Numériser le patrimoine I: standards et bonnes pratiques

Langages

Simon Gabay

Langages et vocabulaires

Langages

Il existe plusieurs langages de programmation

- langages de programmation
- langages de définition de données
- langages de requête
- langages de balisage

Langage de programmation

- Permet de formuler des algorithmes et produire des programmes informatiques qui les appliquent

> Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat.

- C'est ce qui fait fonctionner l'ordinateur et les logiciels de votre ordinateur
- Exemples de langage de programmation: C, C++, R, Python, JavaScript
- Ainsi MacOS ou Linux sont des systèmes UNIX, qui est écrit en C. Windows aussi est écrit en C.

```
import math
# Assign values to x and n
x = 4
n = 3

# Method 1
power = x ** n
print("%d to the power %d is %d" % (x,n,power))

# Method 2
power = pow(x,n)
print("%d to the power %d is %d" % (x,n,power))

# Method 3
power = math.pow(2,6.5)
print("%d to the power %d is %5.2f" % (x,n,power))
```

Petite opération arithmétique en python.

Langage de définition de données (*data definition language*, DDL)

- Manipuler les structures de données d'une base de données, et non les données elles-mêmes
- Dans un tableur (par ex., excel), cela reviendrait à définir le nombre de colonnes et de lignes, ainsi que le domaine des données
- Exemple: SQL pour les bases relationnelles, RELAX NG pour le XML

```sql
-----------------------------------------
-- On crée une base
CREATE DATABASE filmographie;
USE filmographie;
-----------------------------------------
-- On crée une table de films
-----------------------------------------
CREATE TABLE films (
    id          INT(11)        NOT NULL AUTO_INCREMENT,
    titre       VARCHAR(50)    NOT NULL,
    sortie      DATE           NOT NULL,
    PRIMARY KEY (id)
);
-----------------------------------------
-- On crée une table de réalisateurs
-----------------------------------------
CREATE TABLE realisateurs (
    id         INT(11)        NOT NULL AUTO_INCREMENT,
    nom       VARCHAR(30)    NOT NULL,
    film_id    INT(11)        NOT NULL,
    INDEX (film_id)
    PRIMARY KEY (id)
);
```

TABLE FILMS

id	titre	sortie
.	.	.

TABLE REALISATEURS

id	nom	filmId
.	.	.

Langage de manipulation de données (*data manipulation language*, DML)

- Permettent de réaliser les traitements sur les données
- Dans un tableur (par ex., excel), cela reviendrait à remplir le tableau et aller chercher le contenu dans une table
- Exemple: SQL, SPARQL

SQL

```
-------------------------------------------
-- On remplit la table des films
-------------------------------------------
INSERT INTO films (titre, sortie)
  VALUES ('STAR WARS', '1977')
INSERT INTO films (titre, sortie)
  VALUES ('INDIANA JONES', '1981')
INSERT INTO films (titre, sortie)
  VALUES ('TITANIC', '1997')
-------------------------------------------
-- On remplit la table des réalisateurs
-------------------------------------------
INSERT INTO realisateurs (nom, film_id)
  VALUES ('GEORGE LUCAS', 1)
INSERT INTO realisateurs (nom, film_id)
  VALUES ('STEVEN SPIELBERG', 2)
INSERT INTO realisateurs (nom, film_id)
  VALUES ('JAMES CAMERON', 3)
-------------------------------------------
-- On fait une requête
-------------------------------------------
SELECT nom FROM realisateurs
```

Table relationnelle, relie les 2 tableaux

TABLE FILMS

id	titre	sortie
1	Star Wars	1977
2	Indiana Jones	1981
3	Titanic	1997

TABLE REALISATEURS

id	nom	filmId
1	George Lucas	1
2	Steven Spielberg	2
3	James Cameron	3

Le nom et le prénom pas séparés, on peut pas ordonner par nom

Problèmes:

- si on veut faire un classement aphabétique par le nom de famille?
- logique de "silo" dans le nommage

Exercice

```
SELECT titre FROM films
```

TABLE FILMS

id	titre	sortie
1	Star Wars	1977
2	Indiana Jones	1981
3	Titanic	1997

TABLE REALISATEURS

id	nom	filmId
1	George Lucas	1
2	Steven Spielberg	2
3	James Cameron	3

SPARQL

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?nom ?image ?description
WHERE {
       ?personne rdf:type foaf:Person.
       ?personne foaf:name ?nom.
       ?image rdf:type foaf:Image.
       ?personne foaf:img ?image.
       ?image dc:description ?description
}
```

1. `PREFIX` pour les espaces de nom
2. `SELECT` pour les variables
3. `WHERE` pour les clauses de contraintes

Langage de de balisage (*Markup language*)

- Spécialisés dans l'enrichissement d'information textuelle. Ils utilisent des balises, unités syntaxiques délimitant une séquence de caractères ou marquant une position précise à l'intérieur d'un flux de caractères
- Exemple: HTML, XML ou LaTeX

C'est le plus simple. Enrichissement des données textuelles:

Exemple de texte balisé en XML:

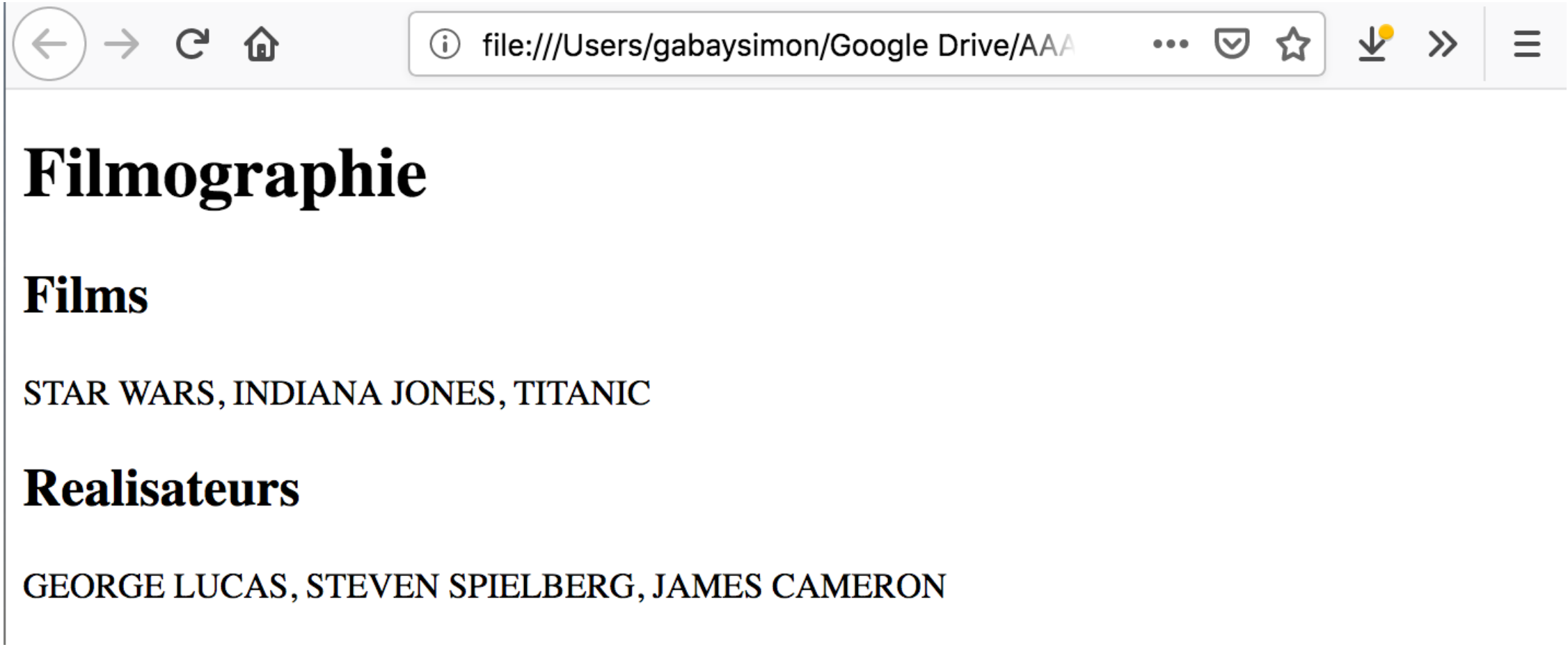
```
<doc>
    <partie>Filmographie</partie>
    <sous-partie>Films</sous-partie>
    <contenu>STAR WARS, INDIANA JONES, TITANIC</contenu>
    <sous-partie>Réalisateurs</sous-partie>
    <contenu>GEORGE LUCAS, STEVEN SPIELBERG, JAMES
      CAMERON</contenu>
</doc>
```

Exemple de texte balisé en XML (l'absence de `</p>` est volontaire)

```
<html>
    <body>
        <h1>Filmographie</h1>
        <h2>Films</h2>
        <p>STAR WARS, INDIANA JONES, TITANIC
        <h2>Realisateurs</h2>
        <p>GEORGE LUCAS, STEVEN SPIELBERG, JAMES
          CAMERON</p>
    </body>
</html>
```

HTML

Ici la balise n'est pas fermée. Si l'erreur est en XML tout beugue. Le but de HTML c'est d'affficher des choses en ligne.



Rendu du HTML dans le navigateur

Vocabulaire: l’exemple de la TEI

Règles principales

Le langage de balisage fonctionne de manière simple

```
<élément attribut="valeur">donnée</élément>
```

1. Un `<élément>` est entre chevrons
2. Une `<balise>` doit être fermé `</balise>`
3. Une `<balise1>` ne doit `<balise2>` pas être croisé `</balise1>` avec un autre `</balise2>`
4. Une `<balise/>` peut être auto-fermante
5. Un `<élément>` peut porter un `@attribut` (noté avec un `@` en langage naturel, mais *pas* dans le code)
6. L'`@attribut` a une `"valeur"` (entre guillemets)

ça sert pour oser des questions à l'ordinateur

Sémantique et procédural

> On emploie *a priori* les italiques pour les locutions et termes empruntés à d’autres langues.

Procédural (en LaTeX: `\textit{a priori}`)

```
On emploie <italique>a priori</italique> les italiques
pour les locutions et termes empruntés à d’autres langues.
```

Sémantique (option I)

```
On emploie<locutionEtrangere>a priori</locutionEtrangere>
les italiques...
```

Sémantique (option II)

```
On emploie<latin>a priori</latin> les italiques...
```

Une question fondamentale

Comment choisir le nom des `<éléments>` et des `@attributs` ?

TEI (*Text Encoding Initiative*) syntaxe de XML

- Elle est créé en 1987 (donc avant internet)
- La TEI est pilotée par un consortium qui maintient et développe des recommandations pour l'encodage des textes
- Ces recommandations sont en constante évolution très lente
- Elles sont disponibles en ligne http://www.tei-c.org/guidelines/

D'autres vocabulaires que la TEI

- *Encoded Archival Description* (EAD) pour les archivistes
- *Dublin Core* (DC) pour les bibliothécaires
- *Translation Memory eXchange* (TMX) pour les traducteurs

Ces vocabulaires peuvent être exprimés avec différents langages:

- RDF-DC (RDF pour *Resource Description Framework*)
- Pour cette raison, on parle de XML-TEI, et il a existé un SGML-TEI (SGML pour *Standard Generalized Markup Language*)

La solution en TEI

> On emploie *a priori* les italiques pour les locutions et termes empruntés à d'autres langues.

```
On emploie<foreign xml:lang="la">a priori</foreign> les
italiques...
```

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <text>
      <body>
         <head>Filmographie</head>
         <div>
            <head>Films</head>
            <p>STAR WARS, INDIANA JONES, TITANIC</p>
         </div>
         <div>
            <head>Réalisateurs</head>
            <p>GEORGE LUCAS, STEVEN SPIELBERG, JAMES
              CAMERON</p>
         </div>
      </body>
  </text>
</TEI>
```

Valide vs bien formé

Valide (*valid XML document*) vs bien formé (*well-formed XML document*)

- *Bien formé* renvoie au langage et signifie que le document respecte les règles précédemment mentionnées (l'élément est entre chevron, une balise ouverte doit être fermée ...)
- *Valide* renvoie au vocabulaire et signifie que le document répond aux exigences de la TEI (d'où l'expression "valide contre TEI ALL")
- Un schéma, qui est une sorte de dictionnaire qui permet de contrôler que le vocabulaire est bien utilisé, et donc que le document est valide

Valide vs bien formé (suite)

- L'emploi d'un vocabulaire précis est une restriction du champ des possibles
- Un document bien formé n'est pas nécessairement valide
- Un document valide est nécessairement bien formé
- Un schéma permet de contrôler que le code est valide contre la TEI

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://www.tei-c.org/release/xml/tei/
custom/schema/relaxng/tei_all.rng"
            type="application/xml" schematypens="http://
relaxng.org/ns/structure/1.0"?>
```

Défauts de la TEI (et des vocabulaires génériques)

La TEI pose des problèmes

- Elle force à utiliser un standard, par définition générique, et qui ne convient pas forcément exactement à nos données
- Elle nécessite un apprentissage, notamment pour respecter le sémantisme du vocabulaire

Avantages de la TEI (et des vocabulaires génériques)

- Elle simplifie l'échange d'information (interopérabilité des données)
- Elle force à adopter de bonnes pratiques, notamment en ce qui concerne les métadonnées
- Elle donne accès à une communauté qui donne de l'aide…
- …et qui développe des outils disponibles pour tous!

Métadonnées

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
      <fileDesc>
         <titleStmt>
            <title>Example d'encodage TEI</title>
         </titleStmt>
         <publicationStmt>
            <p>Université de Genève</p>
         </publicationStmt>
         <sourceDesc>
            <p>Cours original</p>
         </sourceDesc>
      </fileDesc>
  </teiHeader>
```

Le fichier TEI complet

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://www.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_all.rng"
            type="application/xml" schematypens="http://relaxng.org/ns/structure/1.0"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
      <fileDesc>
         <titleStmt>
            <title>Example d'encodage TEI</title>
         </titleStmt>
         <publicationStmt>
            <p>Université de Genève</p>
         </publicationStmt>
         <sourceDesc>
            <p>Cours original</p>
         </sourceDesc>
      </fileDesc>
  </teiHeader>
  <text>
      <body>
         <head>Filmographie</head>
         <div>
            <head>Films</head>
            <p>STAR WARS, INDIANA JONES, TITANIC</p>
         </div>
         <div>
            <head>Réalisateurs</head>
            <p>GEORGE LUCAS, STEVEN SPIELBERG, JAMES
              CAMERON</p>
         </div>
      </body>
  </text>
</TEI>
```