

Les services Web Java EE

François Robert

francois.robert@shipstone.org



1

Le plan !

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

3

De quoi parle t'on ?

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

2

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

4



Initialisation



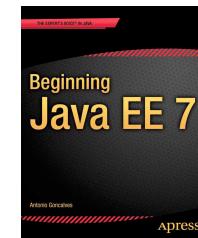
- Lancer la VM
 - login : siad - password : **javaeesiad**
- Ouvrez une console et rendez vous dans le repertoire de dev
 - Créez un repertoire workspaces et placez vous dedans
 - Astuce :`mkdir workspaces && cd $_`
- Il s'agit maintenant de cloner le repository de source (hébergé par github)
 - Exécutez `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) :

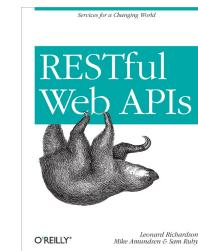
- `git pull origin`
- `mvn clean compile`

5

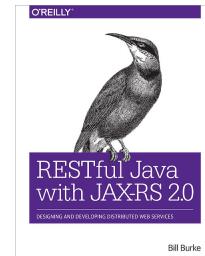
Références



Beginning Java EE 7
Antonio Goncalves
aPress 2013



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



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

6

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

<http://www.oracle.com/technetwork/java/javaee/tech/index.html>

7

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							

8

Java EE 7

Les principaux composants

Vu en MI

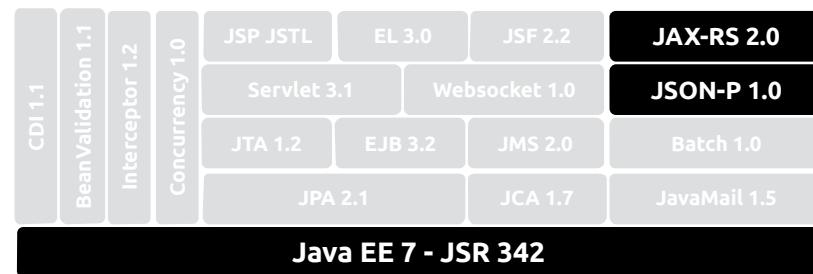


9

Java EE 7

Les principaux composants

Les web Services



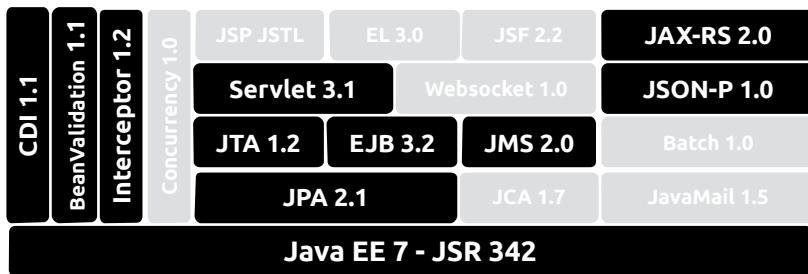
10

Java EE 7

Les principaux composants

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

C'est évidemment une architecture presque minimale



11

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

12

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

13

XML & JAXB

14

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

15

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;

    private String title;

    private String isbn;

    @Enumerated(EnumType.STRING)
    private Type type;

    public Book() {
    }
}
```

16

@XmlRootElement

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

    String namespace() default "##default";

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

17

@XmlAccessorType

```
@Inherited @Retention(RUNTIME) @Target({PACKAGE, TYPE})
public @interface XmlAccessorType {

    XmlAccessType value() default XmlAccessType.PUBLIC_MEMBER;
}

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

18

@XmlAttribute

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

    String name() default "##default";

    boolean required() default false;

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

19

Le cas des énumérés

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

    EnumType value() default ORDINAL;
}

public enum EnumType {
    ORDINAL,
    STRING
}
```

20

@ 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>

```

21

@ Enumerated

```

@XmlType()
@XmlEnum(Integer.class)
public enum Type {
    @XmlEnumValue("10")
    SCIFI,
    @XmlEnumValue("20")
    POLAR,
    @XmlEnumValue("40")
    ROMAN,
    @XmlEnumValue("50")
    EDUCATION,
    @XmlEnumValue("60")
    INFORMATIQUE,
    @XmlEnumValue("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>

```

22

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;
    @XmlElement(name = "title")
    private String title;
    @XmlElement(name = "isbn")
    private String isbn;

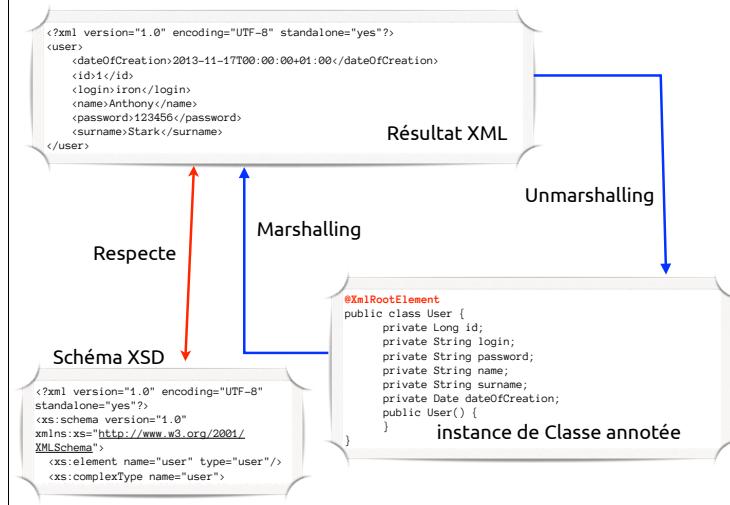
    @Enumerated(EnumType.STRING)
    private Type type;

    public Book() {
    }
...
}

```

23

Sérialisation XML



24

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();
        }
    }
}
```

25

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>
```

26

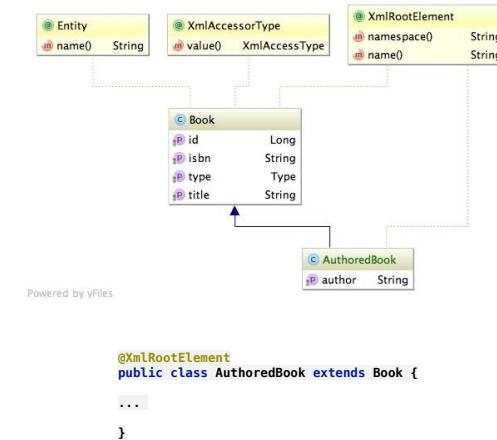
Et l'héritage ?

```
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

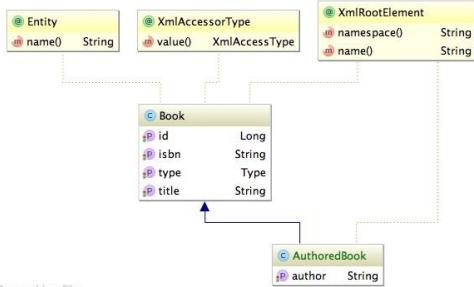
27

Et l'héritage ?



28

Et l'héritage ?



```

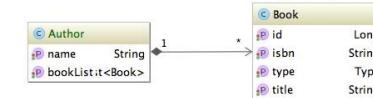
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<authoredBook xmlns:ns2="http://univ-blois.fr/ws/book" id="1">
  <title>Pawn of Prophecy</title>
  <isbn>XXXX</isbn>
  <type>ROMAN</type>
  <author>truc</author>
</authoredBook>
  
```

29

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;
    }
}
  
```



30

Gestion des collections

```

@XmlElement
@XmlAccessorType(XmlAccessType.FIELD)
@XmlAccessorOrder(XmlAccessorOrder.ALPHABETICAL)
public class Author {
    private String name;
    private List<Book> bookList;
    public Author() {
    }
    public Author(String name) {
        this.name = name;
        this.bookList = new ArrayList<>();
    }
    ...
}

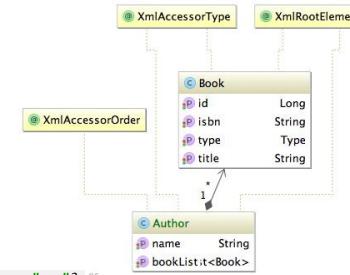
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<author xmlns:ns2="http://univ-blois.fr/ws/book">
  <bookList id="1">
    <book id="1">
      <title>Pawn of Prophecy</title>
      <isbn>XXXX</isbn>
      <type>ROMAN</type>
    </book>
    <book id="2">
      <title>Queen of Sorcery</title>
 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

31

Gestion des collections

```

@XmlElement
@XmlAccessorType(XmlAccessType.FIELD)
@XmlAccessorOrder(XmlAccessorOrder.ALPHABETICAL)
public class Author {
    private String name;
    ...
}
  
```

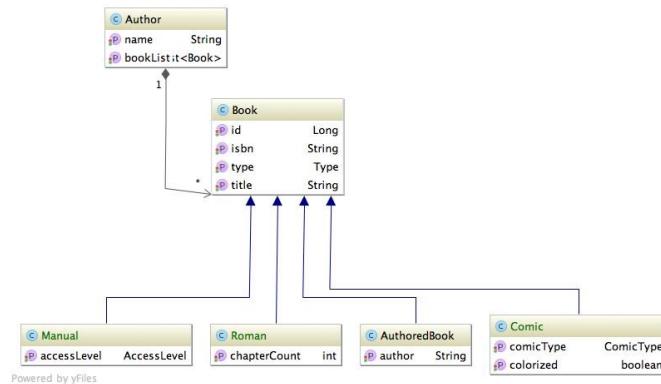


```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<author xmlns:ns2="http://univ-blois.fr/ws/book">
  <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>
  
```

32

Gestion des collections et polymorphisme



33

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, "Maîtriser le poisson de verre", "YYYY", Type.EDUCATION, AccessLevel-BEGINNER));
author.addBook(new Manual(4L, "Le papillon en mode expert", "YYYY", Type.EDUCATION, AccessLevel-BEGINNER));
  
```

35

Gestion des collections et polymorphisme

```

@XmlElementWrapper(name = "books")
@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);
}

...
  
```

34

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>MANGA</comicType>
            <colorized>true</colorized>
        </comic>
        <comic id="4">
            <title>First manga</title>
            <isbn>YYYY</isbn>
            <type>70</type>
            <comicType>MANGA</comicType>
            <colorized>true</colorized>
        </comic>
        <Manual id="4">
            <title>Maîtriser 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>
    <name>David Eddings</name>
</author>
  
```

36

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;

```

37

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="5">
            <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>

```

38

JAXB les annotations

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

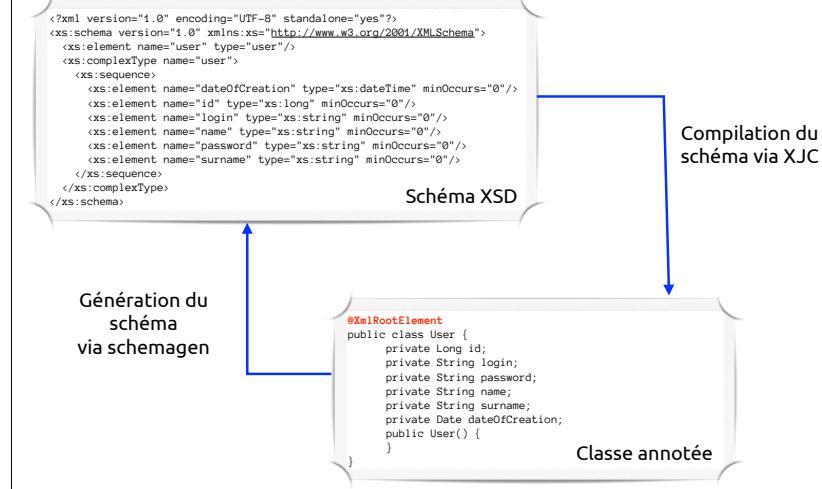
```

@XmlElements, @XmlEnum, @XmlEnumValue, @XmlID, @XmlIDREF, @XmlMimeType,
@XmlNs, @XmlSchema, @XmlValue, @XmlAccessorType, @XmlAccessorOrder

```

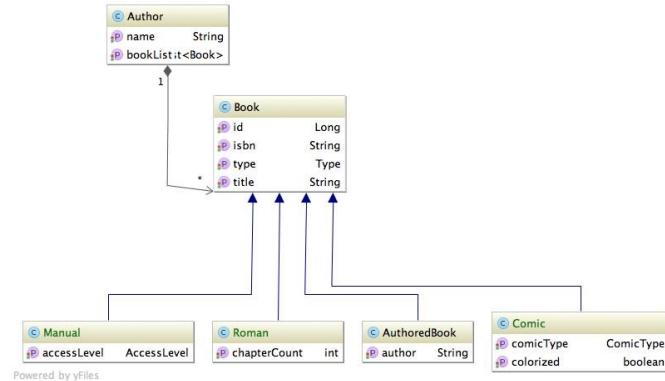
39

Schéma & génération



40

Gestion des collections et polymorphisme



41

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

42

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

43

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="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>
  
```

44

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

45

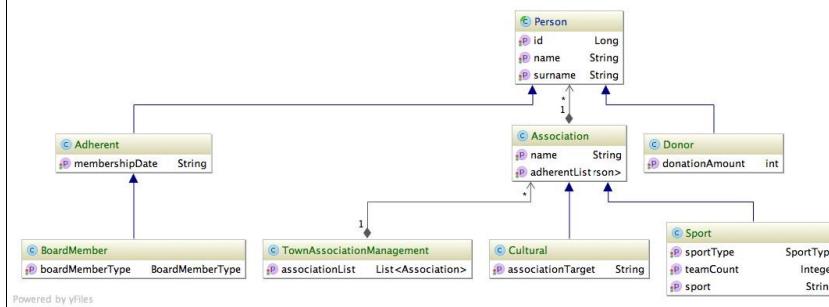
TD 1

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 présenter un champs nombre d'adhérents non sérialisé
- Créer une classe Address, chaque personne devra la présenter
- 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

47

TD 1



Rappelez vous :

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

46

Web Services SOAP

48

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

49

Technologie et protocoles

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

51

Spécifications

Spécifications	Rôle
JAX-WS 2.2	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

50

UDDI

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

52

WSDL

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

53

SOAP

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

54

Protocole de transport

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

55

XML

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

56

Différentes implémentation ?

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

57

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

59

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

58

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

60

Exemples...

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

61

@WebService

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

62

@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 "";  
}
```

63

@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

64

Configuration

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

65

@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;
}
```

66

@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;
}
```

67

@WebMethod

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

68

@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;
}
```

69

@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

70

@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()

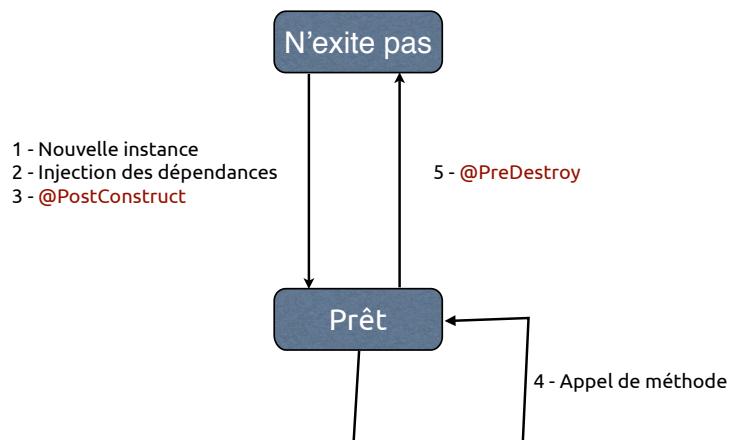
71

Cycle de vie et injection

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

72

Cycle de vie et injection



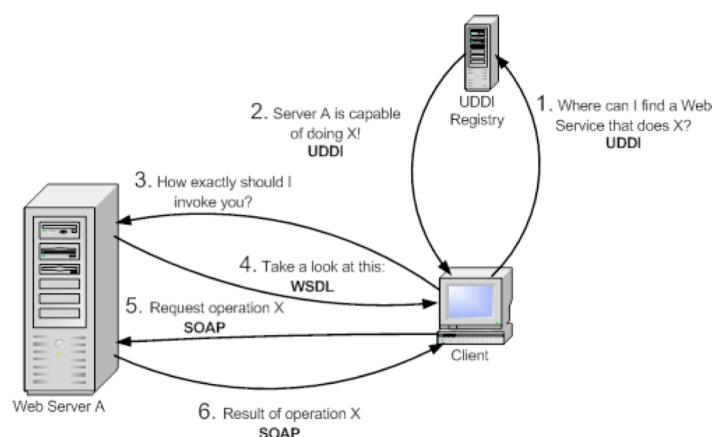
73

WebServiceContext

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

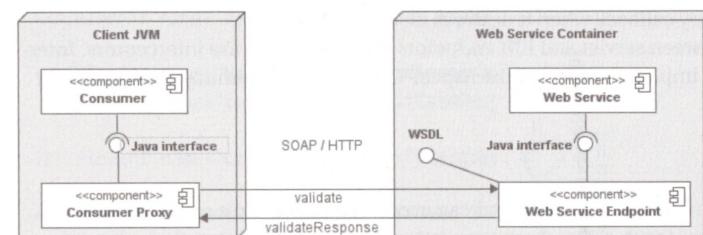
74

Appeler un WebService



75

Appeler un WebService



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

76

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

77

Référence par programmation

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

78

Référence par annotation

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

79

@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 >?;  
};
```

80

@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

81

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...

83

WS-*

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

82

Webservice en résumé...



84

LES SERVICES REST



RULE 1: SOAP IS MUCH MORE
POLITE THAN REST

85

WebService REST



RULE 1: SOAP IS MUCH MORE
POLITE THAN REST

86

REST ?

- REST : Acronyme de REpresentational State Transfert
- Le retour du Web dans le Web Service
- Principes définis dans la thèse de Roy FIEDLING en 2000
- Style d'architecture inspiré de l'architecture du web
- Repose sur HTTP (HyperText Transfert Protocol) et URIs (Uniform Resource Identifiers)
- Standardisé dans Java EE 6 : JSR 311 : JAX-RS

87

REST ?

- REST est
 - Un style d'architecture
 - Une approche pour construire une application
- REST n'est pas
 - un format
 - un protocol
 - un standard

88

REST ?

- Les services web REST sont développés pour mettre en place des architecture orienté ressources
- Les applications respectant les architectures orientées ressources sont appelé RESTful.

89

Ressource ?

- Tout ce que le client peut vouloir
- En général concepts concrets
- Pour une boutique de livres :
 - la liste des livres d'une catégorie
 - le résumé d'un livre
 - la bio d'un auteur

90

Représentation

- Format de présentation de la ressource (texte, JSON, XML, PDF..), la ressource reste sur le serveur
 - La ressource lié à un bon de commande est représenté par un document XML
 - La création de la commande est l'association de la méthode POST et du document XML
- Distinction par URI ou par “Content Negotiation”
- L'état est maintenu par la représentation de la ressource
- De ce fait, c'est le client qui est responsable de l'état.

91

WebService RESTful

- Les services REST sont sans états (Stateless)
 - Toutes les requêtes envoyées au serveurs doivent contenir les informations propre à leur traitement
 - Minimisation des ressources système (pas de session ni de d'état)
- Les services REST fournissent une interface basé sur les verbe HTTP
 - GET, PUT, DELETE & POST
- Les chemin REST sont basé sur les URI (identification unique des ressources)
- ex : <http://www.flickr.com/explore/2013/10/08>

92

URI

- Généralement combinaison d'URL (Uniform Resource Locator) et URN (Uniform Resource Name)
- le plus descriptif possible
- Identifie de manière unique la ressource

`http://shopbook/books/fantasy/belgariad/1`
Ressource de type collection Identifiant primaire de la ressource

- une ressource peut avoir plusieurs URI

`http://shopbook/books/fantasy/belgariad/1`
`http://shopbook/books/fantasy/belgariad/Pawn_of_Prophesy`
Ressource = Pawn of prophecy
`http://shopbook/books/fantasy/belgariad` Tous les livres de la Belgariade
`http://shopbook/books/fantasy` Tous les livres d'heroic fantasy

93

REST & Méthodes

- Une ressource peut subir 4 opérations de base, connues sous le nom de CRUD
 - Create
 - Read / Retrieve
 - Update
 - Delete

94

REST & Méthodes

- REST s'appuie sur les verbes HTTP :
 - Create via POST
 - Read via GET
 - Update via PUT
 - Delete via DELETE
- on n'utilise plus rarement les autres HEAD,TRACE, OPTIONS, CONNECT

95

GET :: Lecture



- Lecture simple
 - doit être implémentée de façon "safe"
 - doit être "idempotent"
-
- **Safe** : se dit d'une méthode qui ne change pas l'état de la ressource
 - **Idempotent** : se dit d'une méthode que l'on peut appeler plusieurs fois sans changer le résultat

96

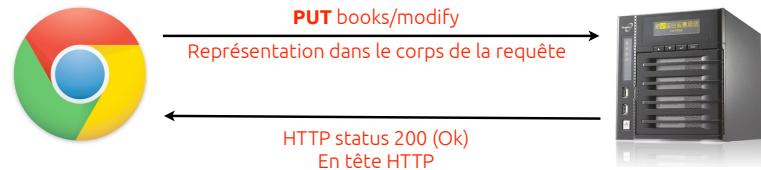
POST :: Création



- pour une représentation (texte, XML...), POST crée un nouvelle ressource comme subalterne d'une ressource principale identifiée par l'URI demandée
- Pas “safe” (modifie l'état)
- Pas idempotent (Post multiples entraînent des doublons)

97

PUT :: Modification



- Met à jour l'état de la ressource correspond à l'URI
- Crée la ressource si elle n'existe pas
- Pas “safe”
- Idempotent

98

Delete :: Suppression



- Supprime la ressource
- Pas “safe”
- Idempotent

99

Et les autres ?

- HEAD : comme GET sans le “body” de la réponse (test la validité de l'URI ou la taille de l'objet)
- TRACE : renvoie la requête reçue (echo)
- OPTIONS : demande les informations sur les options de communications
- CONNECT : utilisé pour demander le changement d'un proxy en tunnel HTTP

100

Négociation de contenu

- dans les headers de la requête HTTP :
 - Accept
 - Accept-Charset
 - Accept-Encoding
 - Accept-Language
 - User-Agent

101

Types de contenu

Content Types

- dans les champs Content-Type et Accept
- 5 catégories : text, image, audio, video, application
- quelques sous-types :
 - text/html, text/plain (type par défaut)
 - image/gif, image/jpeg, image/png
 - text/xml, application/xml
 - application/json (JavaScript Object Notation)

102

Les codes de retour

Ou les codes de statut

- 1xx : Informational : requête reçue, traitement en cours
- 2xx : Success : requête reçue, comprise et traitée
 - 200 : Ok
 - 201 : Crée
- 3xx : Redirection : de nouvelles étapes à faire pour compléter la requête
- 4xx : Client Error : mauvaise syntaxe, la requête ne peut être complétée
 - 404 Not found
- 5xx : Server Error : Le serveur a échoué sur une requête apparemment correcte

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

103

Les codes de retour

Quelques codes

- 200 : Ok
- 201 : Crée
- 301 : déplacé de manière permanente
- 400 : syntaxe erronée
- 401 : non autorisé
- 404 : non trouvé
- 418 : je suis une théière
- 500 : erreur serveur
- etc...

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

104

JEE : Définition service REST

Old flavored **JEE 6**

Un fichier descriptif : WEB-INF/web.xml **Implémentation de référence JSR 311 JAX-RS**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/
    web-app_2_4.xsd">

    <display-name>toolbox</display-name>
    <servlet>
        <servlet-name>calc</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
        <init-param>
            <param-name>com.sun.jersey.config.property.packages</param-name>
            <param-value>fr.univ.blois.jee.rest</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>calc</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Définition de la servlet

Package de base exposant les services

Mapping de la servlet

Composant de l'URL ^{à root} des services REST : blois.univ.fr/rest/....

105

JEE : Définition service REST

Mais c'était avant...

106

JEE : Définition service REST

JEE 7

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
    java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">
</web-app>
```

Rien !

Servlet 3.0

107

JAX-RS

Description de l'application

```
@ApplicationPath("api")
public class SiadRestfulApplication extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        return new HashSet<>(Arrays.asList(Ping.class));
    }
}
```

Optionnel

- Classe étendant la classe Application (implémentation JAX-RS)
- annotation @ApplicationPath avec comme paramètre le root
- Le root peut être vide ("")
- Surcharge de la méthode getClasses pour renvoyer les classe implémentant les services REST
- Si le besoin de limiter l'exposition d'API

108

JAX-RS

Le premier service

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("ping")
public class Ping {

    @GET
    public String pong() {
        return "pong\n";
    }
}
```

- programmation Classe annotée avec `@javax.ws.rs.Path`
- Possibilité d'avoir des capacités d'EJB par ajout de `@Stateless`
- Classe publique, ni finale, ni abstract
- Classe racine de la ressource (annotée `@Path`) doit avoir un constructeur par défaut

109

JAX-RS

Et ça donne quoi ce truc ?

```
[~]$ curl localhost:8080/siad/api/ping/
pong
```

C'est un peu court...

110

JAX-RS

Et avec les headers, cela donne quoi ?

```
[~]$ curl -i localhost:8080/siad/api/ping/
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/octet-stream
Content-Length: 5
Date: Thu, 04 Dec 2014 21:27:34 GMT

pong
```

111

Passage de paramètres

Comment passer des paramètres à votre API

- `@QueryParam`
- `@PathParam`
- `@CookieParam`
- `@FormParam`
- `@HeaderParam`
- `@MatrixParam`

112

@QueryParam

```
@Path("echo")
public class Echo {

    @GET
    public String echo(@QueryParam("value") String value) {
        return value + "\n";
    }

}

[~]$ curl -XGET -i localhost:8080/siad/api/echo?value=test
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/octet-stream
Content-Length: 5
Date: Thu, 04 Dec 2014 21:53:52 GMT

test
```

113

@QueryParam

```
@Path("echo")
public class Echo {

    @GET
    public String echo(@QueryParam("value") String value) {
        return value + "\n";
    }

}

curl -XGET -i localhost:8080/siad/api/echo
?value=un%20truc%20comme%20cela
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/octet-stream
Content-Length: 5
Date: Thu, 04 Dec 2014 21:53:52 GMT

un truc comme cela
```

114

@QueryParam

```
@Path("echo")
public class Echo {

    @GET
    public String echo(@QueryParam("value") String value) {
        return value + "\n";
    }

}
```

Et si on envoi rien ?

```
[~]$ curl -XGET -i localhost:8080/siad/api/echo
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/octet-stream
Content-Length: 5
Date: Thu, 04 Dec 2014 21:53:52 GMT
```

null

Pas tres élégant !

115

@DefaultValue

```
@Path("echo")
public class Echo {

    @GET
    public String echo(@QueryParam("value") @DefaultValue("N/A") String value) {
        return value + "\n";
    }

}
```

```
[~]$ curl -XGET -i localhost:8080/siad/api/echo
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/octet-stream
Content-Length: 5
Date: Thu, 04 Dec 2014 21:53:52 GMT
```

N/A

C'est mieux

116

@DefaultValue

```
@Path("echo")
public class Echo {

    @GET
    @Path("multiple")
    public String echoMultiple(
        @QueryParam("value1") @DefaultValue("Default1") String value1
        , @QueryParam("value2") @DefaultValue("Default2") String value2
        , @QueryParam("value3") @DefaultValue("250") int value3
    ) {
        return String.format("value1 = %s - value2 = %s - value3 = %d\n", value1,
        value2, value3);
    }

}

curl -XGET 'localhost:8080/siad/api/echo/multiple?value3=3&value1="dsds"'
```

value1 = "dsds" - value2 = Default2 - value3 = 3

117

URI paramètre

```
@GET
@Path("{id}")
public String paramUri(
    @PathParam("id") String id
) {
    return String.format("id = %s\n", id);
}
```

```
curl -XGET localhost:8080/siad/api/echo/valeur
id = valeur
```

118

URI paramètre

Combinée ?

```
@GET
@Path("param/{id}/{subId}")
public String parmaSub(
    @PathParam("id") String id
    , @PathParam("subId") String subId
) {
    return String.format("id = %s\nsubId = %s\n", id, subId);
}

curl -XGET localhost:8080/siad/api/echo/param/monId/monsSubId
id = monId
subId = monsubId
```

119

URI Pour finir...

Take all together ?

```
@GET
@Path("complex/{id}/{subId}")
public String complex(
    @PathParam("id") String id
    , @PathParam("subId") String subId
    , @QueryParam("value") @DefaultValue("N/A") String value
) {
    return String.format("id = %s\nsubId = %s\nvalue = %s\n", id, subId, value);
}
```

```
curl -XGET localhost:8080/siad/api/echo/complex/monId/monsSubId
id = monId
subId = monsubId
value = N/A
```

120

URI Pour finir...

Take all together ?

```
@GET  
@Path("complex/{id}/{subId}")  
public String complex(  
    @PathParam("id") String id,  
    @PathParam("subId") String subId,  
    @QueryParam("value") @DefaultValue("N/A") String value  
) {  
    return String.format("id = %s\nsubId = %s\nvalue = %s\n", id, subId, value);  
}
```

```
curl -XGET localhost:8080/siad/api/echo/complex/monId/monsuId?value=maValeur  
id = monId  
subId = monsubId  
value = maValeur
```

121

Objet complexe ?

Renvoyer autre chose que des chaîne

```
@Path("user")  
public class UserAccess {  
  
    @GET  
    public User getUser(@QueryParam("id") int id) {  
        return new User(id, "name");  
    }  
}
```

```
curl -XGET localhost:8080/siad/api/user?id=1
```

ERREUR 500

122

Objet complexe ?

Regardons l'objet

```
@XmlRootElement  
@XmlAccessorType(XmlAccessType.FIELD)  
public class User {  
  
    @XmlAttribute  
    private int id;  
  
    private String name;  
  
    ....  
}
```

123

Objet complexe ?

Forcer le type

```
@Path("user")  
public class UserAccess {  
  
    @GET  
    @Produces(MediaType.APPLICATION_XML)  
    public User getUser(@QueryParam("id") int id) {  
        return new User(id, "name");  
    }  
}
```

```
curl -XGET localhost:8080/siad/api/user?id=1  
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<user id="1">  
<name>name</name>  
</user>
```

```
package javax.ws.rs.core;  
public class MediaType {  
    public final static String APPLICATION_XML = "application/xml";  
}
```

124

Négociation de contenu

1 méthode, plusieurs format...

```
@Path("user")
public class UserAccess {

    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public User getUser(@QueryParam("id") int id) {
        return new User(id, "name");
    }

curl -XGET -i localhost:8080/siad/api/user?id=1
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/xml
Content-Length: 92
Date: Thu, 04 Dec 2014 23:50:25 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><user id="1"><name>name</name></user>
```

125

Négociation de contenu

du XML plus précisément !

```
@Path("user")
public class UserAccess {

    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public User getUser(@QueryParam("id") int id) {
        return new User(id, "name");
    }

curl -XGET -i -H "Accept:application/xml" localhost:8080/siad/api/user?id=1
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Content-Type: application/xml
Content-Length: 92
Date: Thu, 04 Dec 2014 23:50:25 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><user id="1"><name>name</name></user>
```

126

Négociation de contenu

Et le JSON ?

```
@Path("user")
public class UserAccess {

    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public User getUser(@QueryParam("id") int id) {
        return new User(id, "name");
    }

curl -XGET -i -H "Accept:application/json" localhost:8080/siad/api/user?id=1
HTTP/1.1 200 OK
Connection: keep-alive
X-Powered-By: Undertow/1
Server: WildFly/8
Transfer-Encoding: chunked
Content-Type: application/json
Date: Thu, 04 Dec 2014 23:54:55 GMT

{"id":1,"name":"name"}
```

127

Provider par défaut

Type	Description
byte[]	All media type (*/*)
java.lang.String	All media type (*/*)
java.io.InputStream	All media type (*/*)
java.io.Reader	All media type (*/*)
java.io.File	All media type (*/*)
javax.activation.DataSource	All media type (*/*)
javax.xml.transform.Source	XML type
javax.xml.bind.JAXBElement	JAXB class, XML Media Types
MultivaluedMap<String, String>	Form content
javax.ws.rs.core.StreamingOutput	all media type (*/*), MessageBodyWriter only

128

Renvoyer des objets

```
@GET  
@Path("list")  
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})  
public List<User> getUserList(@QueryParam("id") int id) {  
    List<User> userList = new ArrayList<>();  
    userList.add(new User(1, "user1"));  
    userList.add(new User(2, "user2"));  
    return userList;  
}  
  
curl -XGET -i -H "Accept:application/json" localhost:8080/siad/api/user/list  
  
HTTP/1.1 200 OK  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Transfer-Encoding: chunked  
Content-Type: application/json  
Date: Thu, 04 Dec 2014 23:54:55 GMT  
  
[{"id":1,"name":"user1"}, {"id":2,"name":"user2"}]
```

129

Créer (POST)

```
@POST  
public void createUser(  
    @FormParam("id") int id,  
    @FormParam("name") String name  
) {  
    User user = new User(id, name);  
    logger.info("User -> {}", user);  
}  
  
curl -d "id=20&name=truc" -XPOST -i localhost:8080/siad/api/user  
  
HTTP/1.1 204 No Content  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Content-Length: 0  
Date: Fri, 05 Dec 2014 00:23:58 GMT
```

Pardon ??????



130

Créer (POST) Tentons le code 201 !!!!

```
@POST  
@Consumes("application/x-www-form-urlencoded")  
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})  
public User createUser(  
    @FormParam("id") int id,  
    @FormParam("name") String name  
) {  
    User user = new User(id, name);  
    logger.info("User -> {}", user);  
    return user;  
}  
  
curl -d "id=20&name=truc" -XPOST -i localhost:8080/siad/api/user  
  
HTTP/1.1 200 OK  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Content-Type: application/xml  
Content-Length: 93  
Date: Fri, 05 Dec 2014 00:30:18 GMT  
  
<?xml version="1.0" encoding="UTF-8" standalon  
user>
```

Toujours pas !



131

Maitrisons la réponse Tentons le code 201 !!!!

```
@POST  
@Consumes("application/x-www-form-urlencoded")  
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})  
public Response createUser(  
    @FormParam("id") int id,  
    @FormParam("name") String name  
) {  
    User user = new User(id, name);  
    logger.info("User -> {}", user);  
    return Response.status(Response.Status.CREATED).build();  
}  
  
curl -d "id=20&name=truc" -XPOST -i localhost:8080/siad/api/user  
  
HTTP/1.1 201 Created  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Content-Length: 0  
Date: Fri, 05 Dec 2014 00:34:46 GMT
```



132

Soyons Restful

```
@POST  
@Consumes("application/x-www-form-urlencoded")  
public Response createUser(  
    @FormParam("name") String name  
) {  
    User user = new User(userIndex.getAndIncrement(), name);  
    userList.add(user);  
    return  
Response.status(Response.Status.CREATED).entity(String.valueOf(user.getId())).build();  
}  
  
curl -d "name=truc" -XPOST -i localhost:8080/siad/api/user  
curl -d "name=truc" -XPOST -i localhost:8080/siad/api/user  
curl -d "name=truc" -XPOST -i localhost:8080/siad/api/user  
curl -d "name=truc" -XPOST -i localhost:8080/siad/api/user
```

Vérifions...

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":2,"name":"truc"}, {"id":3,"name":"truc"}]
```

133

Modification (PUT)

Vérifions

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":2,"name":"machin"},  
 {"id":3,"name":"truc"}]
```



135

Modification (PUT)

```
@PUT  
@Consumes("application/x-www-form-urlencoded")  
public Response updateUser(  
    @FormParam("id") int id  
    , @FormParam("name") String name  
) {  
    User user = gerUserFor(id);  
    user.setName(name);  
    return Response.ok().build();  
}
```

Ce que nous avons en base...

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":2,"name":"truc"}, {"id":3,"name":"truc"}]
```

Modifions...

```
curl -d "id=2&name=machin" -XPUT -i localhost:8080/siad/api/user  
HTTP/1.1 200 OK  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Content-Length: 0  
Date: Fri, 05 Dec 2014 01:08:15 GMT
```

134

Effacer (DELETE)

```
@DELETE  
public Response deleteUser(  
    @QueryParam("id") int id  
) {  
    User user = gerUserFor(id);  
    processDeleteUser(user);  
    return Response.ok().build();  
}
```

Ce que nous avons en base...

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":2,"name":"truc"}, {"id":3,"name":"truc"}]
```

Effaçons...

```
curl -I -XDELETE localhost:8080/siad/api/user?id=2  
HTTP/1.1 200 OK  
Connection: keep-alive  
X-Powered-By: Undertow/1  
Server: WildFly/8  
Content-Length: 0  
Date: Fri, 05 Dec 2014 01:08:15 GMT
```

136

Effacer (DELETE)

Vérifions

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":3,"name":"truc"}]
```

137

Exception

En cas de problème, on lève une WebApplicationException qui est converti par JAX-RS fonction du paramètre de constructeur.

```
@PUT  
@Consumes("application/x-www-form-urlencoded")  
public Response updateUser(  
    @FormParam("id") int id  
    , @FormParam("name") String name  
) {  
    User user = gerUserFor(id);  
    if (user == null) {  
        throw new WebApplicationException(Response.Status.NOT_FOUND);  
    }  
    user.setName(name);  
    return Response.ok().build();  
}
```

139

Et les erreurs ?

Ce que nous avons en base...

```
curl -H "accept:application/json" localhost:8080/siad/api/user/list  
[{"id":0,"name":"truc"}, {"id":1,"name":"truc"}, {"id":2,"name":"truc"}, {"id":3,"name":"truc"}]
```

```
curl -d "id=5&name=machin" -XPUT -i localhost:8080/siad/api/user  
HTTP/1.1 500 Internal Server Error  
Connection: keep-alive  
Content-Type: text/html; charset=UTF-8  
Content-Length: 8118  
Date: Fri, 05 Dec 2014 01:27:39 GMT
```



138

Exception

- WebApplicationException c'est bien...
- Mais pas tant que cela !!!
- Une gestion fine est préférable

140

Exception

Les provider

- Il est possible de créer des provider pour les exceptions
- Permet de gérer le traitement d'un type d'exception de manière unique
- Gestion de l'héritage des exceptions

141

“provider“ et exception Ce que cela change ?

```
@GET  
@Path("divide")  
@Produces(MediaType.TEXT_PLAIN)  
public String divide(@QueryParam("valA") String valueA, @QueryParam("valB") String valueB) throws  
WebApplicationException {  
    try {  
        System.out.println("valA = " + valueA);  
        System.out.println("valB = " + valueB);  
        return String.valueOf(Integer.valueOf(valueA) / Integer.valueOf(valueB));  
    } catch (NumberFormatException e) {  
        return "N/A";  
    } catch (ArithmeticsException arithmeticException) {  
        throw new WebApplicationException(Response.Status.BAD_REQUEST);  
    }  
}  
  
@GET  
@Path("divide")  
@Produces(MediaType.TEXT_PLAIN)  
public String divide(@QueryParam("valA") String valueA, @QueryParam("valB") String valueB) throws  
WebApplicationException, NumberFormatException, DivideByZeroException {  
    if (Integer.valueOf(valueB) == 0) {  
        throw new DivideByZeroException();  
    }  
    return String.valueOf(Integer.valueOf(valueA) / Integer.valueOf(valueB));  
}
```

143

Exception

Les provider

- Classe annotée avec @Provider
- implémente ExceptionMapper<Type>
- Type == classe de l'exception
- Contrat
 - toResponse pour renvoyer la réponse correspondant à l'exception.

142

“provider“ et exception

```
@Provider  
public class DivideByZeroExceptionMapper implements ExceptionMapper<DivideByZeroException> {  
  
    @Override  
    public Response toResponse(DivideByZeroException exception) {  
        return Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity("Diviser par  
zéro, c'est mal !!!").build();  
    }  
}
```

144

EntityProvider

- conversion de la représentation en Java et vice-versa (JAX-B par exemple)
- deux types : MessageBodyReader ou MessageBodyWriter
- Les deux sont annotés @Provider
- On peut restreindre les types consommés par @Consumes et @Produces

145

MessageBodyWriter

- Permet de gérer la génération de format spécifique
- Annotation @Provider
- Utilisation de l'annotation @Produces pour fixer le type produit.
- Implémente MessageBodyWriter<Type>
- Type == Classe à traduire
 - Contrat
 - isWriteable
 - getSize
 - writeTo

146

JSON / XML

```
{“book” :{  
    “id” :“1234”,  
    “authors” : [  
        {  
            “firstName” : ”Pierre”,  
            “lastName” : ”Dupont”  
        },  
        {  
            “firstName” : ”Paul”,  
            “lastName” : ”Durand”  
        }  
    ]  
}}
```

```
<book>  
    <id>1234</id>  
    <authors>  
        <author>  
            <firstName>Pierre</firstName>  
            <lastName>Dupont</lastName>  
        </author>  
        <author>  
            <firstName>Pierre</firstName>  
            <lastName>Dupont</lastName>  
        </author>  
    </authors>  
</book>
```

147

MessageBodyWriter

```
@Provider  
@Produces("application/json")  
public class UserGsonProvider implements MessageBodyWriter<User>{  
  
    @Override  
    public boolean isWriteable(Class<?> type, Type genericType, Annotation[] annotations,  
    MediaType mediaType) {  
        return true;  
    }  
  
    @Override  
    public long getSize(User t, Class<?> type, Type genericType, Annotation[] annotations,  
    MediaType mediaType) {  
        return 1;  
    }  
  
    @Override  
    public void writeTo(User t, Class<?> type, Type genericType, Annotation[] annotations,  
    MediaType mediaType, MultivaluedMap<String, Object> httpHeaders, OutputStream entityStream)  
    throws IOException, WebApplicationException {  
        try(OutputStreamWriter outputStreamWriter = new OutputStreamWriter(entityStream)) {  
            new Gson().toJson(t, type, outputStreamWriter);  
        }  
    }  
}
```

148

MessageBodyWriter Utilisation

```
@GET  
@Path("user")  
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON, "application/gson"})  
public User getUser() {  
    return new User("name", "surname");  
}
```

Fournit par Glassfish

149

MessageBodyReader

- Permet de gérer la lecture de format spécifique
- Annotation @Provider
- Utilisation de l'annotation @Consumes pour fixer le type produit.
- Implémente MessageBodyReader<Type>
- Type == Classe à traduire
 - Contrat
 - isReadable
 - readFrom

150

MessageBodyReader

```
@Provider  
@Consumes("application/csv")  
public class UserCsvProvider implements MessageBodyReader<User> {  
  
    @Override  
    public boolean isReadable(Class<?> type, Type genericType, Annotation[] annotations,  
    MediaType mediaType) {  
        return type == User.class;  
    }  
  
    @Override  
    public User readFrom(Class<User> type, Type genericType, Annotation[] annotations,  
    MediaType mediaType, MultivaluedMap<String, String> httpHeaders, InputStream entityStream)  
    throws IOException, WebApplicationException {  
        String content = CharStreams.toString(new InputStreamReader(entityStream,  
       Charsets.UTF_8));  
        Doing some stuff !!!!  
        return user;  
    }  
}
```

151

MessageBodyReader Utilisation

```
@POST  
@Path("show")  
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON, "application/gson"})  
@Consumes("application/csv")  
public User getUserFromCommaSeparatedValue(User user) {  
    return user;  
}
```

Généralement fourni par le serveur

152



153