XML et Architectures Logicielles Distribuées

Baabda :
M. E. ABOU ZEID
M. E. MATTA

Zahlé :
Mlle. C. SAAD

Mejdelaya:
M. R. ABI NADER

OBJECTIF DU COURS

- Se familiariser avec:
  - la **représentation** des données
  - les **technologies XML**
  - les **architectures distribuées**

- Employer les technologies XML pour:
  - structurer les données
  - interroger, manipuler et valider les données XML

- Concevoir et implémenter une architecture logicielle distribuée

PLAN DU COURS

- **Chapitre 1: Le Langage XML**
- Chapitre 2: XPath et APIs XML
- Chapitre 3: Les services web
- Chapitre 4: Les architectures distribuées
- Chapitre 5: Le Schéma XML

CHAPITRE 1

LE LANGAGE XML

PLAN DU CHAPITRE

- Concepts et Terminologies
- Le Langage XML
- XML, HTML et JSON
- Syntaxe XML
- Exercice

Concepts et Terminologies

Concepts et Terminologies

- **Langages de Balisage (« *Markup Languages* »):** langages servant à définir et à structurer les informations contenues dans un document
  - HTML
  - XML

- **Balise :** élément sémantique de base des langages de balisage.
  - HTML
    - <h1>Titre 1</h1>
    - <img src="myImage.png" >
  - XML
    - <name>eyrolles</name>
    - <place>paris</place>

Concepts et Terminologies

Service Web	API (Application Programming Interface*)
✓ Des **moyens de communication**. ✓ Les services web sont des API utilisant HTTP	
Facilite l'interaction entre 2 machines **à travers un réseau**	• Assure un **interfaçage entre 2 applications différentes** • Permet aux vendeurs tiers (3rd party) de programmer en interfaçant aisément avec d'autres programmes: Google Maps API, Twitter APIs, etc.
Basée sur le Web	Peut être: • Basée sur le Web: site web ou application mobile connectée • Non basée sur le Web: système d'exploitation (*Linux Kernel API*) ou application (API XML).

**Interface d'application et non pas une interface utilisateur*

Le Langage XML

Le Langage XML

- **XML** (**EX**tensible **M**arkup **L**anguage)
- Langage de balisage **structuré**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <annuaire>
  - <personne>
      <nom>HEUTE</nom>
      <prenom>Thomas</prenom>
      <email>webmaster@xmlfacile.com</email>
    </personne>
  - <personne>
      <nom>CANTAT</nom>
      <prenom>Bertrand</prenom>
      <email>noir@desir.fr</email>
    </personne>
  </annuaire>
```

Document XML

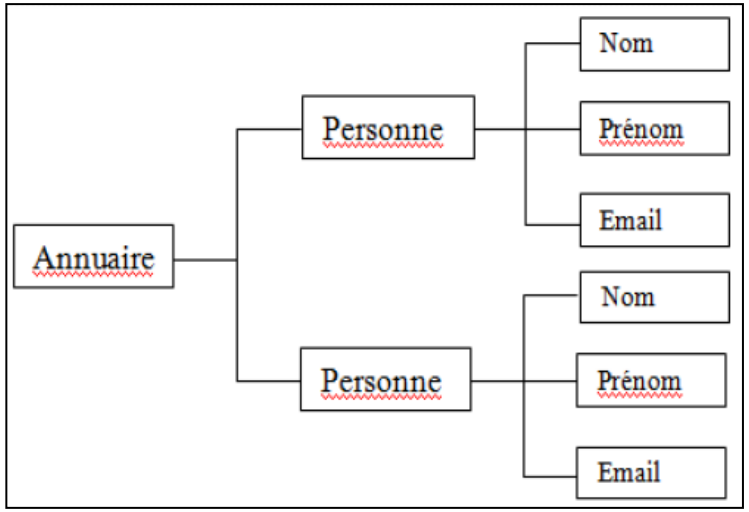


Schéma correspondant (arborescence)

Le Langage XML

- Éléments XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>                    Entête
<!DOCTYPE  DEVIS  SYSTEM"exemple.dtd">

<DEVIS> ←——— Elément racine
   <!-- Entête -->
   <IDENTIFIANT>0401</IDENTIFIANT> ← Elément
   <LIBELLE>Achats Réseaux</LIBELLE>
   <STATUT>Demande d'achat</STATUT>
   <FOURNISSEUR>Alcatel Commutation</FOURNISSEUR>
   <MONTANT devise="FRF">1452805.30</MONTANT>
   <DATE_LIVRAISON>12/12/2000<DATE_LIVRAISON>                             Contenu
   <!-- Lignes article --> ← Commentaire
   <ARTICLE>
      <REFERENCE>ISML43S904</REFERENCE>
      <QUANTITE>280</QUANTITE>  ← Attribut
      <PRIX_UNITAIRE devise="FRF">408.00</PRIX_UNITAIRE>
      <IMAGE href="media/ISML43S904.gif"/> ← Elément vide
   </ARTICLE>
   ...
</DEVIS>
```

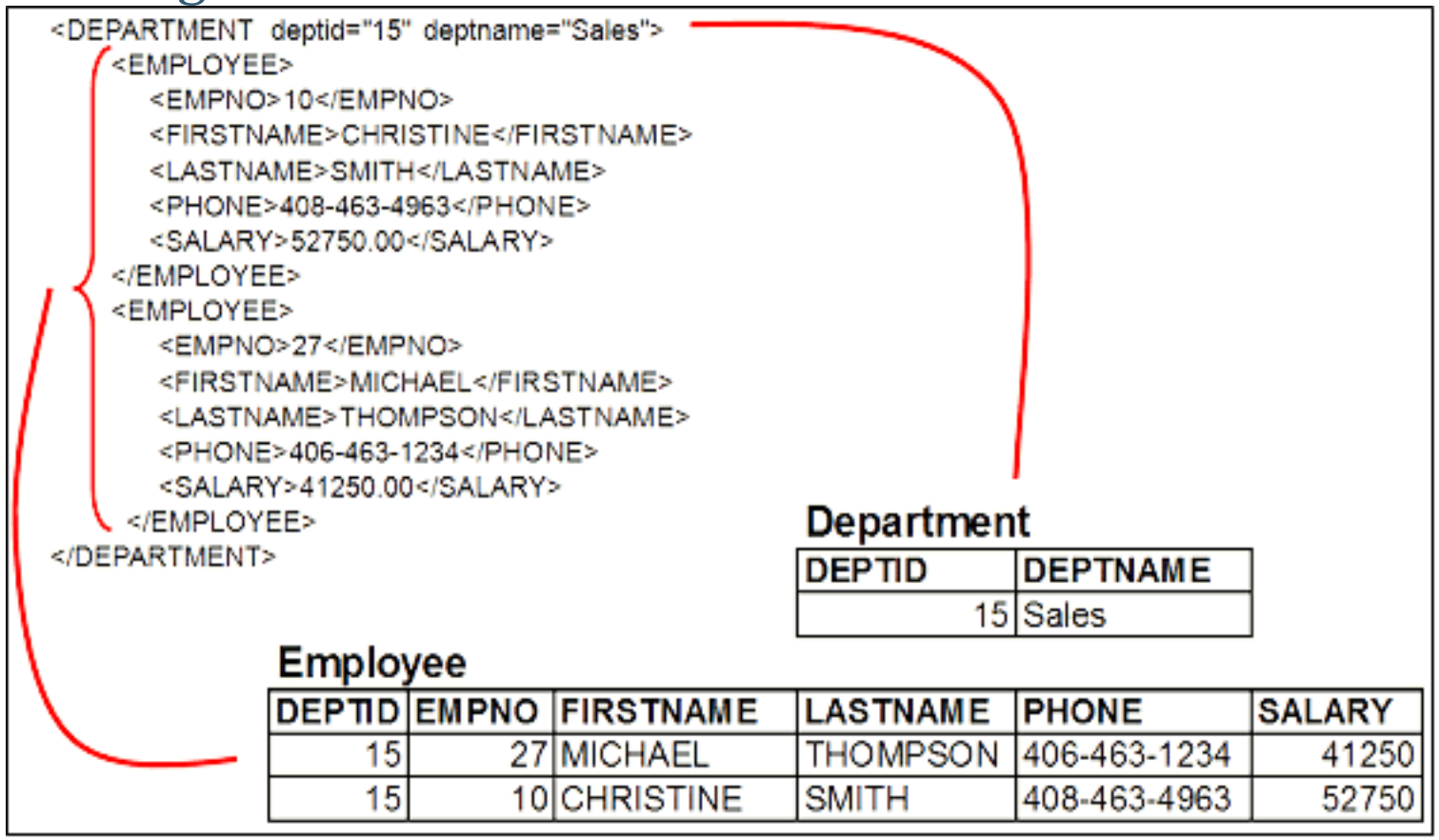
Le Langage XML

- **Basé sur 2 concepts fondamentaux..**
  - **Structure, contenu et présentation séparés**
    - À un même contenu peuvent être associées plusieurs mises en forme
  - **Les balises ne sont pas figées (fixes)**
    - Création des balises lorsque l'on en a besoin

- **Conséquences de ces 2 concepts: XML est un…**
  - **format de document**
    - Extension: .xml
  - **format de données**
  - **mode de structuration de l'information**
  - **métalangage**
    - employé pour créer d'autres langages de balisage

Le Langage XML

- Stockage de données avec XML



XML en tant qu'alternative plus universel que la BDD Relationnelle propriétaire

XML, HTML et JSON

XML, HTML et JSON

XML et HTML

- Comparé à HTML, les balises dans XML:
  - Ont **un sens**
  - Ont une **hiérarchie**: données structurées en arborescence
  - Sont **extensibles**: définition de nouvelles balises
  - Ont pour but de **stocker les données** (et non pas les présenter)

```
<BIBLIO SUBJECT="XML">
    <BOOK ISBN="9782212090819">
        <AUTHOR>
            <FIRSTNAME>Jean-Christophe</FIRSTNAME>
            <LASTNAME>Bernadac</LASTNAME>
        </AUTHOR>
        <AUTHOR>
            <FIRSTNAME>François</FIRSTNAME>
            <LASTNAME>Knab</LASTNAME>
        </AUTHOR>
        <TITLE>Construire une application XML</TITLE>
        <PUBLISHER>
            <NAME>Eyrolles</NAME>
            <PLACE>Paris</PLACE>
        </PUBLISHER>
        <DATEPUB>1999</DATEPUB>
    </BOOK>
</BIBLIO>
```

- Le livre ayant pour code 9782212090819 est écrit par 2 auteurs.
- Chaque élément "author" comprend le prénom ("firstname") et le nom ("lastname") de l'auteur.

XML, HTML et JSON

JSON (JavaScript Object Notation)

- C'est un **format de données**
- Ayant 2 types de structures:
  - un **objet**: ensemble de paires clé/valeur (key/value)
  - un **tableau** d'objets
- largement utilisé pour **échanger des données sur Internet**

Comparé au XML, JSON:

1. Est plus compacte (léger): plus employé pour consulter des données sur le serveur, et les utiliser (services web)
2. A une vitesse de traitement plus grande

Comparé à JSON, XML:

1. Est mieux adapté pour la **présentation**
2. Permet l'emploi des **attributs**
3. Permet la création **d'espaces de noms**
4. Permet **l'extraction** (XPath, XQuery, etc.)
5. Permet la **validation** des données (grammaire DTD/XSD)
6. Permet la **transformation** en d'autres types de formats (XSLT)

XML, HTML et JSON

XML et JSON

```
<?xml version="1.0"?>
<book id="123">
        <title>Object Thinking</title>
        <author>David West</author>
        <published>
                <by>Microsoft Press</by>
                <year>2004</year>
        </published>
</book>
```

Données XML

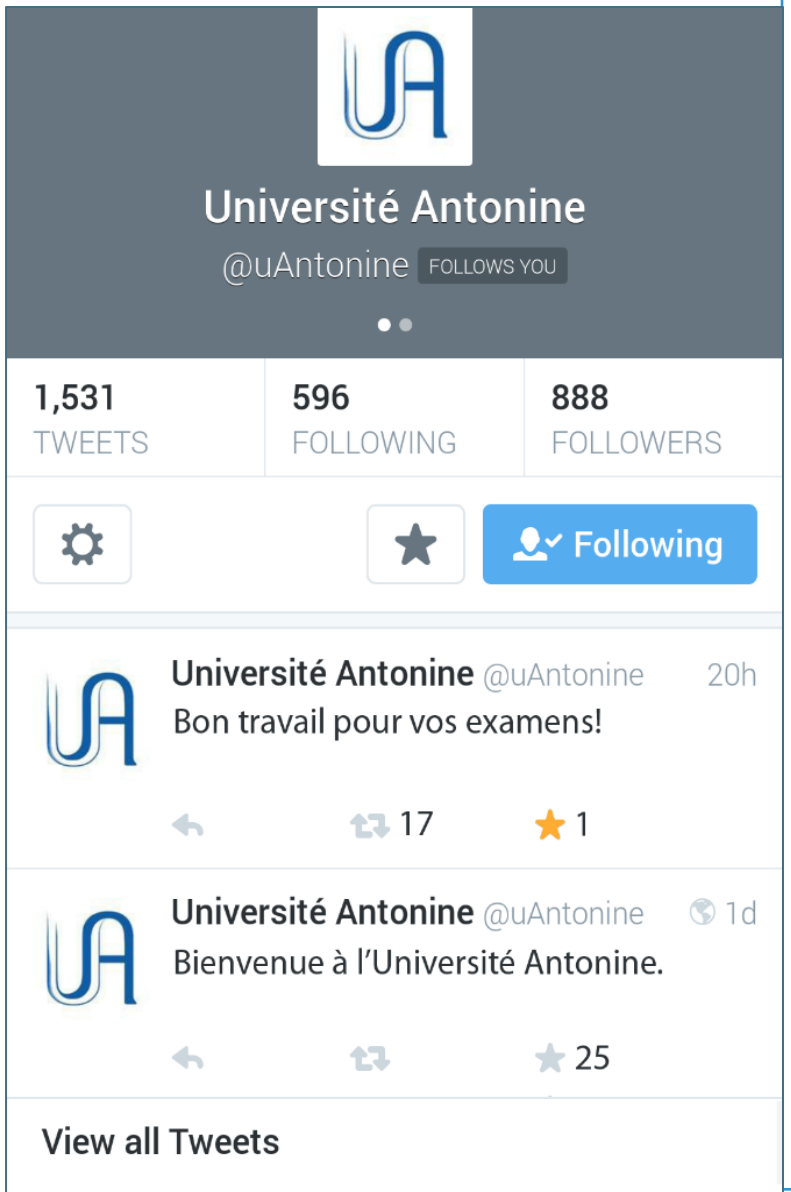
(167 caractères)

```
{
        "id": 123,
        "title": "Object Thinking",
        "author": "David West",
        "published": {
          "by": "Microsoft Press",
          "year": 2004
        }
}
```

Données JSON

(140 caractères)

XML, HTML et JSON




```
{
  "id": 8839,
  "name": "Université Antonine",
   ...
  "statuses": [
    {
      "created_at": "2015-03-21T17:32:00",
      "id": 579334,
      "text": "Bon travail pour vos examens!",
      "retweet_count": 17,
      "favorite_count": 1,
    },
    {
      "created_at": "2015-03-21T07:40:10",
      "id": 579185,
      "text": "Bienvenue à l'Université Antonine.",
      "retweet_count": 0,
      "favorite_count": 25
    }
  ]
}
```


Exemple illustrant:

- **A gauche:** Une capture d'écran du compte de l'Université Antonine sur l'application mobile de Twitter.
- **A droite:** Les informations retournées (modifiées) par l'interrogation de l'API de Twitter.

SYNTAXE XML

SYNTAXE XML

- Un document XML « **bien formé** »
  - Obéit aux règles syntaxiques du XML (prérequis pour la validité)

- Un document XML « **valide** »
  - Respecte toutes les règles de construction (« grammaire ») définies dans la DTD/XSD

Tout document XML

- Commence par **le prologue**, regroupant:
  - La **déclaration XML**
  - Les instructions de traitement: interprétées par l'application à laquelle elle est destinée
  - La déclaration de DTD/XSD

- Contient **l'arbre d'éléments** regroupant :
  - **L'élément racine :** un seul élément qui englobe tous les autres
  - **Les éléments :** les composants majeurs pouvant contenir du texte/d'autres éléments
  - **Les attributs:** <contact university="Antonine"> Didier Donze </ contact>

SYNTAXE XML

- **Déclaration XML** (ordre des attributs à respecter)
  - **<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>**
    - **version** = version du XML utilisé dans le document
    - **encoding** = jeu de codage de caractères utilisé (par défaut UTF-8)
    - **standalone** = dépendance du document par rapport à une grammaire DTD
      - « yes » : le processeur de l'application n'attend aucune DTD extérieur au document
      - « no » : le processeur de l'application attend une référence de DTD extérieur

SYNTAXE XML

Quand-est ce qu'on utilise des éléments?

- lorsque l'ordre est important
- lorsqu'on veut réutiliser un élément plusieurs fois (avec le même parent)
- lorsqu'on veut avoir des descendants/ une structure interne
- pour représenter un type de données (objet); ses propriétés sont des "attributs"

Quand-est ce qu'on n'utilise pas des attributs? Lorsqu'on veut:

- Stocker plusieurs valeurs (éléments à plusieurs enfants)
- Décrire des structures
- Extensibilité
- Manipulation facile

Utiliser des attributs pour identifier des éléments

- Exemple: <note id="p501">

SYNTAXE XML: quelques règles

- Les balises **ne sont pas prédéfinies** et **doivent toutes être fermées**
- Toutes **les balises sans contenu** doivent **se terminer par les caractères />**
- Les noms des balises de début et de fin **correspondent exactement et en terme de casse**
  - *Faux : <Parent>Bob</parent>*
  - *Faux : <Parent>Bob</enfant>*
- Les balises **ne doivent pas se chevaucher**
  - *Faux : <parent><enfant>Alice</parent></enfant>*
  - *Vrai : <parent><enfant>Alice</enfant></parent>*

SYNTAXE XML: quelques règles

- Les noms des **attributs n'apparaissent qu'une fois pour chaque élément**
- Les valeurs des **attributs** sont entre **guillemets** ou **apostrophes**
  - *Exemple: <date anniversaire="020911">*
- Les noms ne peuvent pas:
  - débuter par un nombre ou un signe de ponctuation
  - commencer par les lettres xml, XML ou Xml...
  - contenir des espaces
- A éviter certains signes qui pourraient prêter à confusion comme - ; . < >

SYNTAXE XML

- **Les sections CDATA :**
  - Par défaut, toutes les données sont des **PCDATA** (**P**arsed **C**haracter **Data**): texte analysé par un parseur
    - Les balises seront prises dans le contexte du balisage
    - Les entités seront étendues
  - **CDATA** permet de définir un bloc de caractères ne devant pas être analysés par le processeur XML :

    <![CDATA[Bienvenue dans cette classe du cours <XML et Architecture Logicielle Distribuee>**]]>**

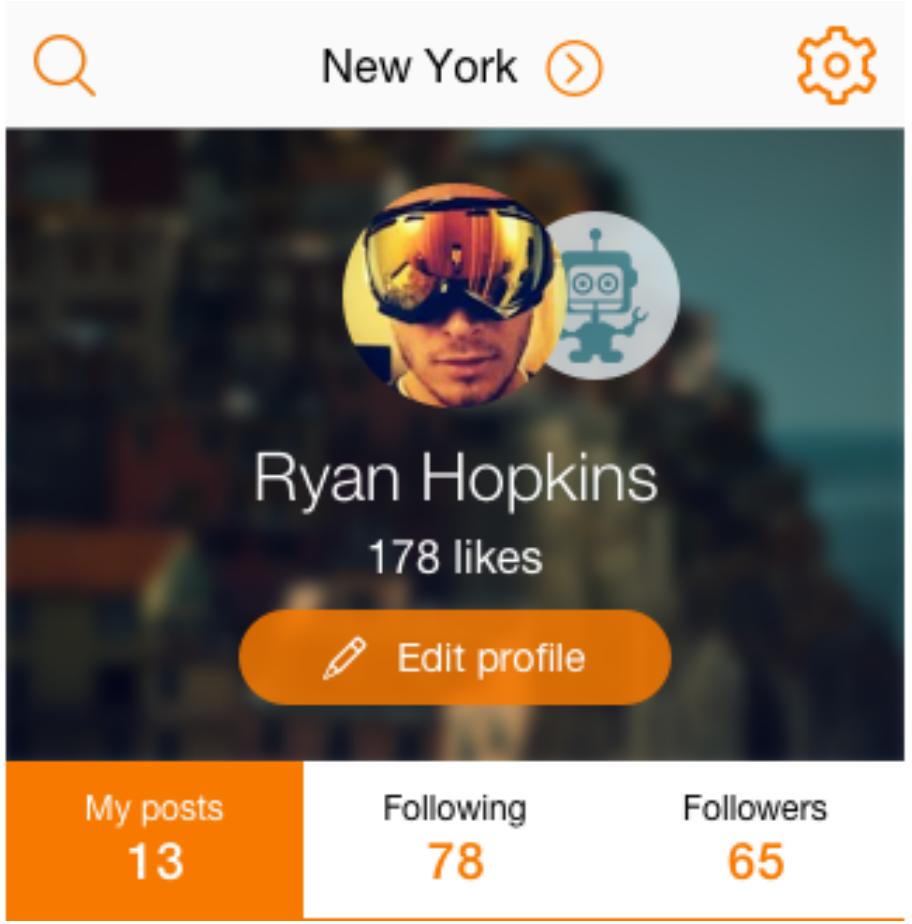
SYNTAXE XML

- **Les commentaires**
  - Commencent par <!-- et se terminent par -->
  - Peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise
    - *Faux : <parent* <! -- ceci n'est pas correct --> *>Bob </parent>*
    - *Vrai :*
      - <! -- ceci est correct -->
      - <enfant>Alice <! -- ceci est correct --> </enfant>

EXERCICE

EXERCICE (énoncé)

- Modéliser une liste d'utilisateurs sachant que chacun doit avoir la structure de données présentée dans la figure ci-contre:
  - Sous le format **XML**
  - Sous le format **JSON**



EXERCICE (solution)

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<users>
  <user id="1">
    <fullname>Ryan Hopkins</fullname>
    <likes>178</likes>
    <totalPosts>13</totalPosts>
    <totalFollowing>78</totalFollowing>
    <totalFollowers>65</totalFollowers>
    <imageUrl>http://sample.com/1.jpg</imageUrl>
    <city>New York</city>
  </user>
  <user id="2">
    <fullname>John Doe</fullname>
    <likes>17</likes>
    <totalPosts>29</totalPosts>
    <totalFollowing>13</totalFollowing>
    <totalFollowers>100</totalFollowers>
    <imageUrl>http://sample.com/2.jpg</imageUrl>
    <city>Paris</city>
  </user>
</users>
```

```json
{
    "users": [{
        "id": 1,
        "fullname": "Ryan Hopkins",
        "likes": 178,
        "totalPosts": 13,
        "totalFollowing": 78,
        "totalFollowers": 65,
        "imageUrl": "http://sample.com/1.jpg",
        "city": "New York"
        },
        {
        "id": 2,
        "fullname": "John Doe",
        "likes": 17,
        "totalPosts": 29,
        "totalFollowing": 13,
        "totalFollowers": 100,
        "imageUrl": "http://sample.com/2.jpg",
        "city": "Paris"
        }
    ]
}
```

Questions