

Topics in Operating Systems (Saya)

Mini-Project

Extraction of contents from RSS

FINAL REPORT

Ilan Shtokhamer & Yuval David

Table of contents

Introduction.....	3
Implementation.....	4
Running Example.....	8
About Rome.....	10
Bibliography.....	11

Introduction

Saya is robot developed by Hiroshi Kobayashi, an associate professor at the Tokyo University of Science, which was purchased by the Department of Computer Sciences to serve as a learning and research tool.

Her lingual abilities span the use of Japanese, English, and now possibly Hebrew.

The robot's face, expressions, her first name, and the personal attitude all contribute to the final product, a human friendly communicative robot. She has a human-like face and can express facial expressions similar to human beings.

Saya is a robotic receptionist, located in the new "Alon" Building (37). She was brought to us as a departmental effort, a project led by Prof. Shlomi Dolev, including Prof. Eyal Shimony, Head of the Paul Ivanier Center for Robotics Research and Production Management; Prof. Matya Katz, Chairman of the Department and Dr. Ohad-Ben Shahr.

Our goal is to provide simple objects and functions for getting RSS information from the World Wide Web, where the purpose is to enrich **Saya** with more ways to communicate with people and supply real "live" information.

Implementation

We implemented the functions using Java.

Our project is composed of three classes:

1. [Rss_Info.java](#) - This class includes the functions for getting the processed RSS information.

There are three ways to get the RSS information:

- As a vector of Strings.
- As an Iterator of Strings.
- As a simple Array of String.
-

2. [Build_Data.java](#) - This class contains the technical implementation for getting the RSS information out of the world wide web.

3. [Make_URLs.java](#) - This class is used to store the url addresses from which RSS information is brought. It is not obligatory to use those pre-defined urls. There are functions which get a url address. This class is also an easy platform to edit and add urls and categories of information.

Detailed description of the classes and their functions:

1. [Rss_Info.java](#) (implements `Rss_inter`)

- ✚ **Constructor 1** - `public Rss_Info(String URL_Addr, Vector<String> key_words)` - Gets a URL address where the RSS is located and a **vector of key words** to be found in the RSS. The RSS is filtered using these key words. Use one of: `Get_Rss_Vector()`, `Get_Rss_Iterator()`, `Get_Rss_Array()` (explained later on) in order to retrieve the RSS in the way you prefer.
- ✚ **Constructor 2** - `public Rss_Info(String URL_Addr, String[] key_words)` - Gets a URL and an **array of key words**. Usage is the same as Constructor 1.
- ✚ **Constructor 3** - `public Rss_Info(Vector<String> key_words, String Category)` - Gets a **vector of key words** and a Category string, which is one of the next three: 'soccer', 'economy' or 'politics'. By choosing a category, the corresponding URL is retrieved by using [Make_URLs.java](#) and then as described before, the RSS is filtered using the key words.
- ✚ **Constructor 4** - `public Rss_Info(String[] key_words, String Category)` - Gets an **array of key words** and a Category string. Same explanation as for Constructor 3.
- ✚ `public Vector<String> Get_Rss_Vector()` - A vector of RSS information which corresponds to the key words provided by the user, is returned. Each item in the vector is a 'news item' on its own that can be fed into Saya.

✚ `public Iterator<String> Get_Rss_Iterator()` -
 An iterator of RSS information which corresponds to the key words. Same explanation as before.

✚ `public String[] Get_Rss_Array()` -
 An array of RSS information which corresponds to the key words. Same explanation as before.

2. Build_Data.java

✚ **Constructor** - `public Build_Data(String URL_Addr, Vector<String> Key_Words)` - Gets a url address and a vector of key words. The RSS on the specified url address is parsed using code taken from the 'code ROME library'.

The field 'entries' of type 'vector' is filled with RSS information taken from the url (line by line). Afterwards the fiels 'good_entries' is filled with entries after the irrelevant RSS items had been filtered using the function: 'clearIRRelevantEnts'.

✚ `public void clearIRRelevantEnts(Vector<String> Key_Words)` -

This function filters the entries of RSS which don't agree with the key words provided to the constructor. In this way, only relevant RSS items are left in the vector 'good_entries'.

3. Make_URLs.java

✚ **Constructor** - `public Make_URLs()` - Simply builds three url strings in the following categories: Soccer, Politics, Economy.

- ✚ `public String` `getUrls_Soccer()` - Returns the url string which corresponds to rss of **Soccer**.
- ✚ `public String` `getUrls_Politics()` - Returns the url string which corresponds to rss of **Politics**.
- ✚ `public String` `getUrls_Economy()` - Returns the url string which corresponds to rss of **Economy**.

4. `Rss_inter.java`

An interface for the functions:

- ✚ `public` `Vector<String>` `Get_Rss_Vector();`
- ✚ `public` `Iterator<String>` `Get_Rss_Iterator()`
- ✚ `public` `String[]` `Get_Rss_Array();`

Running Examples – Main.java

Example No. 1 :

```
import java.util.Iterator;
import java.util.Vector;

public class Main {

    public static Vector<String> URL_Addrs_Sport;
    public static Vector<String> URL_Addrs_Politics;

    public static void main(String[] args){

        String[] Key_Arr = new String[4];

        Key_Arr[0]="Chelsea";
        Key_Arr[1]="ManU";
        Key_Arr[2]="Everton";
        Key_Arr[3]="Liverpool";

        Rss_Info tmp_rss_info = new
        Rss_Info(Key_Arr, "soccer");

        Vector<String> tmp_vector =
        tmp_rss_info.Get_Rss_Vector();

        System.out.println("");

        for(int j=0;j<tmp_vector.size();j++){
            System.out.println(tmp_vector.get(j));
        }

    }

}
```

Console (Result) :

```
Moyes close to £17m Everton deal
Arsenal & Chelsea record big wins
```

Example No. 2 :

```

import java.util.Iterator;
import java.util.Vector;

public class Main {

    public static Vector<String> URL_Addrs_Sport;
    public static Vector<String> URL_Addrs_Politics;

    public static void main(String[] args){

        Vector<String> key_words=new Vector<String>();
        key_words.add("Biden");
        key_words.add("McCain");
        key_words.add("Russia");
        key_words.add("Olmert");
        key_words.add("Israel");
        key_words.add("Middle");
        key_words.add("Obama");

        Rss_Info tmp_rss_info = new
        Rss_Info("http://rss.news.yahoo.com/rss/electio
        ns", key_words);
        Iterator<String> tmp_Iterator =
        tmp_rss_info.Get_Rss_Iterator();

        System.out.println("");

        while(tmp_Iterator.hasNext()){
            System.out.println(tmp_Iterator.next());
        }
    }
}

```

Console (Result) :

Obama to meet economic advisers to offer new plans
(AP)

McCain says he would fire SEC chairman
(AP)

Obama Gains in Polls as Financial Crisis Shifts Focus
(Bloomberg)

McCain Effort to Tie Obama to Fannie Mae Woes Sets off Political
Brawl
(CQPolitics.com)

McCain salutes sacrifice of fallen soldier
(AP)

About ROME

We used code from the core ROME library. A few words about it:

ROME is an set of open source Java tools for parsing, generating and publishing RSS and Atom feeds. The **core ROME library** depends only on the JDOM XML parser and supports parsing, generating and converting all of the popular RSS and Atom formats including *RSS 0.90, RSS 0.91 Netscape, RSS 0.91 Userland, RSS 0.92, RSS 0.93, RSS 0.94, RSS 1.0, RSS 2.0, Atom 0.3, and Atom 1.0*. You can parse to an RSS object model, an Atom object model or an abstract SyndFeed model that can model either family of formats.

Bibliography

- # Michael Orlov's site about Saya - <http://www.cs.bgu.ac.il/~orlovm/teaching/saya/> -
- # The core ROME library - <https://rome.dev.java.net/>
- # News@BGU, Winter 2007 - <http://cmsprod.bgu.ac.il/NR/rdonlyres/F3311087-8BB0-4C3D-83D0-4A544AAA07C5/0/newsatBGUwinter2007.pdf> -