Flipboard Like HTML5 App – Aggregator Assembly

1. Project Overview

Client wants to develop a Hybrid mobile application built in HTML5/JS/CSS, which will be deployed to iOS and Android - Phone & Tablet sized.
This mobile application will aggregate RSS data from external sources, social media content, and internal content served via REST API, similar as Flipboard.
For internal content, we'd like to seek an existing content management sytem that fits our client's requirement to reduce the development effort.

In this module assembly, we will build the aggregators that crawls the external RSS and the internal CMS, and saves the articles (and videos) in the article table.

2. Requirements

The Aggregators Class Diagram is in scope.

2.1. External Aggregator

This aggregator is used to crawl external feeds and save them to article table. Note that the feed can be any of RSS 1.0, RSS 2.0 or Atom feed.
See http://www.mnot.net/rss/tutorial/#Versions on how these feeds can be differentiated.

Implementation Details:

1. Load all the entires in the external_feed table. We need only the feed_url and last_crawl_timestamp columns.
2. For each feed, visit the RSS link and start reading the RSS. The RSS reading should be done using the SAX model. Through example of reading RSS using SAX is provided at http://www.vogella.com/tutorials/RSSFeed/article.html. (see section 4.1)

   Pseudo-code for the above link is provided below:
   ```
   Create an XMLEventReader using the feed XML
   while eventReader.next()
      if event is start element
         if element is item, create a new ArticleHeader object
         if element is pubDate, set pubDate of ArticleHeader
         object. If pubDate is before last_crawl_timestamp, then
         break out of while loop
         If element is title, set title of ArticleHeader object
         If element is description, set description of
         ArticleHeader object
         …..so on for other child elements of item
      if event is end element and element is item
         save ArticleHeader object to database.
   ```

3. For each item element in the RSS (or entry element in Atom):
   3.1. If the date field is not available, then ignore the item. See the mapping for the date field in the table below.

3.2. Read the properties of the item and use them to save a new entry in the article table. The mapping between article table and the fields in various formats is given below:

RSS 2.0	RSS 1.0	Atom	article table column
item/title	item/title	entry/title	title
item/description	item/description	entry/summary	description
item/author	dc:creator	entry/author/name	author
item/pubDate	dc:date	entry/updated	pubDate
item/link	item/link	entry/link	link

3.3. is_internal field will be set to 0 and category_id will be same as the feed's category_id.
3.4. If we see a date field that is before or on the last_crawl_timestamp, then we stop processing for that feed. And we set the last_crawl_timestamp to item with the maximum timestamp.

2.2. Internal Aggregator

This aggregator is used to crawl internal CMS posts (normal and video) and store them in the article table.

Implementation Details:
1. Load the last_crawl_timestamp from the internal_cms_crawl table
2. Connect to the WordPress MySQL database.
3. In the wp_posts table, find all records WHERE post_status = publish AND post_type IN ('page', 'post') AND post_date > last_crawl_timestamp. These are the normal posts that need to be added to the article table
4. In the wp_posts table, find all records WHERE post_type = 'attachment' AND post_mime_type LIKE 'video%' AND post_date > last_crawl_timestamp. These are the video posts that need to be added to the article table
5. The mapping of the WordPress wp_posts columns to our DB article table columns is as follows:

wp_posts	article
post_title	title
post_excerpt	description
select display_name from wp_users where ID = post_author	author
post_date	pubDate
guid	link
The post category name is the one in wp_terms JOIN wp_term_taxonomy ON term_id JOIN wp_term_relationships ON term_taxonomy_id where object_id = post id.	category_id (find the one with the same name as found on the left)
0 or 1. see points 3 and 4 above	is_video

wp_posts	article
1	is_internal
ID	internal_wp_id

6. Update the last_crawl_timestamp in the internal_cms_crawl table to the maximum post timestamp.

Also check if any of the already crawled posts/videos have since been deleted in WordPress, and if so, delete them in our database as well.

Implementation Details:

1. For each article where is_internal = 1
   - 1.1. Check WordPress wp_posts table to see row exists such that wp_posts.ID = article.internal_wp_id is found and wp_posts.publish_status != 'trash'
   - 1.2. If not, then delete the article record.

2.3. Script for cron job

The above mentioned code will be packaged into a jar. And a cron job script must be provided for running the crawlers from the jar.

2.4. Other Requirements

See ADS section 2 for cross cutting requirements like logging, configuration, exception handling, internationalization etc.

2.5. Submission Deliverables

- Source Code
- Deployment Guide to verify the submission.

2.6. Technology Overview

2.6.1. Application Technologies

- Java 7
- RSS 2.0 - http://cyber.law.harvard.edu/rss/rss.html
- RSS 1.0 - http://web.resource.org/rss/1.0/spec
- Atom - http://www.ietf.org/rfc/rfc4287.txt
- XML
- SAX
- MySQL

3. Project Dependencies

3.1. Assemblies

None

3.2. Components

None

3.3. Third Party Libraries

See section 1.4.1

4. Project Deliverable Details

4.1. Source code setup

Standard TopCoder Assembly Contest source code setup

4.2. Build Setup

Standard TopCoder Assembly Contest ant based build setup

5. Final Submission

- For each member, the final submission should be uploaded to the Online Review Tool.
- The final submission will be reviewed using the standard Online Review Assembly Scorecard.