Projet de Java
Feed Calendar (partie 2)
Licence 3 Informatique

forax@univ-mlv.fr, finkel@univ-mlv.fr

But

Le but de ce projet est d'afficher le contenu d'un ensemble de flux RSS/Atom sur des calendriers Google Calendar.
Ce projet est décomposé en deux parties distinctes, la première partie consiste à proposer une architecture de classes et la seconde à écrire le programme respectant l'architecture proposée.

Ce que doit faire le projet

Le projet doit aller lire dans un fichier au format XML (nommé feed-calendar.conf) un ensemble de flux RSS/Atom (des URLs) puis pour chaque flux, créer un calendrier google calendar par flux (si celui-ci n'existe pas déjà) et placer l'ensemble des items du flux sur le calendrier (si ceux-ci n'existent pas déjà) en fonction de leur date de publication. De plus, il devra exister un calendrier général permettant de voir l'ensemble des autres calendriers.
Le logiciel devra supporter les flux RSS au format 0.91+ (0.91, 0.92 etc), 1.0 et 2.0 ainsi que les flux Atom 0.3 et 1.0.

Etude technique

Le projet possède quelques points techniques sur lesquels il faut faire attention.
Ici, j'ai volontairement retiré les codes de tests qui devraient apparaître dans une vraie étude technique, car après tout c'est votre travail. Voici les points que vous avez du aborder:

Une seule représentation abstraite de tous les types de flux

Doit-on avoir deux classes une par type de flux RSS/Atom ou une seule classe abstrayant les deux flux ?

On chercher à gérer de la même façon les deux types de flux. Et comme on ne cherche pas à effectuer des opérations différentes suivant le types de flux, une seule classe suffira.

Par contre, il faudra des décodeurs de flux spécifiques à chaque type de flux.

SAX/StaX ou DOM ?

La grosse différence est que DOM stocke l'ensemble des informations en mémoire, ici, on cherche à abstraite un flux donc toutes les informations d'un flux ne sont pas nécessaire. DOM n'est donc pas adapté. Stax est plutôt utile lorsque le format de fichier est assez rigide, ce qui n'est pas le cas ici. Par élimination, nous utiliserons donc SAX pour décoder les flux XML.
Il faudra juste faire attention à la gestion du texte, la méthode characters() de la classe org.xml.sax.ContentHandler peut être appelé plusieurs fois, il faut donc reconstituer la chaine de caractère en utilisant un StringBuilder.

Comment décoder un flux ?

Le problème n'est pas exactement de décoder le flux mais que le décodage du flux dépend de son type, les balises ne sont pas les mêmes suivant si c'est un flux RSS ou un Flux Atom, voir même entre les versions par exemple des flux RSS. Il faut donc d'abord décoder le type de flux avant de choisir quelles sont les balises intéressantes.
Comme les types de flux sont indiqués en début de fichier, il n'est pas nécessaire de faire deux passes, il faudra juste avoir un système qui, une fois le type de flux décodé, redirige les tags XML vers un décodeur dépendant du type de flux.
Voilà le tableau récapitulatif des attributs intéressant en fonction du type de flux et de sa version.

Description	Type de flux	Version	Attribut
Titre d'un flux ou d'un item	RSS/Atom	all	title
Description d'un flux ou d'un item	RSS/Atom	all	description
Date d'un item	RSS	v0.91	pubDate
	Atom	v0.3	modified
	Atom	v1.0	updated
Lien (URI) d'un item	RSS/Atom	all	link
Identifiant unique d'un item	RSS	v1.0	guid
	Atom	V0.3	id

Le problème de la description d'un item

La description d'un item peut elle même contenir des tags XML, par exemple pour RSS 1.0, un tag <item> à l'intérieur d'un tag <description> lui-même à l'intérieur d'un tag <item> est légal.

Il faut donc distinguer un tag en fonction des tags dans lesquels il est inclus.
Dans le cas particulier des flux, il suffit de savoir si lors du décodage, on est entrain de décoder un flux ou un item.

Le problème des dates

Le format des dates n'est pas identique pour les flux RSS et Atom, les flux RSS utilisent des dates codées avec la RFC 882 et les flux Atom des dates codées avec la RFC 3339. Il est intéressant de noter que les échanges de date avec Google Calendar se font aussi avec des dates codées avec la RFC 3339, c'est pourquoi l'objet utilisé pour représenter des dates sera l'objet com.google.gdata.data.DateTime.

Et si un item n'a pas de date ?

Certain flux possède une date, dans ce cas la date du flux pourra être utilisée. dans le cas contraire, l'item devra simplement être ignoré. Pour éviter de mélanger le décodage du flux avec le problème d'item n'ayant pas de date, après le décodage du flux, une passe sera faite sur l'ensemble des items pour les mettre en conformité avant de faire la synchronisation avec le calendrier Google Calendar.

Lire le fichier de configuration

Le fichier de configuration est aussi en XML, il peut être intéressant de mettre une classe abstraite intermédiare pour le décodage du XML. Ce n'est pas nécessaire ici car le fichier de configuration est simple, il n'est pas nécessaire de gérer le texte : seul les tags et leurs attributs contiennent des données. De plus, L'API SAX contient déjà une classe org.xml.sax.DefaultHandler qui servira de base pour le décodage du fichier de configuration et des flux XML.

La synchronisation d'un calendrier

Une fois le flux décodé, il faut mettre à jour le calendrier Google correspondant, il faut donc savoir si ce calendrier existe déjà et, sinon, le créer. Si le calendrier existe déjà, il y a de grande chance que certains items existent déjà dans le calendrier, il faut donc être capable d'identifier un item de façon unique (avec son id) et mettre celui-ci à jour (sa description, etc) s'il existe déjà dans le calendrier.
Donc pour pouvoir faire cette synchronisation avec les items du calendrier, il faut être capable de récupérer les items d'un calendrier particulier et de mettre à jour ce même calendrier, ce qui n'est pas si facile que ça à faire avec l'API Google Calendar.
Enfin, si un calendrier à beaucoup d'items, récupérer les items trop vieux ne sert à rien et ralentira beaucoup l'application. il est préférable de choisir un temps dans le passé à partir

duquel on regardera les items pour éviter de prendre tous les items. Il faut donc être capable de récupérer les items d'un calendrier donné sur une période donnée. Enfin, comme il risque d'y avoir beaucoup de mises à jour et d'ajouts d'item, il est préférable d'utiliser l'API de traitement par lots (batch) de Google Calendar.

Architecture

On peut voir 3 parties dans le projet:

1.la gestion des flux (fr.umlv.feedcalendar.feed)

2.la gestion des calendriers Google Calendar (fr.umlv.feedcalendar.calendar)

3.la gestion de la configuration (fr.umlv.feedcalendar.config)

La gestion des flux

Comme indiqué dans l'étude technique, les flux RSS et Atom sont représentés par la même classe Feed. Un flux possède un type de flux qui indique format de flux RSS ou Atom (FeedFormat) et aussi le version du format. La version est nécessaire car il est possible d'implanter une *query* sur la version de flux.

La classe Item représente un Item. Un flux connait l'ensemble des items qui lui sont associés et un item connait le flux auquel il appartient. Ce dernier lien permet si un item n'a pas de date de regarder si le flux de l'item ne contient pas une date.

Comme les flux et les items possèdent des attributs communs, ceux-ci ont été regroupés dans la classe abstraite AbstractInfos.

Le decodage des flux

La classe XMLFeedHandler est responsable du decodage des flux XML au niveau de l'API SAX.
Lors de la construction de cette classe, un FeedVisitor est pris en paramètre. Un FeedVisitor fonctionne de la même façon qu'un handler SAX, la méthode startFeedElement() est appelée lorsqu'un tag ouvrant est trouvé et la méthode endFeedElement() est appelée pour les tags fermants correspondants.
Ces méthodes prennent le flux courant (celui qui est en train d'être décodé) en paramètre. Si un FeedVisitor reconnaît un tag ouvrant comme étant un tag de début d'un item, il renvoie un ItemVisitor en valeur de retour de la méthode startFeedElement(). Les méthode startItemElement()/endItemElement() seront alors appelées pour l'ensemble des tags contenus dans le tag de début d'item. L'itemVisitor ne sera plus utilisé après la balise fermante du tag pour lequel il a été renvoyé.

Le décodage du texte à l'intérieur d'une balise est de la responsabilité du

XMLFeedHandler qui reconstitue la chaine initiale et passe celle-ci lorsque la balise fermante est atteinte, donc en argument des méthodes endFeedElement()/endItemElement().

L'interface FeedVisitor contient en plus deux méthodes startPrefixMapping/endPrefixMapping qui permettent de savoir les mappings de namespace (xmlns) contenus dans le document.
Ceci permet par exemple de détecter précisément la version d'un document RDF comme le RSS 1.0.

Le FeedRecognizerVisitor qui est un FeedVisitor détecte le type de flux, choisit en fonction du type de flux le FeedVisitor qui devra être utilisé et redirige ensuite toute les balises suivantes vers le FeedVisitor installé.

L'implantation proposée contient deux FeedVisitor, un pour les flux RSS nommé RSSFeedVisitor et un pour les flux Atom nommé AtomFeedVisitor. Ces deux flux sont aussi des ItemVisitor et donc lorsqu'il détecte la balise de début d'un item se renvoie eux même pour agir en tant qu'ItemVisitor.
Cette architecture permet d'avoir autant d'implantations de FeedVisitor et/ou ItemVisitor que l'on veut.

La classe FeedChercker permet de tester qu'un flux possède des items valides affichant sur un log les erreurs constatées. Les validations consistent à s'assurer qu'un item possède une date (en essayant de lui en trouver une le cas échéant), un lien, une description.

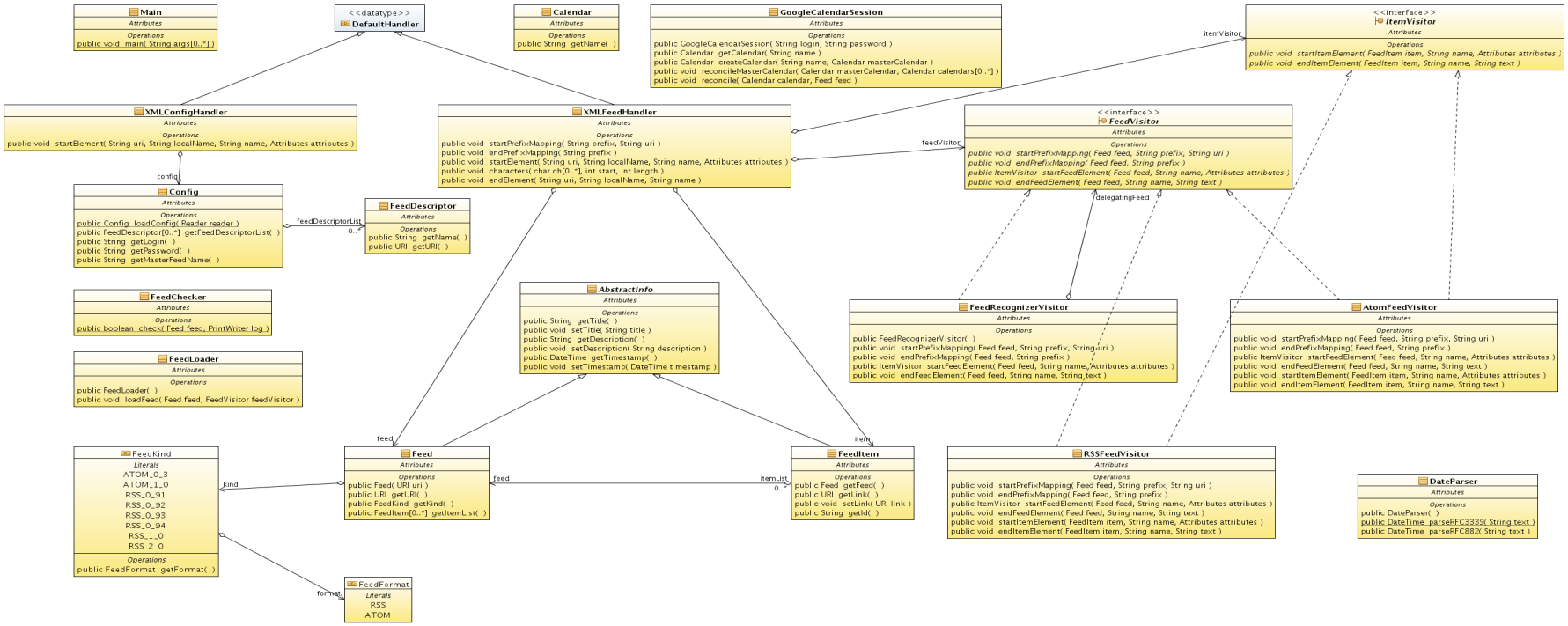
La gestion des calendriers

La classe GoogleCalendarSession permet de gérer des données sur plusieurs calendriers Google Calendar représenté par des objets Calendar.
La méthode getCalendar() permet de récupérer un calendrier (même le master calendar) déjà existant. La méthode createCalendar() permet de créer un calendrier (même la master calendar). La méthode reconcileMasterCalendar permet à partir du master calendar d'importer les autres calendriers. La méthode reconcile permet d'ajouter/mettre à jour un calendrier par rapport à un flux.

La gestion de la configuration

Comme XMLFeedHandler, la classe XMLConfigHandler s'occupe de décoder le fichier de configuration en utilisant l'API SAX. Les données décodées sont stockées dans un objet Config. La configuration de chaque flux est stockée dans un objet FeedDescriptor et l'ensemble des FeedDescriptor peut être obtenu à partir de l'objet Config.

Diagramme des classes du feedcalendar



Seconde partie du projet

La seconde partie du projet consiste à :
•implanter l'architecture proposée
•ajouter le code permettant la gestion des *queries*
•écrire une javadoc digne de ce nom en anglais et compréhensible.
•packager votre code comme indiqué dans les conditions de rendu.

Pour la gestion des *queries*, vous devrez porter un soin tout particulier à la complexité des parcours que vous implanterez pour appliquer les *query*.
En aucun cas, vous ne devrez ajouter de méthodes publiques à l'architecture décrite dans ce document.

Le projet doit être rendu au plus tard le **16 fevrier à 23h59**.

Le format XML du fichier de configuration

Ce format est le même à l'ensemble de vos projet, vous devez donc porter un soin particulier pour en respecter la syntaxe et la sémantique (son sens).
Ce fichier devra contenir les balises :
•**feed-calendar** qui est la balise englobante
•**calendar** qui définie l'accès aux calendriers (Google Calendar)
master-calendar correspond au calendrier à partir duquel tous les autres sont accessibles
•**feeds** qui contient la liste de tous les flux
•**feed** qui décrit un flux avec son nom de calendrier et son URL.

•**queries** qui contient un ensemble de query
•**query** qui définie une requête qui sera appliquer sur un item ou un flux et qui en fonction d'une propriété et d'un filtre effectuera une action pouvant avoir des paramètres
•**selectall** définie l'ensemble d'objet sur lequel le filtre sera testé
object definie le type d'objet (item ou feed)
property la propriété sur lequel le filtre s'applique, les propriétés des objets correspondent aux méthodes get possible sur cette objet.
Pour la propriété xxx, il doit exister une méthode setXxx() dans l'objet correspondant.
filtre:

- •**contains**: vrai si la propriété de l'objet contient l'expression régulière
- •**equals**: vrai si la propriété de objet est vérifié par l'expression régulière

•**apply** applique sur l'ensemble des items une action avec des paramétres

action action à effectuer parmi :

- **change-color** change la couleur des items qui respecte le filtre
- **gadget** applique un gadget aux items qui respectent le filtre.

Un exemple complet :

```
<?xml version="1.0"?>
<feed-calendar>
 <calendar login="login@gmail.com" password="password">
  <master-calendar name="All Feeds"/>
 </calendar>
 <feeds>
  <feed calendar-name="planetJDK" url="http://planetjdk.org/feed.atom-1.0"/>
  <feed calendar-name="planetarium"
url="http://blogs.sun.com/theplanetarium/feed/entries/atom"/>
  <feed calendar-name="linux" url="http://www.linux.com/feature?theme=rss"/>
 </feeds>
 <queries>
  <query>
   <selectall object="item" property="description" contains="java|Java"/>
   <apply action="change-color" parameters="#f00"/>
  </query>
 </queries>
</feed-calendar>
```

Conditions de rendu

Le projet est à rendre par mail aux chargés de TD et à l'enseignant de cours. Le format de rendu est une archive au format zip (**tout rar, tar.gz, 7z et autre ne sera pas ouvert**) contenant:

- un répertoire src contenant les sources du projet
- un répertoire docs contenant le manuel de l'utilisateur (user.pdf) et le manuel qui développeur (dev.pdf) au format **PDF**;
- un répertoire docs/api **vide** dans l'archive et qui contiendra la javadoc
- un répertoire classes **vide** dans l'archive et qui contiendra les classes **une fois compilées**
- un jar exécutable feedcalendar.jar qui fonctionne avec java -jar feedcalendar.jar et donc qui possède un fichier manifest adequat;
- un build.xml qui permet de
  - compiler les sources (target compile)
  - créer le jar exécutable (target jar)
  - compiler et créer l'exécutable (target all)
  - générer la javadoc dans docs/doc (target javadoc)
  - nettoyer le projet pour qu'il ne reste plus que les éléments demandés (target clean)

- un répertoire lib contenant les bibliothèques dont a besoin votre projet pour fonctionner.

Cette archive zip aura comme nom Nom1Nom2_feedcalendar.zip, où les noms sont ceux des membres du binôme par ordre alphabétique. L'extraction de cette archive devra créer un répertoire de nom Nom1Nom2_feedcalendar pour contenir tous les éléments demandés ci-dessus. Elle sera envoyée par mail aux enseignants de TD (adresse en haut de ce document) avec dans le corps du message les noms et prénoms de chacun des membres du binôme.

Notation

- Cas de 0 sans aucune correction:
    - projet n'est pas fait par les mêmes binômes pour la partie 1 et 2.
    - projet envoyé après la date
    - projet ne compile pas
    - projet non envoyé à TOUS les chargés de TDs (AUX BONNES ADRESSES !)
    - mail ne contenant pas les noms et prénoms de chacun des membres du binôme
    - réception d'une archive qui n'a pas comme nom Nom1Nom2_feedcalendar.zip, où les noms sont ceux des membres du binôme par ordre alphabétique
    - fichier d'archives dont l'extraction ne produit pas un répertoire dont le nom est feedcalendar_GalaxMLV
    - fichier d'archive foireux (vérifiez le et mettez vous en copie du mail)
    - l'absence de Javadoc (une seule méthode/classe suffit)
    - le jar feedcalendar.jar n'est pas exécutable
- Base de la notation:
    - le code marche et fait ce qu'il faut
    - la propreté et la lisibilité du code auront un poids très important dans la note
    - la présence de code inutile
    - les différents rapports et, par conséquent, l'orthographe !
    - la soutenance

Le document concernant la première partie est à envoyer par mail aux adresses (notez le pluriel) indiquées au début de ce document au plus tard le lundi 15 décembre 23h59 au format PDF.

Références :

Eclipse :

www.eclipse.org
http://www.forax.org/ens/java-avance/cours/pdf/Eclipse%20pour%20les%20null.pdf

Accéder à une ressource à partir d'une URL :
les classes java.net.URI, java.net.URL et java.net.URLConnection

Flux RSS :

http://en.wikipedia.org/wiki/RSS

Flux Atom :

http://en.wikipedia.org/wiki/Atom_(standard)

Parseur XML (SAX/DOM) :

http://java.sun.com/javase/6/docs/technotes/guides/xml/index.html

Expression régulière en Java (piquée à Perl) :
http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

ANT
http://ant.apache.org/