جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

EST MEKNÈS
المدرسة العليا للتكنولوجيا
ÉCOLE SUPÉRIEURE DE TECHNOLOGIE

Manuel de travaux pratiques corrigés

Module Web-Service

Filière Licence DSIC

M. Lahmer
m.lahmer@umi-.ac.ma

2015-2021

Partie I: TPs Corriges Element XML

1. XML/DTD/Schéma

A. XML depuis DTD

1) Écrire au moins trois document xml valide à la dtd ci-dessous

```
<!DOCTYPE CATALOG [
<!ENTITY AUTHOR "John Doe">
<!ENTITY COMPANY "JD Power Tools, Inc.">
<!ENTITY EMAIL "jd@jd-tools.com">
<!ELEMENT CATALOG (PRODUCT+)>
<!ELEMENT PRODUCT
(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
<!ATTLIST PRODUCT
NAME CDATA #IMPLIED
CATEGORY (HandTool|Table|Shop-Professional) "HandTool"
PARTNUM CDATA #IMPLIED
PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
INVENTORY (InStock|Backordered|Discontinued) "InStock">
<!ELEMENT SPECIFICATIONS (#PCDATA)>
<!ATTLIST SPECIFICATIONS
WEIGHT CDATA #IMPLIED
POWER CDATA #IMPLIED>
<!ELEMENT OPTIONS (#PCDATA)>
<!ATTLIST OPTIONS
FINISH (Metal|Polished|Matte) "Matte"
ADAPTER (Included|Optional|NotApplicable) "Included"
CASE (HardShell|Soft|NotApplicable) "HardShell">
<!ELEMENT PRICE (#PCDATA)>
<!ATTLIST PRICE
MSRP CDATA #IMPLIED
WHOLESALE CDATA #IMPLIED
STREET CDATA #IMPLIED
SHIPPING CDATA #IMPLIED>
<!ELEMENT NOTES (#PCDATA)>
]>
```

Solution : XML minimal valide

```
<CATALOG>
<PRODUCT>
<SPECIFICATIONS>
cuisne
</SPECIFICATIONS>
<PRICE>
20
</PRICE>
</PRODUCT>
</CATALOG>
```

2) Écrire au moins trois document xml valide à la dtd ci-dessous

```
<!DOCTYPE REPORT[
<!ELEMENT REPORT (TITLE,(SECTION|SHORTSECT)+)>
<!ELEMENT SECTION (TITLE,%BODY;,SUBSECTION*)>
<!ELEMENT SUBSECTION (TITLE,%BODY;,SUBSECTION*)>
<!ELEMENT SHORTSECT (TITLE,%BODY;)>
<!ELEMENT TITLE %TEXT;>
<!ELEMENT PARA %TEXT;>
```

```
<!ELEMENT LIST (ITEM)+>
<!ELEMENT ITEM (%BLOCK;)>
<!ELEMENT CODE (#PCDATA)>
<!ELEMENT KEYWORD (#PCDATA)>
<!ELEMENT EXAMPLE (TITLE?,%BLOCK;)>
<!ELEMENT GRAPHIC EMPTY>
<!ATTLIST REPORT security (high | medium | low ) "low">
<!ATTLIST CODE type CDATA #IMPLIED>
<!ATTLIST GRAPHIC file ENTITY #REQUIRED>
<!ENTITY xml "Extensible Markup Language">
<!ENTITY sgml "Standard Generalized Markup Language">
<!ENTITY pxa "Professional XML Authoring">
<!ENTITY % TEXT "(#PCDATA|CODE|KEYWORD|QUOTATION)*">
<!ENTITY % BLOCK "(PARA|LIST)+">
<!ENTITY % BODY "(%BLOCK;|EXAMPLE|NOTE)+">
<!NOTATION GIF SYSTEM "">
<!NOTATION JPG SYSTEM "">
<!NOTATION BMP SYSTEM "">
]>
```

Solution : XML minimal valide

```
<REPORT>
<TITLE><CODE>A109292</CODE></TITLE>
</REPORT>
```

On considère le schéma suivant.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="e1">
<xsd:complexType>
<xsd:sequence maxOccurs="unbounded">
<xsd:element ref="e2" maxOccurs="2"/>
<xsd:choice maxOccurs="unbounded">
<xsd:element ref="e3"/>
<xsd:element ref="e4" minOccurs="0"/>
</xsd:choice>
<xsd:element ref="e1" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="e2" type="xsd:string"/>
<xsd:element name="e3" type="xsd:string"/>
<xsd:element name="e4">
<xsd:complexType>
<xsd:choice maxOccurs="unbounded">
<xsd:element ref="e2" maxOccurs="2"/>
<xsd:sequence maxOccurs="unbounded">
<xsd:element ref="e3"/>
<xsd:element ref="e4" minOccurs="0"/>
</xsd:sequence>
<xsd:element ref="e1" minOccurs="0" maxOccurs="unbounded"/>
</xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

3) Donner un document (exo1.xml) XML valide (le plus simple possible) pour ce schéma.

Solution

```
<e1><e2>hello</e2></e1>
```

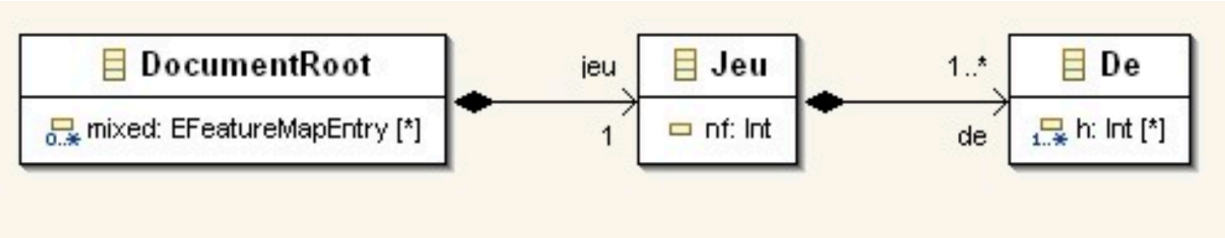
b) Donner une DTD (exo1.dtd) équivalente à ce schéma.

```
<!ELEMENT e1(e2+,(e3|e4*)>
<!ELEMENT e2 (#PCDATA)>
<!ELEMENT e3 (#PCDATA)>
<!ELEMENT e4 (e2+,e3,e4*,e1*)>
```

B. De l'UML vers Schéma

1) On demande de modéliser par un schéma XML un jeu de n dés à nf faces, le modèle gardant un historique de tous les lancers de dés.

a. Donner le diagramme UML du modèle.**(Solution)**



b. Ecrire le schéma XML. **(Solution)**

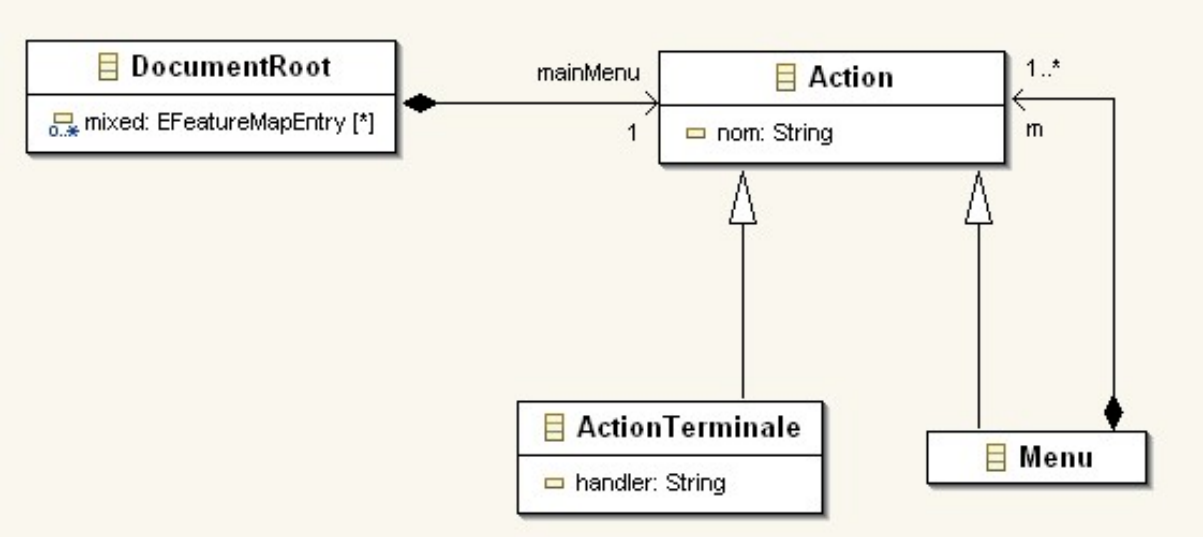
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="des" elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="des">
 <xs:element name="jeu" type="Jeu" />
 <xs:complexType name="Jeu">
  <xs:sequence>
   <xs:element name="de" maxOccurs="unbounded" type="De" />
  </xs:sequence>
  <xs:attribute name="nf" type="xs:int" />
 </xs:complexType>
 <xs:complexType name="De">
  <xs:sequence>
   <xs:element name="h" maxOccurs="unbounded" type="xs:int" />
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

c. Ecrire un document XML valide. **(Solution)**

```
<?xml version="1.0" encoding="UTF-8"?>
<jeu xmlns="des" xmlns:xsi=« http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="des ../xsd/des.xsd " nf="6">
 <de>
  <h>1</h>
  <h>4</h>
  <h>2</h>
</de> </jeu>
```

2) On demande de modéliser un système de menus en cascade:

a. Donner le diagramme UML du modèle **(Solution)** .



b. Ecrire le schéma XML **(Solution)**

```
<xsd:schema targetNamespace="cascade"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="cascade" elementFormDefault="qualified">
 <xsd:element name="mainMenu"  type="Action" />
 <xsd:complexType name="Action"  abstract="true">
  <xsd:attribute name="nom"  type="xsd:string" />
 </xsd:complexType>
 <xsd:complexType name="ActionTerminale">
  <xsd:complexContent>
   <xsd:extension base="Action">
    <xsd:attribute name="handler"  type="xsd:string" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
 <xsd:complexType name="Menu">
  <xsd:complexContent>
   <xsd:extension base="Action">
    <xsd:sequence>
     <xsd:element name="m" maxOccurs="unbounded" type="Action"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:schema>
```

2. DTD, Schéma, XSLT et SAX

On considère le document catalogCDs.xml suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="exo1.xsd">
       <Adress>
              <Name>Main Office</Name>
              <Street>Grosvenor Bvld.</Street>
              <City>Los Angelos</City>
              <State>California</State>
              <Zip>3141</Zip>
```

```
        </Adress>
        <!-- US-Adress : element optionnel -->
        <US-Adress>
                <Name>Main Office</Name>
                <Street>Grosvenor Bvld.</Street>
                <City>Los Angelos</City>
                <State>California</State>
                <Zip>3141</Zip>
                <stat></stat>
        </US-Adress>
        <Division>
                <Division-Name>Sales</Division-Name>
                <Location>Washington</Location>
                <Person Manager="true" Degree="MA">
                        <First>Allison</First>
                        <Last>Anderson</Last>
                        <PhoneExt>111</PhoneExt>
                        <EMail>Anderson@work.com</EMail>
                </Person>
                <Person Manager="false" Degree="BA">
                        <First>xxx</First>
                        <Last>yyy</Last>
                        <PhoneExt>222</PhoneExt>
                        <EMail>test@TEST.com</EMail>
                </Person>
        </Division>
</Company>
```

A. DTD/Schéma

1) Ecrire une DTD la plus simpliste possible

Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT catalog  ( cd+ )>
<!ATTLIST catalog
type CDATA #REQUIRED>
<!ELEMENT cd  ( title , artist , country , company , price , year )>
<!ATTLIST cd
id CDATA #REQUIRED>
<!ELEMENT country  (#PCDATA)>
<!ELEMENT artist  ( #PCDATA)>
<!ELEMENT year  (#PCDATA)>
<!ELEMENT price  (#PCDATA)>
<!ELEMENT company  (#PCDATA)>
<!ELEMENT title  (#PCDATA)>
```

2) Ecrire un schéma XML (exo1.xsd) décrivant la structure du document XML précédent et les hypothèses définies

- La valeur de degree est soit MA soit BA
- Une adresse e-mail doit être de la forme « string.string@string »
- US-Adress est une extension de Address à laquelle on ajoute les éléments stat
- L'élément zip a une valeur comprise entre 1000 et 4000

Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
	<!-- type adress -->
	<xs:complexType name="adressType">
		<xs:sequence>
			<xs:element name="Name" type="xs:string"/>
			<xs:element name="Street" type="xs:string"/>
			<xs:element name="City" type="xs:string"/>
			<xs:element name="State" type="xs:string"/>
			<xs:element name="Zip">
				<xs:simpleType>
					<xs:restriction base="xs:integer">
						<xs:minInclusive value="1000"/>
						<xs:maxInclusive value="4000"/>
					</xs:restriction>
				</xs:simpleType>
			</xs:element>
		</xs:sequence>
	</xs:complexType>

	<!-- US-Adress herite de type adress-->
	<xs:element name="US-Adress">
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="adressType">
					<xs:sequence>
						<xs:element name="stat"
maxOccurs="unbounded"></xs:element>
					</xs:sequence>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<!-- person attributes-->
	<xs:attribute name="Manager">
			<xs:simpleType>
				<xs:restriction base="xs:string">
					<xs:enumeration value="true"/>
					<xs:enumeration value="false"/>
				</xs:restriction>
			</xs:simpleType>
		</xs:attribute>
		<xs:attribute name="Degree">
			<xs:simpleType>
				<xs:restriction base="xs:string">
					<xs:enumeration value="MA"/>
					<xs:enumeration value="BA"/>
				</xs:restriction>
			</xs:simpleType>
		</xs:attribute>

	<!-- person adress -->
	<xs:complexType name="personType">
		<xs:sequence>
			<xs:element name="First" type="xs:string"/>
			<xs:element name="Last" type="xs:string"/>
			<xs:element name="PhoneExt">
```

```
                        <xs:simpleType>
                              <xs:restriction base="xs:integer">
                                    <xs:totalDigits value="3"/>
                              </xs:restriction>
                        </xs:simpleType>
                  </xs:element>
                  <xs:element name="EMail">
                        <xs:simpleType>
                              <xs:restriction base="xs:string">
                                    <xs:pattern value="[0-9a-zA-Z]{3,}\@[0-9a-
zA-Z]{3,}\.[0-9a-zA-Z]{2,}"></xs:pattern>
                              </xs:restriction>
                        </xs:simpleType>
                  </xs:element>
            </xs:sequence>
            <xs:attribute ref="Manager" use="required"/>
            <xs:attribute ref="Degree" use="required"/>
      </xs:complexType>

      <!-- division adress -->
      <xs:complexType name="divisionType">
            <xs:sequence>
                  <xs:element name="Division-Name" type="xs:string"/>
                  <xs:element name="Location" type="xs:string"/>
                  <xs:element name="Person" maxOccurs="unbounded"
type="personType"/>
            </xs:sequence>
      </xs:complexType>
```

B. XSLT/SAX

1) Ecrire une feuille de style xslt catalogue.xslt permettant d'afficher les CD par ordre décroissant de l'année tout en colorant les CDs dont le paye est USA

Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="/">
      <html>
      <body>
      <xsl:for-each select="//cd">
      <xsl:sort select="year" order="descending"></xsl:sort>
      <xsl:choose>
      <xsl:when test="country='USA'">
      <h1><xsl:value-of select="title"></xsl:value-of></h1>
      </xsl:when>
      </xsl:choose>
      </xsl:for-each>
      </body>
      </html>
      </xsl:template>
</xsl:stylesheet>
```

2) Ecrire un parser en Java qui permet de lire le fichier catalogue.xml et charger son contenu dans une hashMap avec comme clé l'année et la valeur un objet de type CD.

Solution

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class Parser extends DefaultHandler {
	boolean bCd, bTitle, bArtist, bCountry, bPrice, bCompany, bYear;;
	private HashMap<Integer, Cd> cds;
	private Cd cd;
	@Override
	public void startDocument() throws SAXException {
		super.startDocument();
		cds = new HashMap<Integer,Cd>();
	}
	@Override
	public void startElement(String uri, String localName, String qName,
Attributes atts) throws SAXException {
		super.startElement(uri, localName, qName, atts);
		if (qName.equals("cd") ) {
			bCd = true;
			cd = new Cd();
			if (atts != null)
				cd.setId(atts.getValue(0));
		}
		if (qName.equals("artist") ) {
			bArtist = true;
		}
		if (qName.equals("title") ) {
			bTitle = true;
		}
		if (qName.equals("country") ) {
			bCountry= true;

		}
		if (qName.equals("price") ) {
			bPrice = true;

		}
		if (qName.equals("company") ) {
			bCompany = true;
		}
		if (qName.equals("year") ) {
			bYear = true;
		}

	}
	@Override
	public void characters(char[] ch, int start, int length) throws SAXException
{
		super.characters(ch, start, length);
		String s =new String(ch, start,length);

		if(bTitle)
			cd.setTitle(s);
		if(bArtist)
			cd.setArtist(s);
		if(bCountry)
			cd.setCountry(s);
		if(bCompany)
			cd.setCompany(s);
		if(bPrice)
			cd.setPrice(s);
```

```
            if(bYear)
                  cd.setYear(Integer.parseInt(s));

      }
      @Override
      public void endElement(String uri, String localName, String qName) throws
SAXException {
      if(    qName.equals("cd")){
            if(bCd){
                  cds.put(new Integer(cd.getYear()), cd);
            bCd=false;
            }
      }
      if (qName.equals("artist") ) {
            bArtist = false;
      }
      if (qName.equals("title") ) {
            bTitle = false;
      }
      if (qName.equals("country") ) {
            bCountry= false;

      }
      if (qName.equals("price") ) {
            bPrice = false;

      }
      if (qName.equals("company") ) {
            bCompany = false;
      }
      if (qName.equals("year") ) {
            bYear = false;
      }
      }
```

3. Atelier Récapitulatif DTD, Schéma, XPATH et XSLT

A. XML/ DTD

Pour chacun des documents 1,2,3, indiquez s'il est valide ou pas. Quand le document n'est pas valide, indiquez la nature de l'erreur (ou des erreurs). Les documents sont indépendants les uns des autres.

1.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE a [
<!ELEMENT a (b*, c)>
<!ELEMENT b EMPTY>
 <!ELEMENT c (#PCDATA)>
<!ATTLIST c x CDATA #FIXED "bold"> ]>
<a><b/><b/>
<c x= "medium"> du texte </c> </a>
```

Solution : x ne correspond pas à bold

2.

```
<?xml version="1.0" encoding="iso-8859-1" ?> <!DOCTYPE a [
<!ELEMENT a (b*, c*, d?)>
<!ELEMENT d EMPTY>
<!ELEMENT b (#PCDATA)>]>
<a>
<b>ljs sldjf sljd </b> <b>mmmqmm qqq mmm qq </b> <d/>
</a>
```

Solution : OK

3.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE a [ <!ELEMENT a (b*, c)> <!ELEMENT b EMPTY>
<!ELEMENT c (#PCDATA)>
<!ATTLIST b
       truc CDATA #IMPLIED> ]>
<a>
<b/>
<b truc="bidule"/>
<c>Et voilà</c>
<b/> </a>
```

Solution : L'élement <b/> est de trop.

4.

```
<!ELEMENT examen (titre, date, questions) > <!ELEMENT titre (#PCDATA)>
<!ATTLIST examen
code NMTOKEN #REQUIRED> <!ELEMENT date EMPTY>
<!ATTLIST date
mois (jan|fev|mar|avr|mai|jun|jui|aou|sep|oct|nov|dec) #REQUIRED
annee NMTOKEN #REQUIRED>

<!ELEMENT questions (question, question, question, question, question,
question?) > <!ELEMENT question ((partie)+)>
<!ELEMENT partie (#PCDATA | partie)*>
```

B. Schémas

Objectif: Les schémas permettent de décrire les modèles de données de façon plus précise.

Transposez la DTD Livres.dtd en un schéma XSD avec les caractéristiques suivantes :

• Une année est une chaîne de 4 caractères compris dans l'espace 0...9
• Un livre contient exactement un titre, un prix, une année et au moins un auteur. De plus, on

associe à des éléments de ce type l'attribut "edition" qui précise si il s'agit d'un paperback

ou d'une édition reliée.
• Un prix est un type complexe, on lui associe deux attributs: "valeur" et "monnaie" ("monnaie" a

un type qui dérive de "string", son champs de valeur se limite aux chaînes "USD" et "EUR")

Livres.dtd

```
<!ELEMENT livres (livre*)>
<!ELEMENT livre (titre, auteur+, année, prix)>
<!ATTLIST livre edition CDATA #REQUIRED> <!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (prenom, nom, laboratoire?, pays)> <!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT année (#PCDATA)>
<!ELEMENT laboratoire (#PCDATA)> <!ELEMENT prix EMPTY>
<!ATTLIST prix
       monnaie CDATA #REQUIRED
       valeur CDATA #REQUIRED> <!ELEMENT pays (#PCDATA)>
```

Solution

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="livres"> <xsd:complexType>
<xsd:sequence>
<xsd:element ref="livre" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="livre" type="livreType"/> <xsd:element name="titre"
type=« xsd:string"/>
 <xsd:element name="auteur" type="auteurType"/>
<xsd:element name="annee"> <xsd:simpleType>
<xsd:restriction base="xsd:string"> <xsd:pattern value="[0-9]{4}"/> </
xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="prix" type="prixType"/>
<xsd:element name="prenom" type=« xsd:string"/>
<xsd:element name="nom" type="xsd:string"/>
<xsd:element name="laboratoire" type="xsd:string"/>
<xsd:element name="pays" type="xsd:string"/>
<xsd:complexType name="livreType"> <xsd:sequence>
<xsd:element ref="titre"/>
<xsd:element ref="auteur" minOccurs="1" maxOccurs="unbounded"/>
<xsd:element ref="annee"/>
<xsd:element ref="prix"/>
</xsd:sequence>
```

```
<xsd:attribute name="edition" use="required">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="paperback"/>
<xsd:enumeration value="relie"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
<xsd:complexType name="auteurType"> <xsd:sequence>
<xsd:element ref="prenom"/>
<xsd:element ref="nom"/>
<xsd:element ref="laboratoire" minOccurs="0"/> <xsd:element ref="pays"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="prixType">
<xsd:attribute name="valeur" type="xsd:decimal" use="required"/>
<xsd:attribute name="monnaie" use="required"> <xsd:simpleType>
<xsd:restriction base="xsd:string"> <xsd:enumeration value="USD"/>
<xsd:enumeration value="EUR"/> </xsd:restriction> </xsd:simpleType>
</xsd:attribute>
</xsd:complexType> </xsd:schema>
```

C. XPATH/XSLT

Voici un extrait du fichier qui contient la liste des gagnants du booker prize (liste de livres avec leur auteur et l'année de l'obtention du prix)

```
<?xml version="1.0"?> <booker>
<award>
<author>Kingsley Amis</author> <title>The Old Devils</title> <year>1986</year>
</award> <award>
[...] </award>
[...] </booker>
```

Trouvez les expressions XPath qui retournent les informations suivantes (on ne suppose que le contexte initial est l'élément racine de nom booker):

a) l'auteur du sixième livre dans la liste **Solution : //award[6]/author**

b) le titre du livre qui a gagné en 2000 **Solution : //award[year='2000']/title**

c) le nom de l'auteur du livre intitulé "Possession"

Solution : //award[title='Possession']/author

d) le titre des livres dont "J M Coetzee" est l'auteur

Solution : //award[author='J M Coetzee']/title

e) le nom de tous les auteurs dont le livre a gagné depuis 1995

Solution : //award[year > '1995']/author

f) le nombre total de prix décerné **Solution : count(//award)**

XSLT

Dans cet exercice on travaille sur un document XML, Cours.xml qui contient des informations sur des cours en informatique comme indiqué en annexe.

1- Donnez le résultat de la transformation du document suivant appliqué à Cours.xml

Commerce Electronique
Salle: Amphi A
Modules : PHP et MySQL, XML
Inscrits (note):

I235678 (16)
I784451
F569834

Bases de données

Salle: Painlevé
Modules : SQL, Optimisation
Inscrits (note):
A675432 (14)
B455978
B568709 (12)

2- Modifiez le programme afin qu'il affiche d'une part uniquement les numéros d'étudiants qui possèdent une note , et d'autre part, les enseignants pour chaque module (entre parenthèses après le nom du module).

Solution

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="text" />
<xsl:template match="/">
<xsl:apply-templates select="ENSEIGNEMENT/COURS"/> </xsl:template>
<xsl:template match="COURS"> ----------
<xsl:value-of select="@INTITULE"/>
Salle : <xsl:value-of select="SALLE"/>
Modules : <xsl:apply-templates select="MODULE"/> Inscrits (note):
<xsl:for-each select="ETUDIANT[NOTE]">
- <xsl:value-of select="@NUMERO"/>
<xsl:if test="NOTE"> (<xsl:value-of select="NOTE"/>)</xsl:if> </xsl:for-each>
</xsl:template>
<xsl:template match="MODULE">
<xsl:value-of select="@INTITULE"/>
<xsl:text> (</xsl:text>
<xsl:for-each select="ENSEIGNANT[position() &lt; last()]"> <xsl:value-of
select="."/>, </xsl:for-each>
<xsl:value-of select="ENSEIGNANT[last()]"/> <xsl:text>)</xsl:text>
<xsl:if test="position() &lt; last()"><xsl:text> , <xsl:text> </xsl:if> </
xsl:template>
</xsl:stylesheet>
```

2.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"/>
<xsl:template match="/"> <xsl:apply-templates select="//MODULE" />
</xsl:template>
<xsl:template match="*"> <xsl:copy>
```

```
<xsl:copy-of select="@*"/>
<xsl:if test="name()='MODULE'">
<xsl:element name="OPTION"> à venir...</xsl:element> </xsl:if>
<xsl:apply-templates/> </xsl:copy>
</xsl:template> </xsl:stylesheet>
```

3.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"/>
<xsl:template match="/"> <xsl:apply-templates select="*"/>
</xsl:template>
<xsl:template match="*"> <xsl:copy>
<xsl:apply-templates select="@*"/>
<xsl:apply-templates/> </xsl:copy>
</xsl:template>
<xsl:template match="@*"> <xsl:element name="{name()}">
<xsl:value-of select="."/> </xsl:element>
</xsl:template> </xsl:stylesheet>
```

4. Modele Examen : DTD, Schéma, XSLT et SAX

A. XML/DTD/XSLT

Soit le fichier catalogue.doc suivant :

Catalogue produits

Code	Marque	Affectation	IP
P00000	HP	S1	192.168.0.75
P00001	IBM	S2	192.168.0.76
P00002	DELL	B1	192.168.0.77
P00003	HP	B1	192.168.0.78

1) Donner le document XML associé (cata.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalogue SYSTEM "cata.dtd">
<?xml-stylesheet type="text/xsl" href="cata.xsl"?>
<catalogue>
	<pc code="P00000">
		<marque>HP</marque>
		<affectation>S1</affectation>
		<ip>192.168.0.75</ip>
	</pc>
	<pc code="P00001">
		<marque>IBM</marque>
		<affectation>S2</affectation>
		<ip>192.168.0.76</ip>
	</pc>
	<pc code="P00002">
		<marque>DELL</marque>
		<affectation>B1</affectation>
		<ip>192.168.0.77</ip>
	</pc>
```

```
	<pc code="P00003">
		<marque>HP</marque>
		<affectation>B1</affectation>
		<ip>192.168.0.78</ip>
	</pc>
</catalogue>
```

2) Ecrire une DTD valide pour le document XML de la question 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % text "(#PCDATA)">
<!ELEMENT catalogue (pc+)>
<!ELEMENT pc (marque, affectation, ip)>
<!ATTLIST pc code ID #REQUIRED>
<!ELEMENT marque %text;>
<!ELEMENT affectation %text;>
<!ELEMENT ip %tex;>
```

3) Créer une feuille de style (cata.xsl) qui permet d'afficher le contenu de catalogue.doc en web. Marque HP en vert

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/catalogue">
<html>
	<body>
	<center>
	<table border="1px">
	<tr>
	<td>Code</td>
	<td>Marque</td>
	<td>Affectation</td>
	<td>@IP</td>
	</tr>
	<xsl:for-each select="//pc">
	<xsl:choose>
	<xsl:when test="./marque='HP'">
	<tr>
	<td><font color="green"><xsl:value-of select="@code"/>
	</font></td>
	<td><font color="green"><xsl:value-of select="./marque"/>
</font></td>
	<td><font color=« green">
<xsl:value-of select="./affectation"/></font></td>
<td><font color="green"><xsl:value-of select="./ip"/>
</font></td>
</tr>
	</xsl:when>
		<xsl:otherwise>
			<tr>
			<td><xsl:value-of select="@code"/></td>
			<td><xsl:value-of select="./marque"/></td>
			<td><xsl:value-of select="./affectation"/></td>
			<td><xsl:value-of select="./ip"/></td>
			</tr>
		</xsl:otherwise>
```

```
</xsl:choose>
	</xsl:for-each>
			</table>
	</center>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

B. App Java (SAX)

On vous propose d'écrire un programme java graphique (voir figure) gestion des livres qui permettra de naviguer et de chercher un livre mot clé (titre).

Pour se faire je vous propose de suivre les étapes suivantes :
1) Créer une classe Pc définie par les champs (code, Marque, Affectation, Ip), les getteres et les setters.

```
public class PC {
	private String id, marque, affectation, ip;
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getMarque() {
		return marque;
	}
	public void setMarque(String marque) {
		this.marque = marque;
	}
	public String getAffectation() {
		return affectation;
	}
	public void setAffectation(String affectation) {
		this.affectation = affectation;
	}
	public String getIp() {
		return ip;
	}
	public void setIp(String ip) {
		this.ip = ip;
	}
}
```

2) Créer une classe Pareser qui hérite du DefaultHandler de l'API SAX et qui permet de charger tous les Pc dans une Liste. La classe comportera en plus une méthode getList() qui retourne la liste précitée.

```
import java.util.ArrayList;
import java.util.List;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class Parser extends DefaultHandler{
	private List<PC> pcs;
	private PC pc;
	private String parent;
	@Override
	public void startDocument() throws SAXException {
		super.startDocument();
		pcs = new ArrayList<>();
	}
	@Override
	public void startElement(String arg0, String arg1, String name,
			Attributes attributes) throws SAXException {
		super.startElement(arg0, arg1, name, attributes);
		parent = name;
		if (parent.equals("pc")){
			pc = new PC();
			for (int i = 0 ; i< attributes.getLength(); i++){
				if (attributes.getQName(i).equals("code"))
					pc.setId(attributes.getValue(i));
			}
		}
	}
	@Override
	public void characters(char[] ch, int start, int length) throws SAXException
{
		super.characters(ch, start, length);
		String value = new String(ch, start, length).trim();
		if (!value.isEmpty())
			switch(parent){
			case "marque":
				pc.setMarque(value);
				break;
			case "affectation":
				pc.setAffectation(value);
				break;
			case "ip":
				pc.setIp(value);
				break;
			}

	}
	@Override
	public void endElement(String arg0, String arg1, String name)
			throws SAXException {
```

```
		super.endElement(arg0, arg1, name);

		if (name.equals("pc"))
			pcs.add(pc);
	}

	@Override
	public void endDocument() throws SAXException {
		super.endDocument();
	}
	public List<PC> getList(){
		return pcs;
	}
}
```

3) Créer une classe Modèle définie par :

- une liste
- un constructeur
- une méthode next () qui retourne le livre suivant si on arrive à la fin en reprends sur le premier
- une méthode previous() qui retourne le livre précédent si on arrive au début on passe au dernier

```
import java.io.File;
import java.util.List;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
public class Model {
	private List<PC> pcs;
	private int current;
	public Model() {
		try {
		SAXParserFactory factory = SAXParserFactory.newInstance();
			SAXParser parser = factory.newSAXParser();
			Parser handler = new Parser();
			parser.parse(new File("cata.xml"), handler);
			pcs = handler.getList();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
	public PC current(){
		return pcs.get(current);
	}
	public PC next(){
		if (current==pcs.size()-1)
```

```
                current=0;
        else
                current++;
        return current();
    }
    public PC previous(){
        if (current==0)
                current=pcs.size()-1;
        else
                current--;
        return current();
    }
}
```

4) Créer une classe Ihm qui met en ouvre l'interface ci-dessus et implémente les événements associés aux boutons (next, prevouis)

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class IHM extends JFrame implements ActionListener{
    private JLabel lblCode, lblMarque, lblAffectation, lblIp;
    private JTextField txtCode, txtMarque, txtAffectation, txtIp;
    private JButton previous, next;
    private Model model;
    public IHM() {
        lblCode = new JLabel("Code");
        lblMarque = new JLabel("Marque");
        lblAffectation = new JLabel("Affectation");
        lblIp = new JLabel("IP");
        txtCode = new JTextField(20);
        txtMarque = new JTextField(20);
        txtAffectation = new JTextField(20);
        txtIp = new JTextField(20);
        previous = new JButton("PrÈcÈdent");
        next = new JButton("Suivant");
        previous.addActionListener(this);
        next.addActionListener(this);
        JPanel panel = new JPanel(new GridLayout(2, 2));
        JPanel pnlCode = new JPanel(new FlowLayout());
        JPanel pnlMarque = new JPanel(new FlowLayout());
        JPanel pnlAffectation = new JPanel(new FlowLayout());
        JPanel pnlIp = new JPanel(new FlowLayout());
        pnlCode.add(lblCode);
        pnlCode.add(txtCode);
```

```
		pnlMarque.add(lblMarque);
		pnlMarque.add(txtMarque);
		pnlAffectation.add(lblAffectation);
		pnlAffectation.add(txtAffectation);
		pnlIp.add(lblIp);
		pnlIp.add(txtIp);
		panel.add(pnlCode);
		panel.add(pnlMarque);
		panel.add(pnlAffectation);
		panel.add(pnlIp);
		JPanel pnlButtons = new JPanel(new FlowLayout());
		pnlButtons.add(previous);
		pnlButtons.add(next);

		setLayout(new BorderLayout());
		add(panel, BorderLayout.NORTH);
		add(pnlButtons, BorderLayout.SOUTH);
		setTitle("Catalogue des PCs");
		pack();
		setVisible(true);
		setDefaultCloseOperation(EXIT_ON_CLOSE);
		model = new Model();
		fill(model.current());
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		if (e.getSource().equals(previous)){
			fill(model.previous());
		}
		if (e.getSource().equals(next)){
			fill(model.next());
		}
	}
	public void fill(PC pc){
		txtCode.setText(pc.getId());
		txtMarque.setText(pc.getMarque());
		txtAffectation.setText(pc.getAffectation());
		txtIp.setText(pc.getIp());
	}
	public static void main(String[] args) {
		new IHM();
	}
}
```

Partire II: TPs corrigés Web-Service

1. Configuration Eclipse/Axis 2

Configuration axis avec eclipse

1. Télécharger axis2 est le décompresser dans $HOMEAXIS
2. Télécharger le plugin eclipse pour axis2 et mettre le .jar dans le répertoire dropins de $HOMEECLIPSE
3. Démarrer eclipse, puis dans préférences—>Webservice configurer axis2 en précisant le chemin $HOMEAXIS
4. Créer un nouveau projet webdynamic AXIS001 en précisant la version du module web 2.5, le runtime tomcat et la facette axis 2
5. Ajouter manuellement dans lib du projet AXIS001 les jars: jstl.jar et standrd.jar puis xmlschema-core-2.2.1.jar et xmlschema-core-2.2.1.jar depuis $HOMEAXIS/lib

6. Dans le répertoire src du projet, créer un package estm.dsic.ws dans lequel vous créer une interface Ihello.java et une classe POJO Hello.java avec une seule méthode sayHello(String Lang)
7. Générer le service en mettant la souris sur la classe Hello.java, puis à partir du bouton droit vous sélectionner générer web service (Dans le type d'encodage utiliser Document/literal.
8. Remplacer dans service.xml, les attributs suivants :
   http://www.w3.org/**2004/08**/wsdl/in-only --> http://www.w3.org/**ns**/wsdl/in-only
   http://www.w3.org/**2004/08**/wsdl/in-out --> http://www.w3.org/**ns**/wsdl/in-out
9. Une fois le web service généré correctement, lancer le projet sous tomcat et vérifier. Accéder à la page http://localhost:8080/AxisWs0001/services/Hello?wsdl pour voir le fichier wsdl
10. Régénérer le service en utilisant les autres types d'encodage vu en cours et comparer les fichier wsdl gérérés

Client Lourds Axis

1. Crée un projet java AXISCLIENT
2. A partir du terminal exécuter la commande $HOMEAXIS /bin/wsdl2java.bat -uri http://localhost:8080/AxisWs0001/services/Hello?wsdl -u -o $HOMEAXISCLIENT
3. Ajouter au BuildPath du projet les .jars de $HOMEAXIS/lib
4. Créer une classe Main dans laquelle vous tester le service :

```
Stub service=new Stub();
IStub.SayHello request=new Stub.SayHello();
request.setLang("ar");
ServiceStub.SayHelloResponse response=service.sayHello(request);
System.out.println(response.getSayHelloReturn());
```

2. Service web SOAP en JAX-WS

Créer un service web simple qui fournit des informations sur les pays.

1. Créez une classe Java `CountryService` qui contient une méthode permettant de récupérer les informations sur un pays (par exemple, le nom du pays, la capitale, la population, etc.).

2. Utilisez JAX-WS pour exposer cette classe comme un service web. Définissez une interface `CountryWebService` avec une méthode `getCountryInfo(String countryName)`.

3. Remporter le contenu du fichier wsdl

4. Publiez le service web à l'aide de `Endpoint.publish()`. Assurez-vous que le service est accessible via une URL.

5. Créez un client JAX-WS simple qui appelle la méthode `getCountryInfo` du service web en utilisant l'URL publiée.

6. Créez un client Python simple qui appelle la méthode `getCountryInfo` du service web en utilisant l'URL publiée. Vous pouvez utilisé l'api zeep (pip install zeep)

Solution

1.

```
// CountryService.java
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class CountryService {

    @WebMethod
    public String getCountryInfo(String countryName) {
       switch(countryName) :
       {
       case "France" :
       return "Country: France, Capital: Paris, Population: 67 million";
       case "USA":
       return "Country: USA, Capital: Washington D.C., Population: 331 million";

       }
          }
}
```

2. // CountryWebService.java

```
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface CountryWebService {
    @WebMethod
    String getCountryInfo(String countryName);
}
```

3.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             name="CountryService"
             targetNamespace="http://webservice.example/">

    <message name="getCountryInfo">
        <part name="countryName" type="xsd:string"/>
    </message>

    <message name="getCountryInfoResponse">
        <part name="result" type="xsd:string"/>
    </message>

    <portType name="CountryWebService">
        <operation name="getCountryInfo">
            <input message="tns:getCountryInfo"/>
            <output message="tns:getCountryInfoResponse"/>
        </operation>
    </portType>

    <binding name="CountryServiceBinding" type="tns:CountryWebService">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
http"/>
        <operation name="getCountryInfo">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>

    <service name="CountryServiceService">
        <port name="CountryServicePort" binding="tns:CountryServiceBinding">
            <soap:address location="http://localhost:8080/country"/>
        </port>
    </service>

</definitions>
```

4. // CountryServicePublisher.java

```
import javax.xml.ws.Endpoint;

public class CountryServicePublisher {
    public static void main(String[] args) {
        String url = "http://localhost:8080/country";
        Endpoint.publish(url, new CountryService());
        System.out.println("Service published at: " + url);
    }
}
```

5. **// CountryServiceClient.java**

```
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import java.net.URL;
public class CountryServiceClient {
    public static void main(String[] args) throws Exception {
        String url = "http://localhost:8080/country?wsdl";
        URL wsdlUrl = new URL(url);
        QName qname = new QName("http://webservice.example/",
"CountryServiceService");
        Service service = Service.create(wsdlUrl, qname);

        CountryWebService countryService =
service.getPort(CountryWebService.class);

       String result = countryService.getCountryInfo("France");
        System.out.println("Result for France: " + result);
    }
}
```

6.

```
from zeep import Client

# L'URL du fichier WSDL du service web
wsdl_url = 'http://localhost:8080/country?wsdl'
# Création d'une instance du client en utilisant l'URL du WSDL
client = Client(wsdl_url)
# Récupération du port du service web
country_service = client.bind('CountryServiceService', 'CountryServicePort')
# Appel de la méthode du service web pour obtenir des informations sur la # France
result = country_service.getCountryInfo('France')

# Affichage du résultat
print(f'Result for France: {result}')
```

3. Web service SOAP avec Conteneur

L'objectif est de mettre en place un service web annuaire avec conteneur ici Tomcat Axis2. Pour ce faire il faut suivre les étapes suivantes :

1. Créez la classe du service d'annuaire

```
// ContactDirectoryService.java
public class ContactDirectoryService {
    public Contact getContactInfo(String contactId) {
                Contact contact = new Contact();
        contact.setId(contactId);
        contact.setName("John Doe");
        contact.setEmail("john.doe@example.com");
        contact.setPhone("555-1234");
        return contact;
    }
}
```

2. // Contact.java

```
import java.io.Serializable;

public class Contact implements Serializable {
    private String id;
    private String name;
    private String email;
    private String phone;

    // Getters and setters (omis pour la brièveté)
}
```

3. Créez un fichier `services.xml` dans le répertoire `WEB-INF/services` de votre projet web avec le contenu suivant :

```
<serviceGroup>
    <service name="ContactDirectoryService" scope="application">
        <parameter name=« ServiceClass">estm.dut.ws.ContactDirectoryService</
parameter>
        <operation name="getContactInfo">
        <messageReceiver class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/
>
        </operation>
    </service>
</serviceGroup>
```

4. Deployer l'application dans tomcat

5. Pour tester on va créer un Client java main pour consommer ce service

```
Importer dans le répertoire src de votre projet eclipse les fichiers nécessaires :
wsdl2java -uri http://localhost:8080/axis2/services/ContactDirectoryService?wsdl -o
/path/to/src/folder

import org.apache.axis2.AxisFault;
import org.apache.axis2.addressing.EndpointReference;
import org.apache.axis2.client.Options;
import org.apache.axis2.rpc.client.RPCServiceClient;

public class Axis2Client {
    public static void main(String[] args) {
        try {
            RPCServiceClient serviceClient = new RPCServiceClient();
            Options options = serviceClient.getOptions();
            EndpointReference targetEPR = new EndpointReference("http://
localhost:8080/axis2/services/ContactDirectoryService");
            options.setTo(targetEPR);

            // Paramètres de l'opération getContactInfo
            Object[] parameters = new Object[]{"123456"}; // Remplacez "123456" par
l'identifiant du contact que vous souhaitez récupérer
```

```
            Class[] returnTypes = new Class[]{Contact.class}; // Supposons que
Contact soit la classe représentant les informations du contact
            Object[] response = serviceClient.invokeBlocking("getContactInfo",
parameters, returnTypes);


// Manipulez la réponse comme nécessaire
            Contact contact = (Contact) response[0];
            System.out.println("Contact ID: " + contact.getId());
            System.out.println("Contact Name: " + contact.getName());
            System.out.println("Contact Email: " + contact.getEmail());
            System.out.println("Contact Phone: " + contact.getPhone());
        } catch (AxisFault e) {
            e.printStackTrace();
        }
    }
}
```

4. Web service Rest en JAX-RS

On vous propose de mettre en place un projet java back-end en jax-rs pour l'authentification.

1. Créer la base de données users et la table T_user(login,password)
2. Créer un projet maven en ajoutant les dépendances suivantes :

```
<dependencies>
    <!-- Jersey -->
    <dependency>
        <groupId>org.glassfish.jersey.containers</groupId>
        <artifactId>jersey-container-servlet</artifactId>
        <version>2.34</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
        <version>2.34</version>
    </dependency>
    <!-- MySQL JDBC Driver -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.23</version>
    </dependency>
</dependencies>
Créer le fichier web.xml pour la configuration du jax-rs dans Tomcat
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>Jersey App</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
        <init-param>
```

```
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>your.package.name.AppConfig</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
         <servlet-name>Jersey App</servlet-name>
        <url-pattern>/api/*</url-pattern>
    </servlet-mapping>

</web-app>
```

4. Classe service

```
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

@Path("/auth")
public class AuthService {

    @POST
    @Path("/login")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response login(User user) {
        try {
            Connection connection = DataBase.getConnection();
           String query = "SELECT * FROM users WHERE username = ? AND password
= ?";
            try (PreparedStatement statement = connection.prepareStatement(query))
{
                statement.setString(1, user.getUsername());
                statement.setString(2, user.getPassword());

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        return Response.status(Response.Status.OK).entity("Login
successful").build();
                    } else {
                        return
Response.status(Response.Status.UNAUTHORIZED).entity("Invalid
credentials").build();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            return
Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity("Error during
login").build();
        }
    }
}
```

5. Creer ensuite la classe de configuration jax-rs

```
import org.glassfish.jersey.server.ResourceConfig;
```

```
public class AppConfig extends ResourceConfig {
    public AppConfig() {
        packages("estm.dsic. rs.auth");    }
}
```