http://webdam.inria.fr

Web Data Management

Putting into Practice: Mashups with YAHOO! PIPES and XProc

Serge Abiteboul
INRIA Saclay & ENS Cachan

Ioana Manolescu
INRIA Saclay & Paris-Sud University

Philippe Rigaux
CNAM Paris & INRIA Saclay

Marie-Christine Rousset
Grenoble University

Pierre Senellart
Télécom ParisTech

Contents

Mashups are Web applications that integrate and combine data from multiple Web sources to present them in a new way to a user. This chapter shows two different ways to construct mashup applications in practice: YAHOO! PIPES, a graphical user interface for building mashups, and XProc, a W3C language for describing workflows of transformations over XML documents. Pros and cons of either approach will be made clear as one follows the indicated steps. The goal will be to present information about news events, each event being accompanied by its localization displayed on a map. For that purpose, we integrate three sources of information:

1. A Web feed about current events in the world, in RSS format (e.g., CNN's top stories at `http://rss.cnn.com/rss/edition.rss`). Any such RSS feed is fine, though English is preferable to ensure precision of the geolocalization.

2. A geolocalization service. We use information from the GeoNames[1] geographical database, and specifically their RSS to GeoRSS converter, whose API is described at `http://www.geonames.org/rss-to-georss-converter.html`.

3. A mapping service. We use Yahoo! Maps[2].

1 YAHOO! PIPES: A Graphical Mashup Editor

YAHOO! PIPES[3] allows creating simple mashup applications (simply called *pipe*) using a graphical interface based on the construction of a pipeline of boxes connected to each other, each box performing a given operation (fetching information, annotating it, reorganizing it, etc.) until the final output of the pipeline. It can be used by non-programmers, though defining complex mashups still requires skill and experience with the platform. The mashup we want to build is demonstrated at `http://pipes.yahoo.com/webdam/geolocalized_news`: it asks the user for a feed URL, and displays with markers on a map the result of the geolocalization of each news item.

1. Go to the YAHOO! PIPES website and either log in using an existing Yahoo! account or create a free account. Once you follow the links for creating a pipe, you will be presented with the interface of the graphical editor: on the left, a list of all boxes that can be used inside a pipe; in the center, the workspace where you can build your pipe; in the bottom part, a debugger shows the output of the currently selected box.

2. Drag a "Fetch Feed" box on the workspace. Enter the URL in the box and connect it to the "Pipe Output" box at the bottom of the workspace by dragging a link from the output port of the initial box (shown as a circle on its bottom border) to the input port of the final box. Save your pipe and click on "Run pipe. . . " to see the result.

[1]`http://www.geonames.org/`
[2]`http://maps.yahoo.com/`
[3]`http://pipes.yahoo.com/`

3. We are going to add some geolocalization information by using the "Location Extractor" operator of YAHOO! PIPES, which should be put in the middle of the two existing boxes. Save and run the pipe.

4. The location extractor of YAHOO! PIPES is not always as precise or complete as GeoNames. Study the documentation of the RSS to GeoRSS converter REST API. Use this API by trying to form URLs directly in your browser until you fully understand how it works. Then integrate it into your pipe by using a "URL Builder" whose output port is connected to the *url* parameter of the existing "Fetch Feed" box. Compare the results to what you had before.

5. To give a final touch to your pipe, add a "URL input" box to ask the user for the URL of the feed to be geolocalized. Save and test.

You can decide to publish your pipe to give other users access to it; if you want to keep playing with YAHOO! PIPES, you can try enriching your pipe by retrieving data from multiple RSS feeds, using a Web search operator to discover feeds dealing with a given topic, adding to feed items images obtained by querying Flickr with keywords from the description of the item, and so on. You can also look at the vast library of published pipes to get some inspiration.

2 XProc: An XML Pipeline Language

XProc is a W3C Recommendation for describing transformation workflows on XML documents. Throughout this section, refer to the XProc specification[4] for more detail about the language. As with YAHOO! PIPES, a workflow is seen as a pipeline of operations (here called *steps*) that fetch or process information; these operations heavily rely on other XML standards (XPath, XSLT, XInclude, XML Schema, XQuery, etc.). In YAHOO! PIPES, connections between boxes are described in a graphical manner; in XProc they are described using an XML syntax. Finally, contrary to YAHOO! PIPES, which deals with Web data at large, XProc is dedicated to the processing of XML data only. Any XProc processor can be used; we recommend XML CALABASH[5], a Java implementation that is easy to install and to use.

1. Download the skeleton pipeline `skeleton.xpl` from the book website, `http://webdam.inria.fr/Jorge/`. Test your XProc processor; if you use XML CALABASH and its installation directory is in your path environment variable, you can just type

   ```
   calabash skeleton.xpl > result.html
   ```

   No error message should show (only information messages), and `result.html` should contain a basic view of CNN's feed.

2. Look at the `skeleton.xpl` file. The whole pipeline is described inside a top-level `<p:pipeline>` element. First, a variable is declared; declared variables can be used in XPath expressions further in the file (all values of `select` attributes are XPath expressions). Then the RSS file is loaded with the help of the standard `<p:load>`

[4]`http://www.w3.org/TR/xproc/`
[5]`http://xmlcalabash.com/`

step (again, see the XProc specification for the definition of standard steps). All items of the RSS feed are put into a sequence (`<p:for-each>`), and this sequence is then wrapped under a common `item` element (`<p:wrap-sequence>`); these two steps are arguably not very useful, but this structure will help us in extending this pipeline. Finally, an inline XSLT stylesheet is reformatting the list of items into a table, where each line has a single cell, containing the title of the item and pointing to the corresponding article.

3. Change the `<p:load>` so that a geolocalized version of the RSS feed is loaded instead of the original one. Once again, refer to the documentation of the API of GeoNames to determine which URL to load. You can use the XPath 2.0 function *encode-for-uri*() to properly encode special characters in a URL.

4. Items should now have `geo:lat` and `geo:long` child elements with geolocalization information. Test this by adding in the XSLT stylesheet, after the item's title, two `<xsl:value-of>` elements that show both coordinates. Test.

5. We now want to filter out items that do not have any geolocalization information (if any). For this purpose, you can modify the `select` attribute of the `<p:iteration-source>` to keep only items with `geo:long` and `geo:lat` child elements.

6. We will use the Yahoo! Maps Image API[6] to add a map for each news item. Carefully study the API documentation and apply for a Yahoo! Application ID.

7. Replace the `<p:identity>` step with a `<p:load>` step that calls the Yahoo! Maps Image API appropriately. Remember you can use any XPath expression inside a `select` attribute. In the XSLT stylesheet, add a cell:

   ```
   <td><xsl:value-of select="." /></td>
   ```

   before the existing cell to display the URL of the map image. The display of the title and link to the article does not work any more because we discarded the news items to keep only the map image. We are going to fix this later on.

8. Replace the display of the URL by an HTML `<img>` element that loads this URL. In XSLT, to input an XPath expression inside an arbitrary attribute, surround the XPath expression with curly braces.

9. To keep in the sequence both map images and information about news items, you will need two `<p:for-each>` steps and a `<p:pack>` step to combine the two sequences. Refer to the XProc specification. The `<p:pack>` step will introduce an extra wrapping element, so remember to adapt the XPath expressions used in the XSLT stylesheets.

[6] `http://developer.yahoo.com/maps/rest/V1/`