SAX & DOM

Préparé par
M.G. BELKASMI

Problème à résoudre

Comment manipuler un document XML dans un programme?

- très bas niveau : analyse syntaxique (*parsing*)
- bas niveau : validation
- niveau intermédiaire :
  - manipulations du document chargé en mémoire
  - sauvegarde d'un document
- haut niveau :
  - sérialisation (*binding*)
  - transformations d'un document XML
  - interprétation d'un format XML particulier : SVG, MathML, XSL:FO, … etc.

Analyse syntaxique

- **Principe** : lire le document et reconnaître les structures XML (balise ouvrante, attribut, etc.)
- Un grand standard : *Simple API for Xml* (SAX)
  - Un concurrent (très spécifique) : *Xerces Native Interface* (XNI)
- Avantages (du travail au niveau *parsing*) :
  - efficacité maximale
  - occupation mémoire minimale
  - très souple
- Inconvénients :
  - délicat à programmer
  - il faut tout faire !
- En général, la validation est déjà possible au niveau de SAX (ou de XNI)

Chargement en mémoire

- **Principe** : lire le document (un arbre), le placer en mémoire et fournir une API permettant sa manipulation
- Un grand standard : *Document Object Model* (DOM) du W3C
- Des concurrents :
  - JDOM (http://www.jdom.org) même esprit que DOM, mais spécifique à Java et plus simple
  - XOM (http://cafeconleche.org/XOM/) : tente de faire mieux que DOM et JDOM (pour Java)
  - Electric XML : intéressant mais non *open source*
- Avantages :
  - beaucoup de fonctions déjà définies
  - modifications, manipulations globales, etc.
- Inconvénient majeur : performances (mémoire et temps)

API Binding

- **Principe** : transformation “automatique” et réversible d’un document XML en un objet
- une forme de persistance pour les langages OO
- le futur pour de nombreuses applications
- Des solutions :
  - Java Architecture for XML Binding (JAXB)
  - CASTOR (http://castor.exolab.org) : permet le *binding* d’objets Java vers XML mais aussi la persistance par d’autres techniques (bases de données, LDAP, etc.)
  - ZEUS (http://zeus.enhydra.org/), toujours pour Java
- Point négatif : vision objet pure et dure (pas vraiment adapté à des documents XML textuels)

Autres API

Pour les opérations de bas niveau (i.e., chargement et validation), on trouve :

- Java API for XML Processing (JAXP) :
  - sorte de couche d'abstraction
  - comment charger et configurer un analyseur (SAX ou DOM)
  - depuis la version 1.1, comment lancer une transformation XSLT : TrAX (transformation API for XML)
- datatype : ajout de nouveaux types de données aux validateurs RELAX NG
- autres API disponibles, mais pour des tâches de plus haut niveau (par exemple RPC XML)

Introduction

- Les deux principales interfaces de programmation XML:
- DOM (*Document Object Model*), basé sur une représentation hiérarchique
- SAX (*Simple API for XML*), basé sur des déclencheurs (événements/action)

DOM & SAX

- L'API DOM:
  - Construit une représentation du document en mémoire sous forme d'arbre
  - Adaptée aux applications qui modifient ou traitent dans leur globalité un document.

API SAX :

  - Définit des *triggers* qui se déclenchent sur certaines balises.
  - Adaptée aux applications qui extraient de l'information d'un document