

# RSSAggregator.java

```

1 import components.simplereader.SimpleReader;
2
3 /**
4  * Program to convert a set of XML RSS (version 2.0) feed from a given URL into
5  * the corresponding HTML output file. There will be an index page with links to
6  * all the single RSS feeds.
7  *
8  * @author Robert Frenken
9  */
10 public final class RSSAggregator {
11
12     /**
13      * Private constructor so this utility class cannot be instantiated.
14      */
15     private RSSAggregator() {
16
17     }
18
19     /**
20      * Outputs the "opening" tags in the generated HTML file. These are the
21      * expected elements generated by this method:
22      *
23      * <html> <head> <title>the channel tag title as the page title</title>
24      * </head> <body>
25      * <h1>the page title inside a link to the <channel> link</h1>
26      * <p>
27      * the channel description
28      * </p>
29      * <table border="1">
30      * <tr>
31      * <th>Date</th>
32      * <th>Source</th>
33      * <th>News</th>
34      * </tr>
35      *
36      * @param channel
37      *         the channel element XMLTree
38      * @param out
39      *         the output stream
40      * @updates out.content
41      * @requires [the root of channel is a <channel> tag] and out.is_open
42      * @ensures out.content = #out.content * [the HTML "opening" tags]
43      */
44     private static void outputHeader(XMLTree channel, SimpleWriter out) {
45         assert channel != null : "Violation of: channel is not null";
46         assert out != null : "Violation of: out is not null";
47         assert channel.isTag() && channel.label().equals("channel") : ""
48             + "Violation of: the label root of channel is a <channel> tag";
49         assert out.isOpen() : "Violation of: out.is_open";
50
51         // know from earlier that this is a valid rss, so all these elements are
52         // a part of the rss
53
54         int titleIndex = getChildElement(channel, "title");
55         int linkIndex = getChildElement(channel, "link");
56         String title = "";
57         String link = "";
58     }
59 }

```

# RSSAggregator.java

```

63     if (channel.child(titleIndex).numberOfChildren() > 0) {
64         title = channel.child(titleIndex).child(0).label();
65     }
66     if (channel.child(linkIndex).numberOfChildren() > 0) {
67         link = channel.child(linkIndex).child(0).label();
68     }
69
70     out.print("<html xmlns=\"http://www.w3.org/1999/xhtml\">\r\n"
71             + "<head>\r\n"
72             + "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\"
73             + "<title>" + title + "</title>\r\n" + "\r\n" + "</head>\r\n"
74             + "\r\n" + "\r\n" + "<body>");
75
76     out.print("<h1>\r\n" + "<a href= \"" + link + "\"> " + title + "</a>\r\n"
77             + "</h1>\r\n");
78
79     int descriptionIndex = getChildElement(channel, "description");
80     if (channel.child(descriptionIndex).numberOfChildren() > 0) {
81         String description = channel.child(descriptionIndex).child(0)
82             .label();
83         out.print("<p>" + description + "</p>\r\n");
84     } else {
85         out.print("<p>" + "No description available" + "</p>\r\n");
86     }
87
88     out.print("<table border = \"1\">\r\n" + "<tr>\r\n"
89             + "<th>Date</th>\r\n" + "<th>Source</th>\r\n"
90             + "<th>News</th>\r\n" + "</tr>\r\n");
91
92 }
93
94 /**
95  * Outputs the "closing" tags in the generated HTML file. These are the
96  * expected elements generated by this method:
97  *
98  * </table>
99  * </body> </html>
100  *
101  * @param out
102  *         the output stream
103  * @updates out.contents
104  * @requires out.is_open
105  * @ensures out.content = #out.content * [the HTML "closing" tags]
106  */
107 private static void outputFooter(SimpleWriter out) {
108     assert out != null : "Violation of: out is not null";
109     assert out.isOpen() : "Violation of: out.is_open";
110
111     out.print("</table>\r\n" + "</body>\r\n" + "</html>");
112 }
113
114 /**
115  * Finds the first occurrence of the given tag among the children of the
116  * given {@code XMLTree} and return its index; returns -1 if not found.
117  *
118  * @param xml

```

# RSSAggregator.java

```

119     *           the {@code XMLTree} to search
120     * @param tag
121     *           the tag to look for
122     * @return the index of the first child of type tag of the {@code XMLTree}
123     *         or -1 if not found
124     * @requires [the label of the root of xml is a tag]
125     * @ensures <pre>
126     * getChildElement =
127     * [the index of the first child of type tag of the {@code XMLTree} or
128     * -1 if not found]
129     * </pre>
130     */
131     private static int getChildElement(XMLTree xml, String tag) {
132         assert xml != null : "Violation of: xml is not null";
133         assert tag != null : "Violation of: tag is not null";
134         assert xml.isTag() : "Violation of: the label root of xml is a tag";
135
136         int counter = 0;
137         int elementIndex = -1;
138         while (counter < xml.numberOfChildren() && elementIndex != counter) {
139             if (xml.child(counter).isTag()
140                 && tag.equals(xml.child(counter).label())) {
141                 elementIndex = counter;
142             }
143             counter++;
144         }
145
146         return elementIndex;
147     }
148
149     /**
150     * Processes one news item and outputs one table row. The row contains three
151     * elements: the publication date, the source, and the title (or
152     * description) of the item.
153     *
154     * @param item
155     *           the news item
156     * @param out
157     *           the output stream
158     * @updates out.content
159     * @requires [the label of the root of item is an <item> tag] and
160     *           out.is_open
161     * @ensures <pre>
162     * out.content = #out.content *
163     * [an HTML table row with publication date, source, and title of news item]
164     * </pre>
165     */
166     private static void processItem(XMLTree item, SimpleWriter out) {
167         assert item != null : "Violation of: item is not null";
168         assert out != null : "Violation of: out is not null";
169         assert item.isTag() && item.label().equals("item") : ""
170             + "Violation of: the label root of item is an <item> tag";
171         assert out.isOpen() : "Violation of: out.is_open";
172
173         out.print("<tr>\r\n");
174
175         int pubDateIndex = getChildElement(item, "pubDate");

```

# RSSAggregator.java

```

176 // pubdate column, either there or not
177 if (pubDateIndex != -1) {
178     String pubDate = item.child(pubDateIndex).child(0).toString();
179     out.print("<td>" + pubDate + "</td>\r");
180 } else {
181     out.print("<td>No date available</td>");
182 }
183
184 int sourceIndex = getChildElement(item, "source");
185 // source column, either there with url, or not
186 if (sourceIndex != -1) {
187     if (item.child(sourceIndex).numberOfChildren() > 0) {
188         String source = item.child(sourceIndex).child(0).label();
189         String sourceURL = item.child(sourceIndex)
190             .attributeValue("url");
191         out.println("<td>\r\n" + "<a href= \"" + sourceURL + "\">"
192             + source + "</a>\r\n" + "</td>\r\n");
193     }
194 } else {
195     out.print("<td>No source available</td>");
196 }
197
198 // title column, with either a title with or without a link, or a description
199 // with or without a link, or "no title available" with or without a link
200
201 int titleIndex = getChildElement(item, "title");
202 int descriptionIndex = getChildElement(item, "description");
203 int linkIndex = getChildElement(item, "link");
204 if (titleIndex != -1 && item.child(titleIndex).numberOfChildren() > 0) {
205     String title = item.child(titleIndex).child(0).label();
206
207     // checks if a link tag exists in the particular item
208
209     if (linkIndex != -1) {
210         String link = item.child(linkIndex).child(0).label();
211         out.println("<td>\r\n" + "<a href= \"" + link + "\">" + title
212             + "</a>\r\n" + "</td>\r\n");
213     } else {
214         out.println("<td>\r\n" + title + "</td>\r\n");
215     }
216 }
217
218 } else if (descriptionIndex != -1) {
219     if (item.child(descriptionIndex).numberOfChildren() > 0) {
220         String description = item.child(descriptionIndex).child(0)
221             .label();
222         // checks if a link tag exists in the particular item
223         if (getChildElement(item, "link") != 1) {
224             String link = item.child(linkIndex).child(0).label();
225             out.println("<td>\r\n" + "<a href= \"" + link + "\">"
226                 + description + "</a>\r\n" + "</td>\r\n");
227         } else {
228             out.println("<td>\r\n" + description + "</td>\r\n");
229         }
230     }
231 }
232

```

# RSSAggregator.java

```

233     } else {
234         if (linkIndex != -1) {
235             String link = item.child(linkIndex).child(0).label();
236             out.println("<td>\r\n" + "<a href= \"" + link + "\">"
237                 + "No title available" + "</a>\r\n" + "</td>\r\n");
238         } else {
239             out.print("<td>No title available</td>");
240
241         }
242     }
243     out.print("</tr>\r\n");
244 }
245
246 /**
247  * Processes one XML RSS (version 2.0) feed from a given URL converting it
248  * into the corresponding HTML output file.
249  *
250  * @param url
251  *     the URL of the RSS feed
252  * @param file
253  *     the name of the HTML output file
254  * @param out
255  *     the output stream to report progress or errors
256  * @updates out.content
257  * @requires out.is_open
258  * @ensures <pre>
259  * [reads RSS feed from url, saves HTML document with table of news items
260  *  to file, appends to out.content any needed messages]
261  * </pre>
262  */
263 private static void processFeed(String url, String file, SimpleWriter out) {
264     XMLTree xml = new XMLTree1(url);
265     SimpleWriter outHTML = new SimpleWriter1L(file);
266     outputHeader(xml.child(0), outHTML);
267     // create channel xml similar to lab
268     XMLTree channel = xml.child(0);
269
270     // go through all child tags of channel, looking for item tags
271     for (int i = 0; i < channel.numberOfChildren(); i++) {
272
273         if (channel.child(i).isTag()
274             && channel.child(i).label().equals("item")) {
275
276             processItem(channel.child(i), outHTML);
277         }
278     }
279     outputFooter(outHTML);
280
281     outHTML.close();
282 }
283
284 /**
285  * Main method.
286  *
287  * @param args
288  *     the command line arguments; unused here
289  */

```

# RSSAggregator.java

```

290 public static void main(String[] args) {
291     SimpleReader in = new SimpleReader1L();
292     SimpleWriter out = new SimpleWriter1L();
293
294     out.print(
295         "Enter the URL XML containing a list of RSS 2.0 news feeds: ");
296     String urlFeed = in.nextLine();
297
298     XMLTree xmlFeed = new XMLTree1(urlFeed);
299
300     // makes sure the top tag is correct
301     while (!xmlFeed.hasAttribute("title")
302         || !xmlFeed.label().equals("feeds")) {
303         out.println("This is not a valid xml with a list of RSS feeds");
304         out.print(
305             "Enter the URL XML containing a list of RSS 2.0 news feeds: ");
306         urlFeed = in.nextLine();
307         xmlFeed = new XMLTree1(urlFeed);
308     }
309
310     // ask user to output to an html file
311     out.print("Enter an html file to publish the list of RSS feeds: ");
312     String htmlFile = in.nextLine();
313     SimpleWriter outHTML = new SimpleWriter1L(htmlFile);
314
315     // create header for HTML page
316     outHTML.print("<html>\r\n" + "<head>\r\n" + "<title>\r\n"
317         + xmlFeed.attributeValue("title"));
318     outHTML.print("</title>\r\n" + "</head>\r\n");
319     outHTML.print("<body>\r\n" + "<h2>" + xmlFeed.attributeValue("title")
320         + "</h2>\r\n");
321     outHTML.print("<ul>\r\n");
322     // create body for HTML page
323     for (int i = 0; i < xmlFeed.numberOfChildren(); i++) {
324         outHTML.print("<li>\r\n");
325         outHTML.print("<a href = " + "\""
326             + xmlFeed.child(i).attributeValue("file") + "\""
327             + xmlFeed.child(i).attributeValue("name") + "</a>\r\n");
328         outHTML.print("</li>\r\n");
329     }
330     // create footer for HTML page
331     outHTML.print("</ul>\r\n" + "</body>\r\n" + "</html>");
332
333     // iterate through the feeds to determine if each feed is a valid RSS feed
334     for (int i = 0; i < xmlFeed.numberOfChildren(); i++) {
335         String url = xmlFeed.child(i).attributeValue("url");
336         // Puts url into xml tree object
337         XMLTree xml = new XMLTree1(url);
338
339         /*
340          * Makes sure XML is an rss Checks if it has an attribute version,
341          * that it's value is "2.0", and if the label name is "rss".
342          */
343         while (!xml.hasAttribute("version")
344             || !(xml.attributeValue("version").equals("2.0"))
345             || !(xml.label().equals("rss"))) {
346

```

RSSAggregator.java

```
347         out.println("This is a valid XML, but not an RSS.");
348         out.print("Enter the URL of an RSS 2.0 news feed: ");
349         url = in.nextLine();
350         xml = new XMLTree1(url);
351     }
352 }
353
354 for (int i = 0; i < xmlFeed.numberOfChildren(); i++) {
355     processFeed(xmlFeed.child(i).attributeValue("url"),
356                 xmlFeed.child(i).attributeValue("file"), out);
357     out.println(xmlFeed.child(i).attributeValue("file"));
358 }
359
360 in.close();
361
362 out.close();
363 outHTML.close();
364 }
365
366 }
```