INF600A: Laboratoire #5
Analyse (simplifiée) de fichiers XML

Jeudi 8 novembre 2018
13h30–15h30
PK-S1565

Un premier objectif de ce laboratoire est de vous familiariser avec l'utilisation des expressions régulières et des méthodes de *pattern-mathching* en Ruby. Un autre objectif est de vous familiariser avec la mise en œuvre d'un itérateur `each` sur une collection de façon à pouvoir ensuite utiliser les diverses méthodes du module `Enumerable` sur cette collection.

Pour obtenir le code à compléter :

```
$ git clone http://www.labunix.uqam.ca/~tremblay_gu/git/XML.git
```

---

1 Un bref aperçu de XML

XML est défini comme suit dans Wikipedia :[1]

> L'*Extensible Markup Language* (XML, «langage de balisage extensible» en français) est un métalangage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite «extensible» car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG… **Elle est reconnaissable par son usage des chevrons (<, >) encadrant les balises**. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations hétérogènes (interopérabilité).

[1] `http://fr.wikipedia.org/wiki/Extensible_Markup_Language`

Nous allons illustrer XML et les notions de **balises** et d'attributs à l'aide de deux exemples, présentés à la page suivante.

a. Figure 1 : La description d'un document.

   Chaque élément d'information est identifié par une balise ouvrante — `<NomBalise>` — et une balise fermante — `</NomBalise>`. Les éléments compris entre la balise ouvrante et la balise fermante peuvent être soit du texte simple (par ex., `The Pragmatic Programmer`, `Andy`, `Hunt`, `Dave`, etc.), soit d'autres balises (par ex., une balise `AUTEUR` contient une balise `PRENOM` et une balise `NOM`). Cela permet donc de générer des structures hiérarchiques **récursives**.

b. Figure 2 : La page d'accueil du site Web du cours INF600A, une page HTML, le HTML pouvant être défini par une application XML.

   - La présence de certains espaces blancs n'est pas significative, plus précisément, ceux qui suivent le nom de la balise. Par contre, aucun espace blanc n'est supposé apparaitre entre le chevron ouvrant «<» et le nom de la balise.
   - Une balise peut contenir des **attributs**, qui sont indiqués directement dans la balise elle-même, avant le chevron fermant «>», par exemple :
     - `<FRAMESET ROWS="10%,*">`
       - Balise ouvrante `FRAMESET` ;
       - Attribut `ROWS` avec valeur `"10%,*"` ;
     - `<FRAME NAME="Banniere" src="haut.html">` :
       - Balise ouvrante `FRAME` ;
       - Attribut `NAME` avec valeur `"Banniere"` ;
       - Attribut `src` avec valeur `"haut.html"` ;
   - Certaines balises sont «auto-fermantes». Par exemple, `<br/>` est une abréviation pour deux balises : une balise ouvrante et une balise fermante englobant **un texte vide**. Cette balise auto-fermante est donc équivalente à la suivante : `<br></br>`

```
<DOCUMENT>
  <TITRE>The Pragmatic Programmer</TITRE>

  <AUTEURS>
    <AUTEUR>
      <PRENOM>Andy</PRENOM>
      <NOM>Hunt</NOM>
    </AUTEUR>

    <AUTEUR>
      <PRENOM>Dave</PRENOM>
      <NOM>Thomas</NOM>
     </AUTEUR>
  </AUTEURS>

  <EDITEUR>Addison-Wesley</EDITEUR>

  <ANNEE>2000</ANNEE>
</DOCUMENT>
```

Figure 1: Une description XML d'un document (avec titre, auteurs, etc.) pour le livre « *The Pragmatic Programmer* » de Hunt & Thomas.

```
<HTML>
<HEAD>
<TITLE>INF600A Langages de script et langages dynamiques</TITLE>
</HEAD>


<FRAMESET ROWS="10%,*">
  <FRAME NAME="Banniere" src="haut.html">

  <FRAMESET COLS="20%,*">
  <FRAME NAME="Menu" src="menu.html" TARGET="Principal">
  <FRAME NAME="Principal" src="accueil.html">
  </FRAMESET>

</FRAMESET>

</HTML>
```

Figure 2: La page d'accueil (HTML) pour le site Web du cours INF600A.

2 Ce que vous devez faire

Vous devez développer, en complétant divers éléments qui vous sont fournis, une mise en œuvre pour un analyseur de fichiers XML.

Le rôle de cet analyseur est d'identifier les diverses balises qui apparaissent dans un fichier XML ainsi que les attributs qui leur sont associés. Pour chaque balise rencontrée, il faut donc identifier les attributs qui apparaissent dans l'utilisation de cette balise.

a. La classe `BaliseXML`, définie dans le fichier `balise-xml.rb`, permet de créer un objet pour une balise avec ses attributs (voir plus bas pour les trois façons de créer une balise).

   Vous devez compléter la mise en œuvre de la méthode `BaliseXML.les_attributs` pour qu'elle traite correctement **l'analyse des divers attributs.** Pour ce faire, vous devez aussi définir le motif approprié (`ATTRIBUT_ET_VALEUR`) dans le fichier `motifs.rb`.

   Les tests appropriés sont dans le fichier `balise-xml_test.rb`.

   La cible du `makefile` = `test_balise`

   Rappel : La méthode `MatchData#post_match` permet de récupérer la partie **qui suit** une balise identifiée correctement, partie qu'on peut ensuite analyser (⇒ boucle `while`).

b. La classe `FichierXML` dans le fichier `fichier-xml.rb` permet de créer **un flux de balises** à partir d'une source, qui peut être une série de lignes ou un nom de fichier.

   Vous devez compléter la mise œuvre de la méthode `FichierXML.each` qui permet de générer les différentes balises identifiées par l'analyse d'une source de données.

   Les tests appropriés sont dans le fichier `fichier-xml_test.rb`.

   La cible du `makefile` = `test_fichier`

   Indice : Même remarque que précédemment. Et utilisez la méthode `BaliseXML.parse`.

c. Vous devez ensuite utiliser des méthodes du module `Enumerable` pour obtenir diverses informations sur une série de lignes contenant des balises.

   Pour ce faire, vous devez compléter trois cas de tests, tests qui sont définis dans le fichier `enumerables_test.rb`.

   La cible du `makefile` = `test_enumerables`

   Indice : Voir les méthodes suivantes du module `Enumerable` : `uniq`, `flat_map`,[2] `group_by`, de même que la méthode `Array#to_h`.

[2] Notez que `f.flat_map { |x| ... } = f.map { |x| ... }.flatten`

3 Les fichiers fournis

- `balise-xml.rb` : Fichier qui définit une classe `BaliseXML`.

  Un objet de cette classe peut être construit de différentes façons :

  - En utilisant `new` et en spécifiant l'ensemble des propriétés de l'objet.
  - En utilisant `new` et en spécifiant les propriétés de l'objet à l'exception des attributs, lesquels peuvent ensuite être spécifiés avec la méthode `[]=(attribut, valeur)`.
  - En utilisant la méthode (de classe) `self.parse`, qui crée une balise en effectuant l'analyse d'une chaine — qui doit contenir une balise valide.

  Un objet `BaliseXML` possède aussi diverses méthodes (d'instance) pour obtenir les informations caractérisant la balise.

- `motifs.rb` : Divers motifs pour décrire une balise, dont `ATTRIBUT_ET_VALEUR` que vous devez compléter.

- `balise-xml_test.rb` : Des tests unitaires `MiniTest` pour les méthodes de la classe `BaliseXML`.

- `fichier-xml.rb` : Une classe `FichierXML` dont le constructeur est la méthode `balises`. Cette méthode est appelée en fournissant une source de lignes — qui peut être une chaine indiquant le nom d'un fichier texte contenant du XML, ou bien un tableau de chaines de caractères.

  Une fois un objet de cette classe créé, on utilise alors la méthode `each` pour énumérer les diverses balises rencontrées.

  Le fichier `fichier-xml.rb` peut aussi être utilisé comme un script, auquel cas ce sont les lignes provenant de `STDIN` qui sont traitées, comme l'illustre l'exemple suivant :

```
$ echo '<A  a1= "v1" a2 = "v2"> </A> <X x="x"/>' | ./fichier-xml.rb
<A a1="v1" a2="v2">
</A>
<X x="x"/>
```

- `fichier-xml_test.rb` : Des tests unitaires `MiniTest` pour les méthodes de la classe `FichierXML`.

- `enumerables_tests.rb` : Trois cas de tests utilisant des méthodes du module `Enumerable` pour obtenir diverses informations sur un flux de balises.

- `makefile` : Un fichier pour automatiser l'exécution des tests unitaires.

  Voir plus haut pour les principales cibles.