
- Permet transformation XML <-> Java
- Marshalling : transformer un objet en XML
- Unmarshalling : transformer du XML en objet
- Gestion du XSD
- xjc et schemagen pour la génération

Annotation (1)

```
@XmlElement(name = "book", namespace = "http://univ-blois.fr/ws/book")
@XmlAccessorType(XmlAccessType.FIELD)
@Entity
public class Book {

    @XmlAttribute(name = "id", required = true)
    @Id
    private Long id;

    private String title;

    private String isbn;

    @Enumerated(EnumType.STRING)
    private Type type;

    public Book() {
    }

    ...
}
```

@XmlRootElement

```
@Retention(RUNTIME)
@Target({TYPE})
public @interface XmlRootElement {

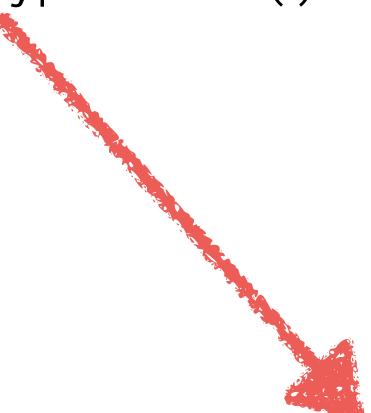
    String namespace() default "##default";

    String name() default "##default";

}
```

@ XmlAccessorType

```
@Inherited @Retention(RUNTIME) @Target({PACKAGE, TYPE})
public @interface XmlAccessorType {  
  
    XmlAccessType value() default XmlAccessType.PUBLIC_MEMBER;
}
```



```
public enum XmlAccessType {  
    PROPERTY,  
    FIELD,  
    PUBLIC_MEMBER,  
    NONE  
}
```

@ XmlAttribute

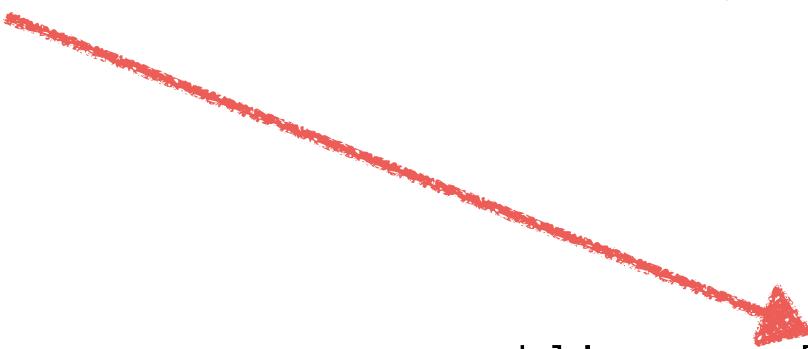
```
@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlAttribute {

    String name() default "##default";

    boolean required() default false;

    String namespace() default "##default" ;
}
```

Le cas des énumérés

```
@Target({METHOD, FIELD})  
@Retention(RUNTIME)  
public @interface Enumerated {  
  
    EnumType value() default ORDINAL;  
}  
  
  
public enum EnumType {  
    ORDINAL,  
    STRING  
}
```

@ Enumerated

```
@XmlType()  
@XmlEnum(String.class)  
public enum Type {  
    SCIFI, POLAR, ROMAN, EDUCATION, INFORMATIQUE, COMICS;  
}
```

```
<xs:simpleType name="type">  
    <xs:restriction base="xs:string">  
        <xs:enumeration value="COMICS"/>  
        <xs:enumeration value="SCIFI"/>  
        <xs:enumeration value="ROMAN"/>  
        <xs:enumeration value="POLAR"/>  
        <xs:enumeration value="INFORMATIQUE"/>  
        <xs:enumeration value="EDUCATION"/>  
    </xs:restriction>  
</xs:simpleType>
```

@ Enumerated

```
@XmlType()  
@XmlEnum(Integer.class)  
public enum Type {  
    @XmlElementValue("10")  
    SCIFI,  
    @XmlElementValue("20")  
    POLAR,  
    @XmlElementValue("40")  
    ROMAN,  
    @XmlElementValue("50")  
    EDUCATION,  
    @XmlElementValue("60")  
    INFORMATIQUE,  
    @XmlElementValue("70")  
    COMICS;  
}
```

```
<xs:simpleType name="type">  
    <xs:restriction base="xs:int">  
        <xs:enumeration value="20"/>  
        <xs:enumeration value="40"/>  
        <xs:enumeration value="70"/>  
        <xs:enumeration value="10"/>  
        <xs:enumeration value="60"/>  
        <xs:enumeration value="50"/>  
    </xs:restriction>  
</xs:simpleType>
```

Annotation (1)

```
@XmlRootElement(name = "book", namespace = "http://univ-blois.fr/ws/book")
@XmlAccessorType(XmlAccessType.FIELD)
@Entity
public class Book {

    @XmlAttribute(name = "id", required = true)
    @Id
    private Long id;
    @XmlElement(name = "title")
    private String title;
    @XmlElement(name = "isbn")
    private String isbn;

    @Enumerated(EnumType.STRING)
    private Type type;

    public Book() {
    }
    ...
}
```

Sérialisation XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
    <dateOfCreation>2013-11-17T00:00:00+01:00</dateOfCreation>
    <id>1</id>
    <login>iron</login>
    <name>Anthony</name>
    <password>123456</password>
    <surname>Stark</surname>
</user>
```

Résultat XML

Respecte
Schéma XSD

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<xss:schema version="1.0"
xmlns:xss="http://www.w3.org/2001/
XMLSchema">
    <xss:element name="user" type="user"/>
    <xss:complexType name="user">
```

Marshalling

Unmarshalling

```
@XmlRootElement
public class User {
    private Long id;
    private String login;
    private String password;
    private String name;
    private String surname;
    private Date dateOfCreation;
    public User() {
    }
}
```

instance de Classe annotée

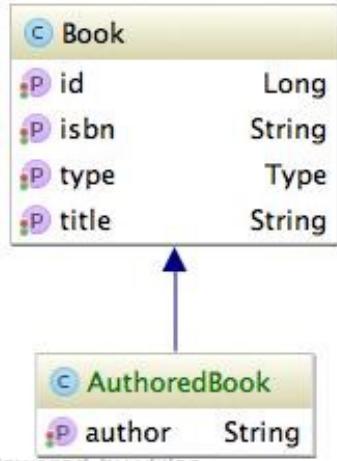
Serialisation XML

```
public class BookGeneration {  
  
    public static void main(String... arg) {  
        Book book = new Book(1L, "Pawn of Prophecy", "XXXX", Type.ROMAN);  
        try {  
            JAXBContext jaxbContext = JAXBContext.newInstance(Book.class);  
            Marshaller marshaller = jaxbContext.createMarshaller();  
            marshaller.setProperty("jaxb.encoding", "UTF-8") ;  
            marshaller.setProperty("jaxb.formatted.output", true) ;  
            marshaller.marshal(book, new File("book.xml"));  
        } catch (JAXBException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Résultat

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:book xmlns:ns2="http://univ-blois.fr/ws/book" id="1">
    <title>Pawn of Prophecy</title>
    <isbn>XXXX</isbn>
    <type>ROMAN</type>
</ns2:book>
```

Et l'héritage ?

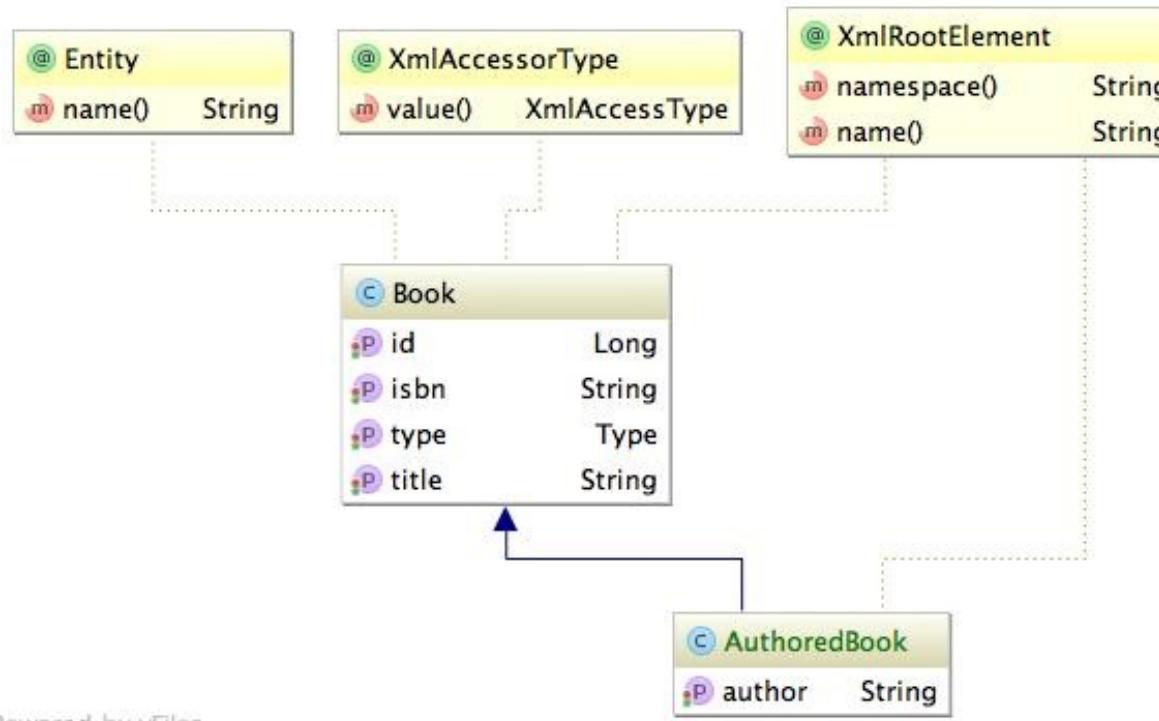


A UML class diagram illustrating inheritance. A box labeled 'Book' contains attributes: id (Long), isbn (String), type (Type), and title (String). An arrow points from a box labeled 'AuthoredBook' to the 'Book' box, indicating that 'AuthoredBook' inherits from 'Book'. The 'AuthoredBook' box contains one attribute: author (String).

```
public class AuthoredBook extends Book {  
    private String author;  
  
    public AuthoredBook() {}  
  
    public AuthoredBook(Long id, String title, String isbn, Type type, String author) {  
        super(id, title, isbn, type);  
        this.author = author;  
    }  
  
    public String getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(String author) {  
        this.author = author;  
    }  
}
```

Powered by yFiles

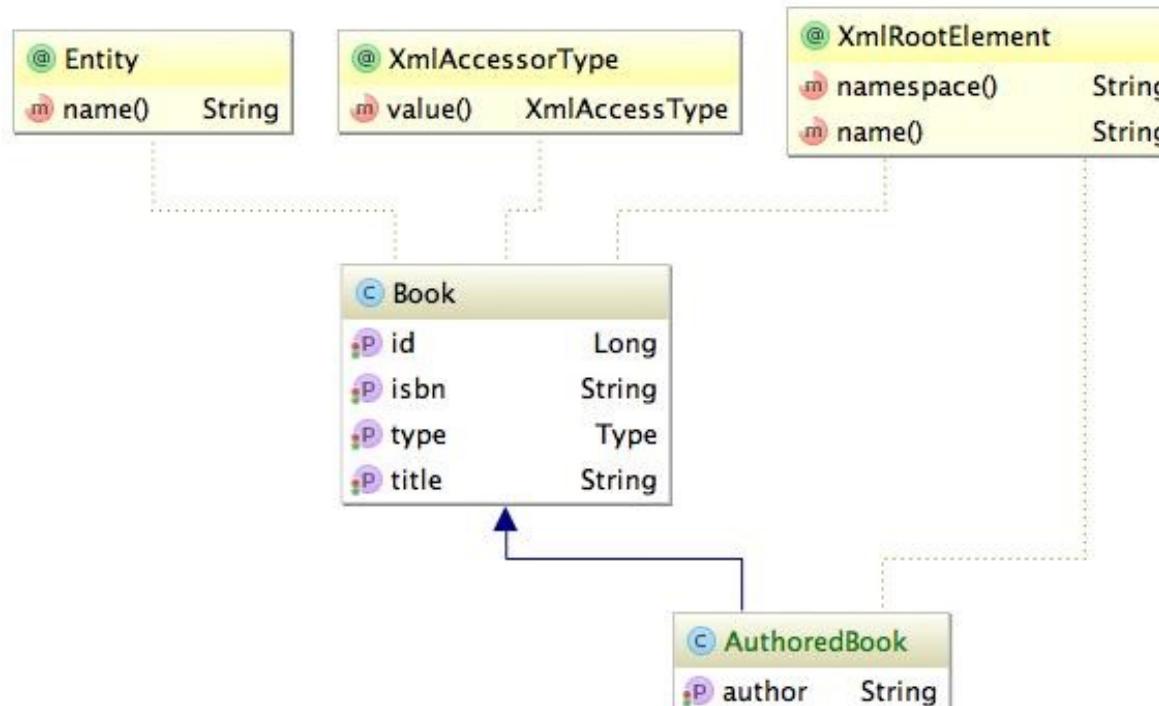
Et l'héritage ?



Powered by yFiles

```
@XmlRootElement
public class AuthoredBook extends Book {
    ...
}
```

Et l'héritage ?

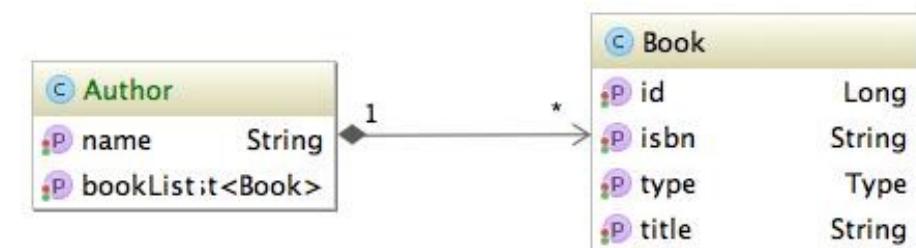


Powered by yFiles

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<authoredBook xmlns:ns2="http://univ-blois.fr/ws/book" id="1">
    <title>Pawn of Prophecy</title>
    <isbn>XXXX</isbn>
    <type>ROMAN</type>
    <author>truc</author>
</authoredBook>
```

Gestion des collections

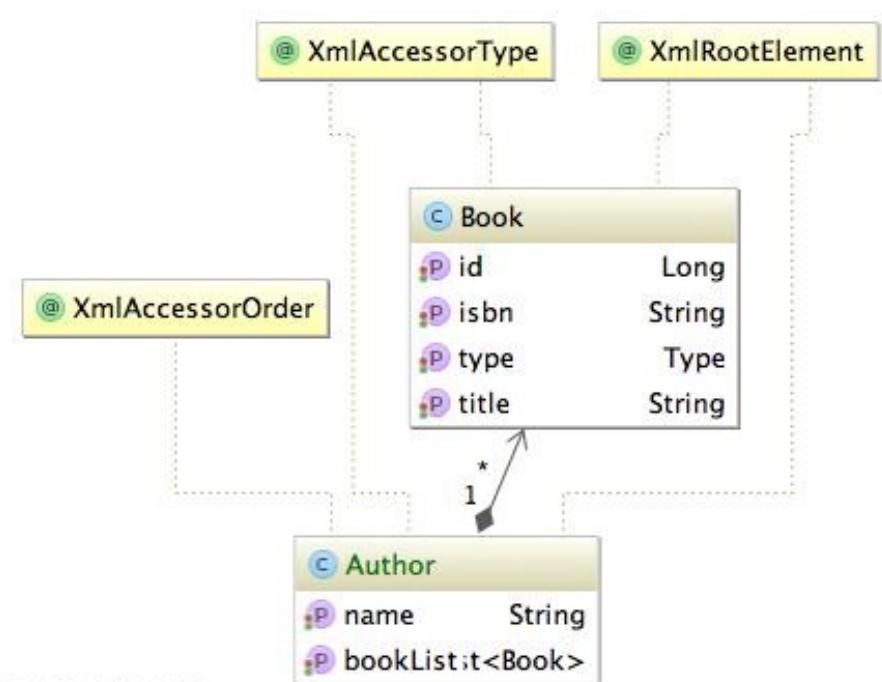
```
public class Author {  
  
    private String name;  
  
    private List<Book> bookList;  
  
    public Author() {}  
  
    public Author(String name) {  
        this.name = name;  
        this.bookList = new ArrayList<>();  
    }  
  
    public void addBook(Book book) {  
        this.bookList.add(book);  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public List<Book> getBookList() {  
        return bookList;  
    }  
  
    public void setBookList(List<Book> bookList) {  
        this.bookList = bookList;  
    }  
}
```



Powered by yFiles

Gestion des collections

```
@XmlRootElement  
@XmlAccessorType(XmlAccessType.FIELD)  
@XmlAccessorOrder(XmlAccess0rder.ALPHABETICAL)  
public class Author {  
  
    private String name;  
  
    private List<Book> bookList;  
  
    public Author() {  
    }  
  
    public Author(String name) {  
        this.name = name;  
        this.bookList = new ArrayList<>();  
    }  
  
    ...  
  
}  
  
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<author xmlns:ns2="http://univ-blois.fr/ws/book">  
    <bookList id="1">  
        <title>Pawn of Prophecy</title>  
        <isbn>XXXX</isbn>  
        <type>ROMAN</type>  
    </bookList>  
    <bookList id="2">  
        <title>Queen of Sorcery</title>  
        <isbn>XXXX</isbn>  
        <type>ROMAN</type>  
    </bookList>  
    <name>David Eddings</name>  
</author>
```

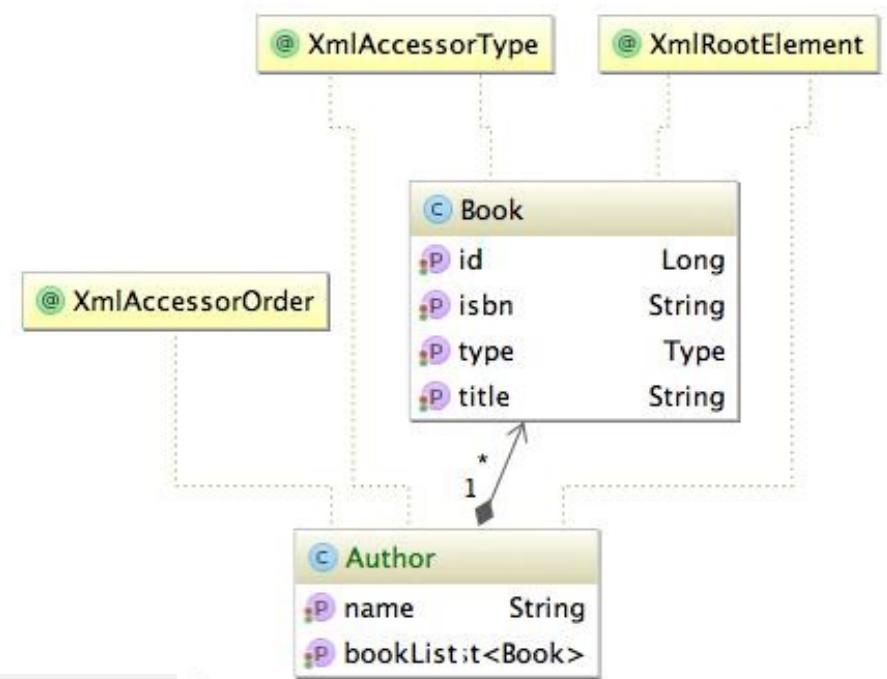


Powered by yFiles

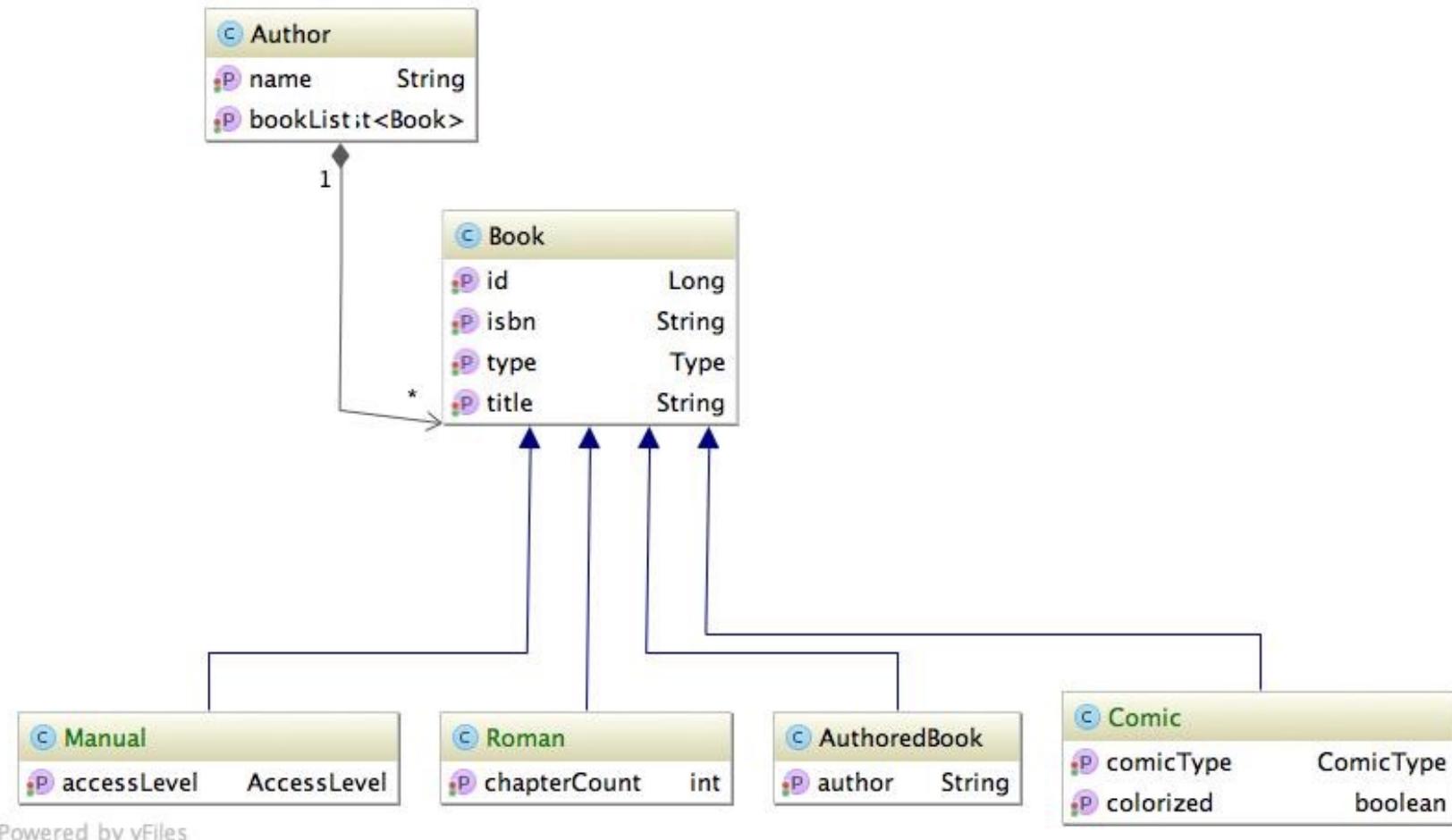
Gestion des collections

```
@XmlRootElement  
@XmlAccessorType(XmlAccessType.FIELD)  
@XmlAccessorOrder(XmlAccessOrder.ALPHABETICAL)  
public class Author {  
  
    private String name;  
  
    @XmlElementWrapper(name = "books")  
    @XmlElement(name = "book")  
    private List<Book> bookList;  
  
    public Author() {  
    }  
  
    ...  
}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<author xmlns:ns2="http://univ-blois.fr/ws/book">  
    <books>  
        <book id="1">  
            <title>Pawn of Prophecy</title>  
            <isbn>XXXX</isbn>  
            <type>ROMAN</type>  
        </book>  
        <book id="2">  
            <title>Queen of Sorcery</title>  
            <isbn>XXXX</isbn>  
            <type>ROMAN</type>  
        </book>  
    </books>  
    <name>David Eddings</name>  
</author>
```



Gestion des collections et polymorphisme



Gestion des collections et polymorphisme

```
@XmlRootElement  
@XmlAccessorType(XmlAccessType.FIELD)  
@XmlAccessorOrder(XmlAccessOrder.ALPHABETICAL)  
public class Author {  
  
    private String name;  
  
    @XmlElementWrapper(name = "books")  
    @XmlElements({  
        @XmlElement(name = "comic", type = Comic.class),  
        @XmlElement(name = "Manual", type = Manual.class),  
        @XmlElement(name = "Roman", type = Roman.class)  
    })  
    private List<Book> bookList;  
  
    public Author() {  
    }  
  
    public Author(String name) {  
        this.name = name;  
        this.bookList = new ArrayList<>();  
    }  
  
    public void addBook(Book book) {  
        this.bookList.add(book);  
    }  
  
    ...  
}
```

Gestion des collections et polymorphisme

```
Author author = new Author("David Eddings");
author.addBook(new Roman(1L, "Pawn of Prophecy", "XXXX", Type.ROMAN, 10));
author.addBook(new Roman(2L, "Queen of Sorcery", "XXXX", Type.ROMAN, 10));
author.addBook(new Comic(3L, "First comic", "YYYY", Type.COMICS, ComicType.TRADITIONAL, true));
author.addBook(new Comic(4L, "First manga", "YYYY", Type.COMICS, ComicType.MANGA, true));
author.addBook(new Manual(4L, "Maitriser le poisson de verre", "YYYY", Type.EDUCATION, AccessLevel-BEGINNER));
author.addBook(new Manual(4L, "Le papillon en mode expert", "YYYY", Type.EDUCATION, AccessLevel-EXPERT));
```

Gestion des collections et polymorphisme

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<author xmlns:ns2="http://univ-blois.fr/ws/book">
  <books>
    <Roman id="1">
      <title>Pawn of Prophecy</title>
      <isbn>XXXX</isbn>
      <type>40</type>
      <chapterCount>10</chapterCount>
    </Roman>
    <Roman id="2">
      <title>Queen of Sorcery</title>
      <isbn>XXXX</isbn>
      <type>40</type>
      <chapterCount>10</chapterCount>
    </Roman>
    <comic id="3">
      <title>First comic</title>
      <isbn>YYYY</isbn>
      <type>70</type>
      <comicType>TRADITIONAL</comicType>
      <colorized>true</colorized>
    </comic>
  </books>
  <name>David Eddings</name>
</author>
```

```
<comic id="4">
  <title>First manga</title>
  <isbn>YYYY</isbn>
  <type>70</type>
  <comicType>MANGA</comicType>
  <colorized>true</colorized>
</comic>
<Manual id="4">
  <title>Maitriser le poisson de verre</title>
  <isbn>YYYY</isbn>
  <type>50</type>
  <accessLevel>BEGINNER</accessLevel>
</Manual>
<Manual id="4">
  <title>Le papillon en mode expert</title>
  <isbn>YYYY</isbn>
  <type>50</type>
  <accessLevel>EXPERT</accessLevel>
</Manual>
</books>
```

Cas @XmlSchema

```
@XmlSchema(  
    xmlns = {  
        @XmlNs(prefix="univ", namespaceURI="http://univ-blois.fr/ws/"),  
        @XmlNs(prefix="xsd", namespaceURI="http://www.w3.org/2001/XMLSchema")  
    },  
    namespace = "http://univ-blois.fr/ws/",  
    elementFormDefault= XmlNsForm.QUALIFIED,  
    attributeFormDefault=XmlNsForm.UNQUALIFIED  
)  
package fr.univ.blois.siad.m2.ws.jaxb;  
  
import javax.xml.bind.annotation.XmlNs;  
import javax.xml.bind.annotation.XmlNsForm;  
import javax.xml.bind.annotation.XmlSchema;
```

Cas @XmlSchema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<univ:author xmlns:univ="http://univ-blois.fr/ws/" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <univ:books>
    <univ:Roman id="1">
      <univ:title>Pawn of Prophecy</univ:title>
      <univ:isbn>XXXX</univ:isbn>
      <univ:type>40</univ:type>
      <univ:chapterCount>10</univ:chapterCount>
    </univ:Roman>
    <univ:Roman id="2">
      <univ:title>Queen of Sorcery</univ:title>
      <univ:isbn>XXXX</univ:isbn>
      <univ:type>40</univ:type>
      <univ:chapterCount>10</univ:chapterCount>
    </univ:Roman>
    <univ:comic id="3">
      <univ:title>First comic</univ:title>
      <univ:isbn>YYYY</univ:isbn>
      <univ:type>70</univ:type>
      <univ:comicType>TRADITIONAL</univ:comicType>
      <univ:colorized>true</univ:colorized>
    </univ:comic>
    <univ:comic id="4">
      <univ:title>First manga</univ:title>
      <univ:isbn>YYYY</univ:isbn>
      <univ:type>70</univ:type>
      <univ:comicType>MANGA</univ:comicType>
      <univ:colorized>true</univ:colorized>
    </univ:comic>
    <univ:Manual id="4">
      <univ:title>Maitriser le poisson de verre</univ:title>
      <univ:isbn>YYYY</univ:isbn>
      <univ:type>50</univ:type>
      <univ:accessLevel>BEGINNER</univ:accessLevel>
    </univ:Manual>
    <univ:Manual id="4">
      <univ:title>Le papillon en mode expert</univ:title>
      <univ:isbn>YYYY</univ:isbn>
      <univ:type>50</univ:type>
      <univ:accessLevel>EXPERT</univ:accessLevel>
    </univ:Manual>
  </univ:books>
  <univ:name>David Eddings</univ:name>
</univ:author>
```

JAXB les annotations

Annotation	Description
<code>@XmlAccessorType</code>	<code>FIELD, NONE, PROPERTY, PUBLIC_MEMBER</code>
<code>@XmlRootElement</code>	mappe une classe comme racine d'un XML
<code>@XmlAttribute</code>	mappe comme un attribut XML
<code>@XmlElement</code>	mappe comme un élément XML
<code>@XmlTransient</code>	empêche le mapping XML
<code>@XmlList</code>	Mappe une propriété comme une liste XML
<code>@XmlType</code>	Mappe une classe comme type complexe XML

`@XmlElement`, `@XmlAttribute`, `@XmlAttribute`, `@XmlAttribute`, `@XmlAttribute`, `@XmlAttribute`, `@XmlAttribute`,
`@XmlAttribute`, `@XmlAttribute`, `@XmlAttribute`, `@XmlAccessorType`, `@XmlAccessorType`

Schéma & génération

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xss:schema version="1.0" xmlns:xss="http://www.w3.org/2001/XMLSchema">
  <xss:element name="user" type="user"/>
  <xss:complexType name="user">
    <xss:sequence>
      <xss:element name="dateOfCreation" type="xss:dateTime" minOccurs="0"/>
      <xss:element name="id" type="xss:long" minOccurs="0"/>
      <xss:element name="login" type="xss:string" minOccurs="0"/>
      <xss:element name="name" type="xss:string" minOccurs="0"/>
      <xss:element name="password" type="xss:string" minOccurs="0"/>
      <xss:element name="surname" type="xss:string" minOccurs="0"/>
    </xss:sequence>
  </xss:complexType>
</xss:schema>
```

Schéma XSD

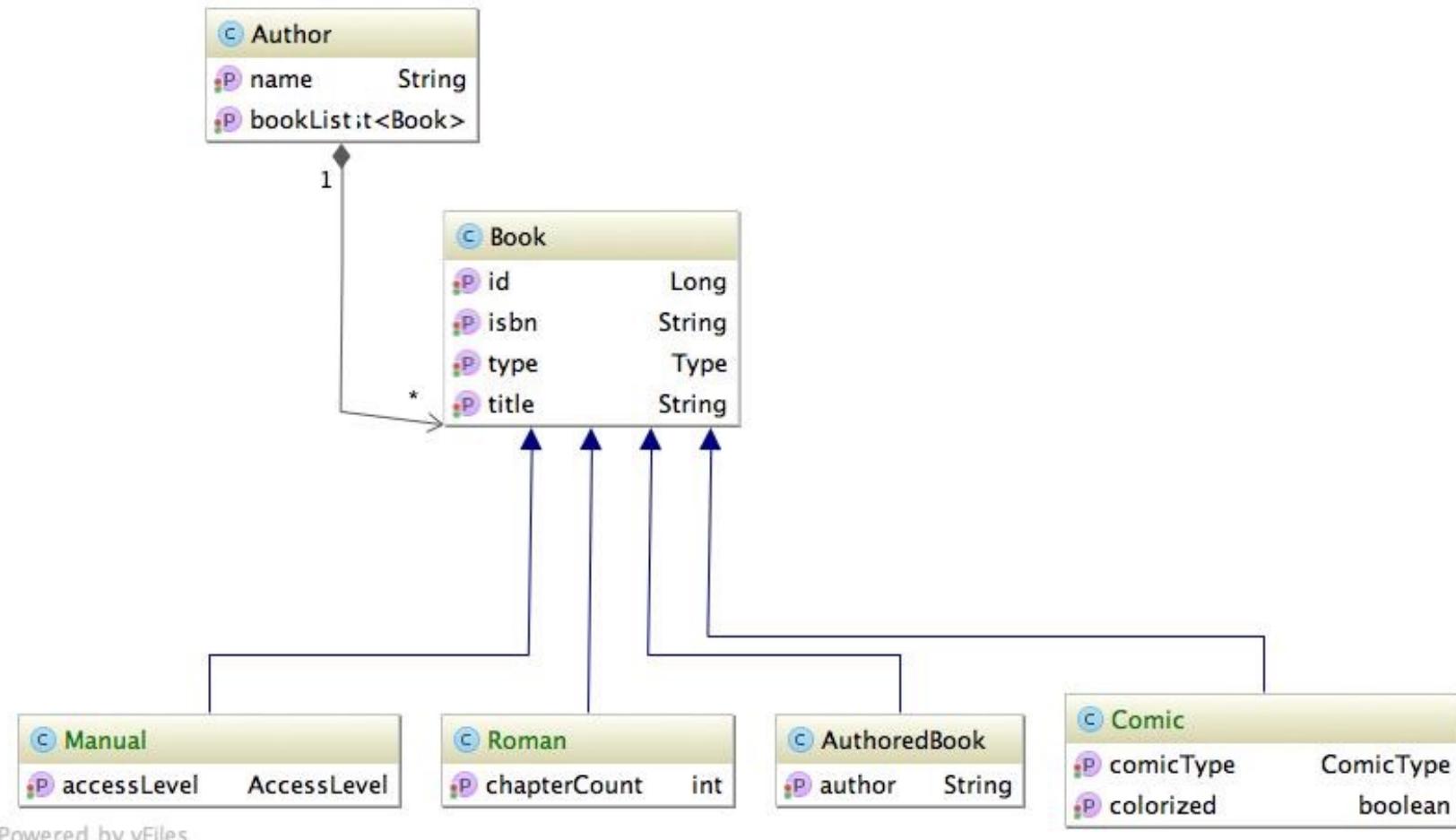
Génération du schéma via schemagen

```
@XmlRootElement
public class User {
    private Long id;
    private String login;
    private String password;
    private String name;
    private String surname;
    private Date dateOfCreation;
    public User() {
    }
}
```

Classe annotée

Compilation du schéma via XJC

Gestion des collections et polymorphisme



Génération des schéma

- Compilation via Maven : mvn clean compile
- Utilisation de schemagen
- schemagen -cp -d gen/ ...
...target/classes/ fr.univ.blois.siad.m2.ws.jaxb.Author

Génération des schéma

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0" targetNamespace="http://univ-blois.fr/ws/">
  xmlns:univ="http://univ-blois.fr/ws/" xmlns:tns="http://univ-blois.fr/ws/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="author" type="tns:author"/>

    <xsd:element name="book" type="tns:book"/>

    <xsd:complexType name="author">
      <xsd:sequence>
        <xsd:element name="books" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="comic" type="tns:comic"/>
                <xsd:element name="Manual" type="tns:manual"/>
                <xsd:element name="Roman" type="tns:roman"/>
              </xsd:choice>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="name" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="comic">
      <xsd:complexContent>
        <xsd:extension base="tns:book">
          <xsd:sequence>
            <xsd:element name="comicType" type="tns:comicType" minOccurs="0"/>
            <xsd:element name="colorized" type="xsd:boolean"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="book">
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string" minOccurs="0"/>
        <xsd:element name="isbn" type="xsd:string" minOccurs="0"/>
        <xsd:element name="type" type="tns:type" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:long" use="required"/>
    </xsd:complexType>
```

Génération des schéma

```
<xsd:complexType name="manual">
  <xsd:complexContent>
    <xsd:extension base="tns:book">
      <xsd:sequence>
        <xsd:element name="accessLevel" type="tns:accessLevel" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="roman">
  <xsd:complexContent>
    <xsd:extension base="tns:book">
      <xsd:sequence>
        <xsd:element name="chapterCount" type="xsd:int"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="comicType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="MANGA"/>
    <xsd:enumeration value="COMIC"/>
    <xsd:enumeration value="TRADITIONAL"/>
  </xsd:restriction>
</xsd:simpleType>

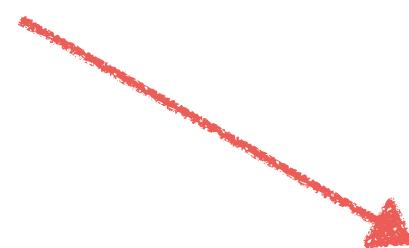
<xsd:simpleType name="type">
  <xsd:restriction base="xsd:int">
    <xsd:enumeration value="10"/>
    <xsd:enumeration value="40"/>
    <xsd:enumeration value="50"/>
    <xsd:enumeration value="60"/>
    <xsd:enumeration value="20"/>
    <xsd:enumeration value="70"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="accessLevel">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="BEGINNER"/>
    <xsd:enumeration value="EXPERT"/>
    <xsd:enumeration value="INTERMEDIATE"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

Génération des classes

Utilisation de l'utilitaire xjc

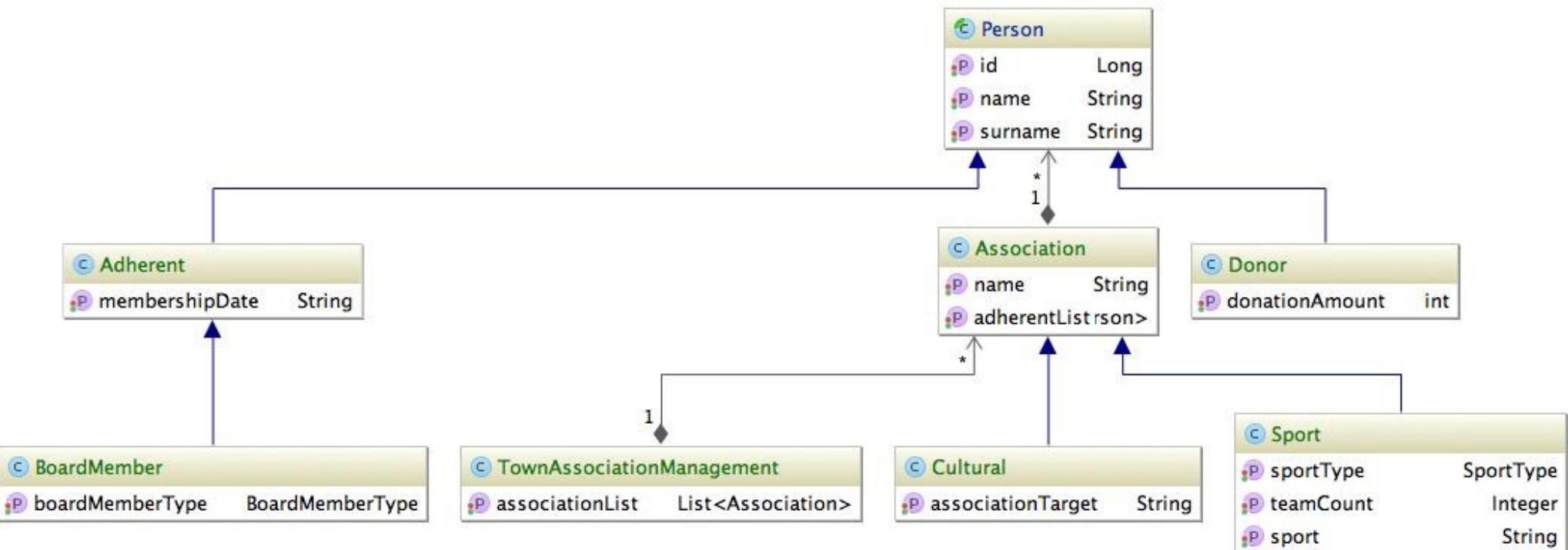
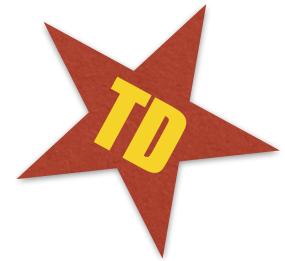
xjc schema1.xsd



Génération de 9 classes

- Author.java
- Book.java
- Comic.java
- Manual.java
- Roman.java
- Type.java
- AccessLevel.java
- ComicType.java
- Une factory

TD 1



Powered by yFiles

Rappelez vous :

- un bean a toujours un constructeur vide
- Le code est en anglais
- les commentaires et javadoc en français

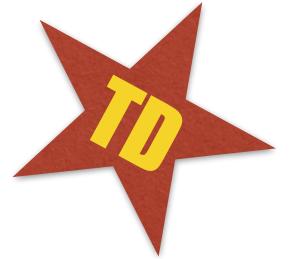
TD 1



A partir de la classe de test TownAssociationTest et de la méthode principale...

- Suivez les demandes exprimées en commentaire
 - Dans la classe de test et dans chaque bean
- Sur la classe Person
 - Ajouter un id qui sera obligatoirement en attribut
- Sur la classe BoardMember
 - Le type de poste devra être en attribut
- La classe association devra presenter un champs nombre d'adhérents non sérialisé
- Créer une classe Address, chaque personne devra la presenter
- Créer un schéma pour chaque package avec comme namespace
 - base (fr.univ.blois.siad.m2.ws.jaxb.td01) : town
 - Person : adh
 - Club : clb
- Pour l'énum SportType, il devra être serialisé sous forme d'entier
- Générer le fichier
- *Si il vous reste du temps, amusez vous à générer les schémas*

Web Services SOAP



SOAP Initialisation

- Pour ce TD, nous allons utiliser NetBeans
 - Utilisation du serveur Glassfish intégré
 - Utilisation du serveur de base de données Derby
- Récupérer les dernières sources du projet
 - cf. instruction d'initialisation
 - mvn clean compile
- Il convient de l'initialiser
 - Lancer Netbeans
 - Création de la base
 - Lancer le serveur Derby
 - créer une base activity
 - user : root
 - password : root
 - Importer le projet siad_m2_ws
 - ouvrir le module soap
 - le lancer
 - Voir la page d'index qui s'affiche

Spécifications

Spécifications	Rôle
JAX-WS 2.0	Spécifications
JAXB 2.2.3	Binding XML (utilisé pour le WSDL)
WS-BP 1.1	Interopérabilité .NET
WS-Metadata	Metadata approche pour définition des WS
JAX-RPC 1.1	Compatibilité avec J2EE 1.4 WS

Technologie et protocoles

- UDDI
- WSDL
- SOAP
- Protocole de transport
- XML

UDDI

- Annuaire de web-services
- fournit les informations sur les web- services sous forme de document WSDL

WSDL

- Web Service Description Language
- définit le type de message, le port, le protocole, les opérations, l'adresse...
- Fichier XML

SOAP

- Simple Object Access Protocol
- Protocole standard de communication des web services
- Fichier XML

Protocole de transport

- Moyen d'échange des messages
 - HTTP souvent
 - HTTPS,TCP/IP, SMTP, FTP, JMS...

XML

- Permet l'indépendance et l'interopérabilité des web services
- dans la définition (WSDL) et l'échange (SOAP)
- xsd permet la validation des messages

Différentes implémentations ?

- Choix entre 2 type d'implémentations
 - Classe standard Java (POJO)
 - EJB Stateless

Différentes implémentation ?

Fonctionnalité	Java WS	EJB
POJO	OUI	OUI
DI de ressources, PU, ...	OUI	OUI
Lifecycle methods	OUI	OUI
Transaction déclarative	NON	OUI
Sécurité déclarative	NON	OUI
Annotation Processing dans un APT externe	OUI	non pour la plupart des containers EJB
Fonctionne dans un container web (Tomcat)	OUI	NON

Contraintes d'implémentation

- Annotation @WebService ou équivalent XML dans le descripteur de déploiement
- classe publique, non final et non abstract
- un constructeur public par défaut
- pas de définition de finalize()
- objet stateless

Approche de développement

- Bottom up : de l'implémentation au WSDL
- Top Down : du WSDL à l'implémentation (contract first)
- Meet in the middle : Rattache le WSDL à l'implémentation

Exemples...

```
@Stateless  
@LocalBean  
@WebService  
public class UserServices {  
  
    ...  
  
    public String getUserName(long userId) {  
        User user = entityManager.find(User.class, userId);  
        return user.getName();  
    }  
  
    ...  
}  
}
```

@WebService

- S'utilise sur l'interface ou le bean (le container génère alors l'interface)
- Toutes les méthodes publiques sont exposées

@WebService

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = {ElementType.TYPE})
public @interface WebService {
    public String name() default "";
    public String targetNamespace() default "";
    public String serviceName() default "";
    public String portName() default "";
    public String wsdlLocation() default "";
    public String endpointInterface() default "";
}
```

@WebService

- name : nom du WS
- targetNamespace : précise le namespace du WSDL
- serviceName : quand on annote le bean
- wsdlLocation : précise l'emplacement du WSDL (pas de génération)
- endpointInterface : précise le nom du Service Endpoint Interface
- portname : nom du port

Configuration

- plusieurs styles :
 - DOCUMENT
 - RPC
- style de message :
 - LITERAL
 - ENCODED
 - Style de paramètres
 - BARE
 - WRAPPED

@SOAPBinding

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = {ElementType.TYPE, ElementType.METHOD})
public @interface SOAPBinding {
    public enum ParameterStyle {
        BARE, WRAPPED;
    }
    public enum Style {
        DOCUMENT, RPC;
    }
    public enum Use {
        LITERAL, ENCODED;
    }
    public Style style() default DOCUMENT;
    public Use use() default LITERAL;
    public ParameterStyle parameterStyle() default WRAPPED;
}
```

@WebMethod

- Permet de choisir les méthodes exposées.

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = {ElementType.METHOD})
public @interface WebMethod {

    public String operationName() default "";
    public String action() default "";
    public boolean exclude() default false;
}
```

@WebMethod

- `operationName` : operation name du WSDL
- `action` : soap:operation du WSDL
- `exclude` : pour empêcher l'exposition

@WebParam

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = {ElementType.PARAMETER})
public @interface WebParam {
    public enum Mode {
        IN, OUT, INOUT;
    }
    public String name() default "";
    public String partName() default "";
    public String targetNamespace() default "";
    public Mode mode() default IN;
    public boolean header() default false;
}
```

@WebParam

- name : name du paramètre dans le message du WSDL
- targetNamespace : configuration du WSDL
- mode : mode d'échange du paramètre
- header : true si paramètre dans le header et pas le message body
- partName : name element du WSDL

@OneWay

- se positionne sur une méthode
- quand il n'y a pas de valeur de retour

@HandlerChain

- Equivalent des Intercepteurs des EJB
- WebServiceContext.getMessageContext() équivalent de InvocationContext.getContextData()

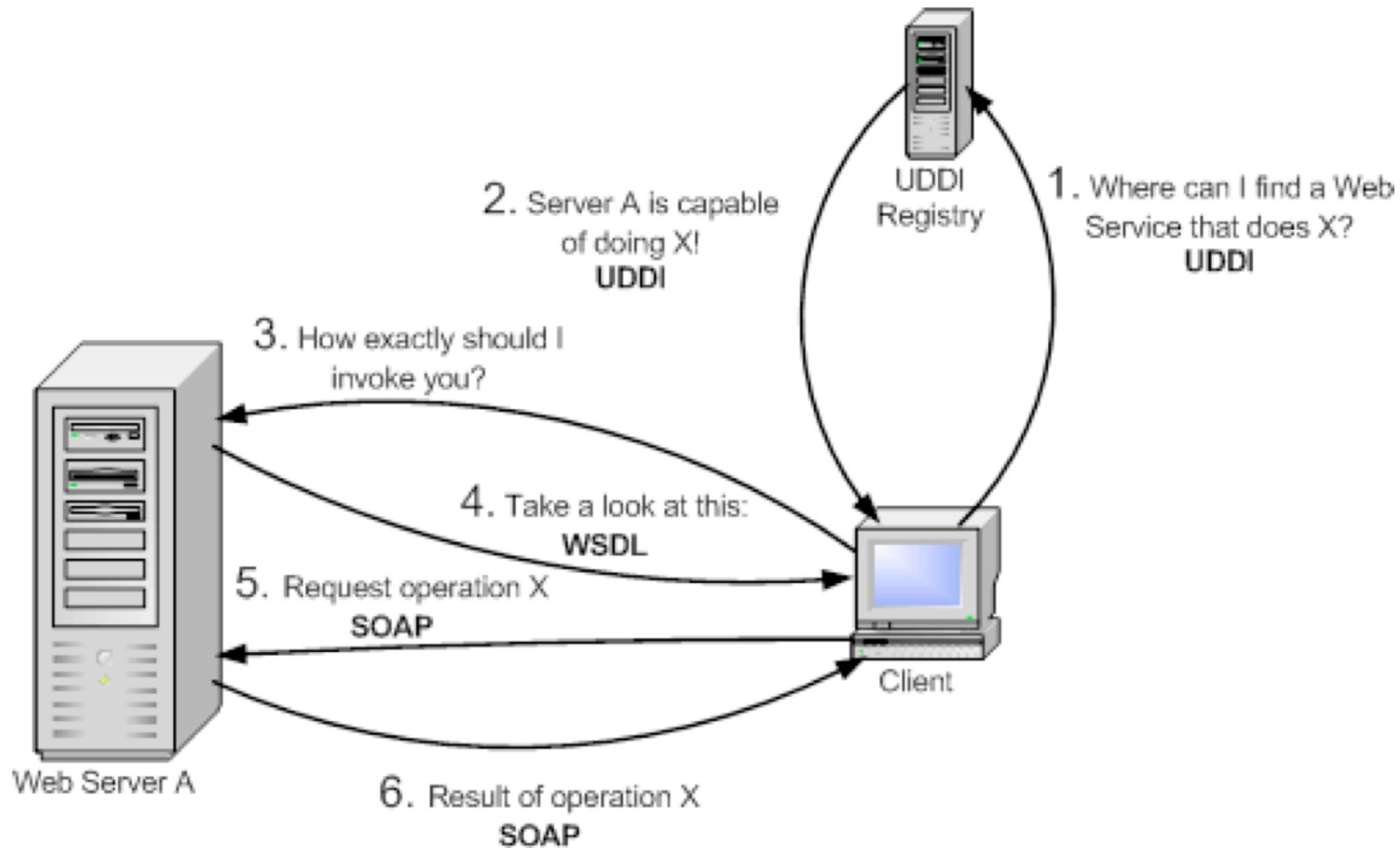
Cycle de vie et injection

- Comme pour les EJB, entité : `@PostConstruct` & `@PreDestroy`
- Injection de dépendance possible (`EntityManager` par exemple)

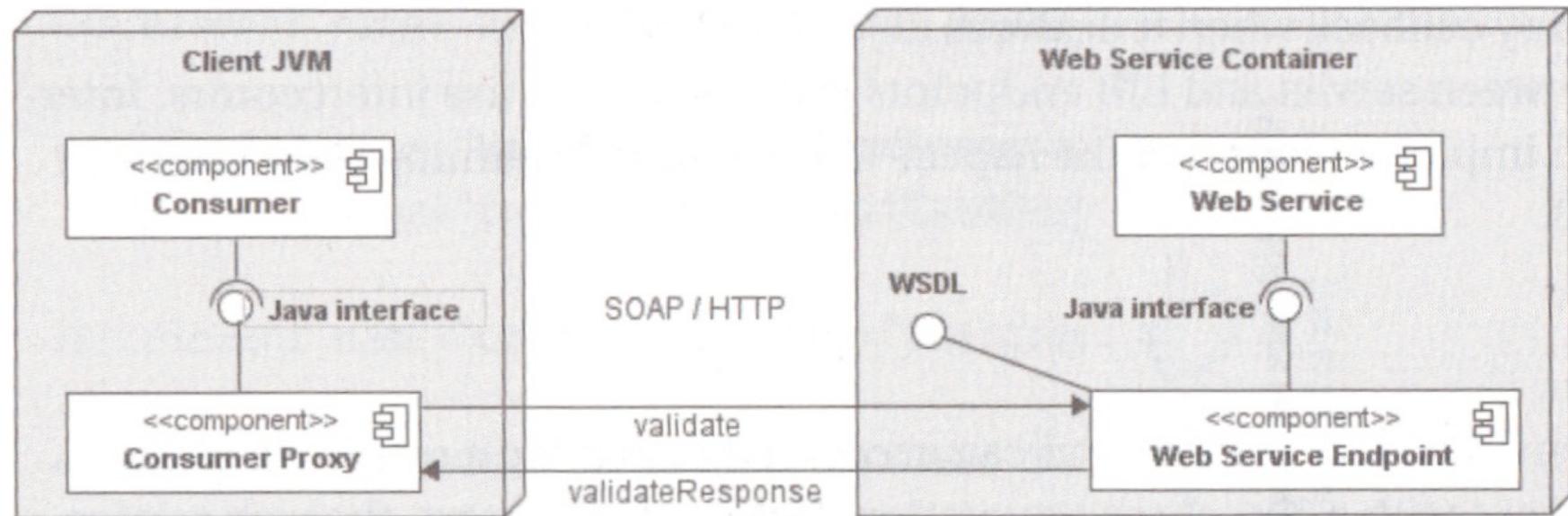
WebServiceContext

- Injection par @Resource
- méthodes :
 - getMessageContext
 - getUserPrincipal
 - isUserInRole
 - getEndPointReference

Appeler un WebService



Appeler un WebService



Beginning Java EE 6 Platform with GlassFish 3, Antonio Goncalves, Apress, p. 418

Appeler un WebService

- Utilisation de
 - wsgen : génération du WSDL
 - wsimport : génération du SEI et des classes à partir du WSDL
- Récupération du SEI par programmation ou par injection, puis du proxy et invocation

Référence par programmation

- HelloWSService service = new HelloWSService();
- HelloWS hello = service.getPort(HelloWS.class);
- String result = hello.sayHello(@WebServiceB);

Référence par annotation

```
@WebServiceRef  
HelloWSService service ;  
.....  
HelloWS hello = service.getPort(HelloWS.class);  
String result = hello.sayHello(AWebServiceB);
```

@WebServiceRef

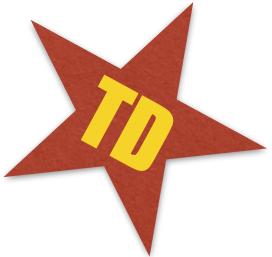
```
@Target({TYPE})
public @interface WebServiceRef {
    String name() default >?;
    String mappedName() default >?;
    class type() default java.lang.Object.class;
    class value() default java.lang.Object.class;
    String wsdlLocation() default >?;
};
```

@WebServiceRef

- name : nom JNDI lié à java:comp/env/<name>
- mappedName : nom JNDI vendeur spécifique
- type : type de la resource
- value : classe du service (extends javax.xml.ws.Service)
- wsdlLocation : URL du WSDL

WS-*

- Quelques spécifications complémentaires (pas toutes standardisées) :
 - WS-Security
 - WS-Reliability
 - WS-Transaction
 - WS-Addressing



TD 2

- Avec la classe PingWS
 - La transformer en WS
 - Ajouter une méthode qui renvoi une chaîne en majuscule
- Créer un service Web à destination des utilisateurs
 - Crédit d'un utilisateur (qui renvoie l'utilisateur créé)
 - Liste des utilisateurs
- Créer un service Web à destination des activités
 - Renvoyer la liste des activités
 - Pour une activité
 - Renvoyer la liste des adhésions
 - Ajouter une adhésion pour un utilisateur

Evidemment, pour tout cela, il convient de tester les méthodes...

Webservice en résumé...

