Java 2 Micro Edition
XML

F. Ricci

2010/2011

J2Me XML overview

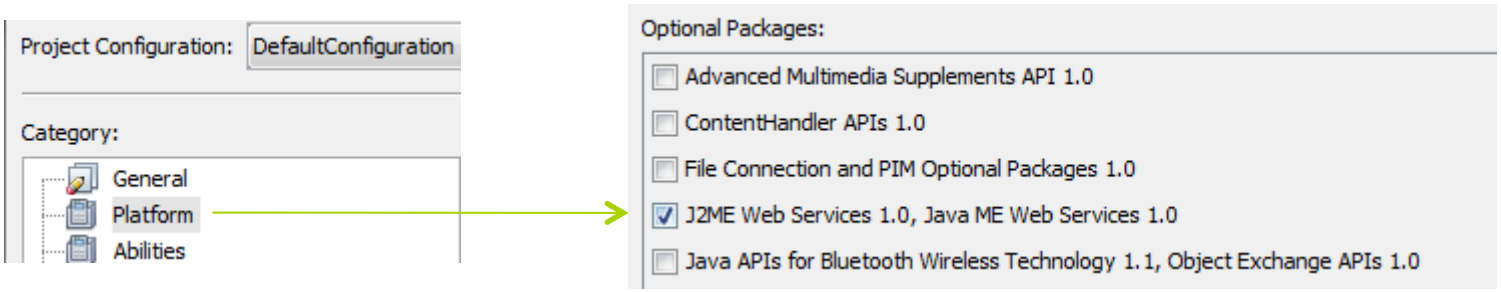
- **XML, REST**

- **Parsing XML**:
  - **JSR 172:** defines an XML parsing (JAXP) API and a JAX-RPC API (mostly used for Web Services, and currently not available on many devices)
  - **kXML2**: Parsing XML without JSR 172

- **REST:**
  - Simple URLs
  - XML responses
  - Good documentation
  - RESTful Flickr Client

JSR 172

- JSR 172: Java ME Web Services
  - Access to remote SOAP / XML based web services
  - **Parsing XML data**
- JSR 172 is a subset of JSR 63
  - JSR 63: Java API for XML Processing
- Requires only CLDC 1.0
- Basic XML Parsing capabilities



API Features and Restrictions

- SAX 2.0 subset
- XML Namespaces
- UTF-8 and UTF-16 support
- No DOM handling
- No XSLT Transformations

Packages

- **javax.xml.parsers**
  - To obtain and reference platform's parser
    - SAXParser, SAXParserFactory
- **org.xml.sax**
  - Subset of SAX 2.0 API Classes and interfaces.
    - Attributes, Locator, InputSource, …
- **org.xml.sax.helpers**
  - Class to extend to receive parse events
    - DefaultHandler

Parsing XML – using JSR 172

- SAX parser (**S**imple **A**PI for **X**ML) It's a **push parser** (runs through an entire document, and generate a series of events as it goes)
- Get the parser:

```
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser = spf.newSAXParser();
```

- Creating a Handler and give it a document:

```
// InputStream in = …
// DefaultHandler dh = …
spf.parse(in, dh);
```

- Create your own handler by inheriting the DefaultHandler and override methods you want:

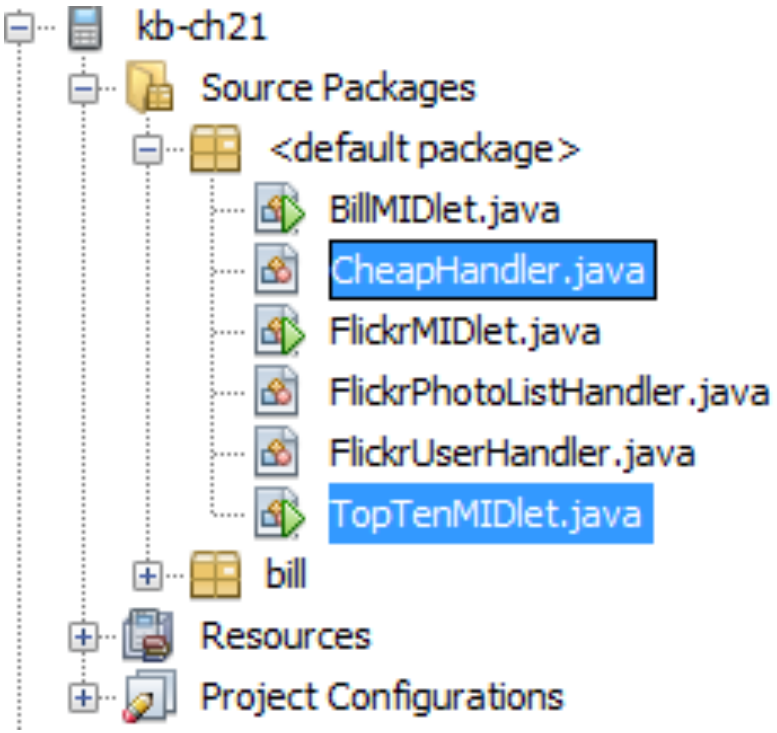
```
startDocument(), endDocument(),
startElement(), endElement(),
characters()
```

Parsing RSS

- RSS is a simple XML format used for summaries of blogs, news Web sites, and other content.
- We'll see a MIDlet that retrieves an RSS feed that contains the top ten songs sold in the iTunes music store.
- You could easily modify this MIDlet to retrieve other types of RSS feeds.

Ex: Parsing RSS with JSR 172

- Download the file **kb-ch21-nb.zip**
- Unzip and open the project **kb-ch21** from NetBeans

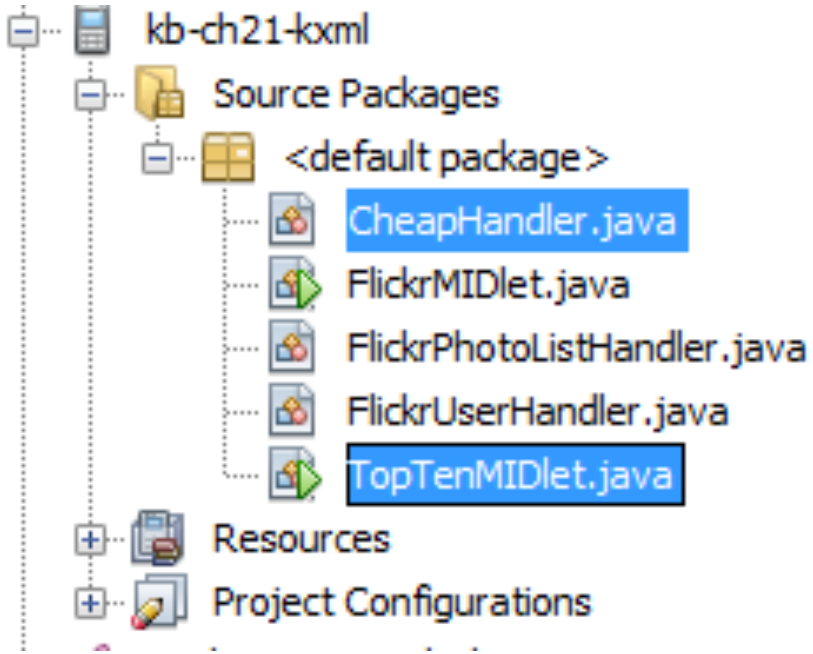


kXML2: Parsing XML without JSR 172

- Not many current devices have support for either of the JSR 172 APIs.
- However, if you just want to parse XML in a MIDlet, you can embed a small parser in your application. kXML 2 is an excellent choice:
  - http://kxml.sourceforge.net/
- **kXML** is a small XML pull parser, specially designed for constrained environments such as MIDP devices.
- **Pull based XML** parsing combines some of the advantages of SAX and DOM.
- The size of the XML parser adds directly to your MIDlet suite JAR file size, but the kXML 2 JAR file is only 43 KB.
- You can run kXML on any MIDP 2.0 device (kb-ch21-kxml source code)
- Using the kXML parser, `org.kxml2.io.KXmlParser`, is easy!

Ex: Parsing RSS with kXML2

- Download the file **kb-ch21-nb.zip**
- Unzip and open the project **kb-ch21-kxml** from NetBeans



REST (**Re**presentational **S**tate **T**ransfer)

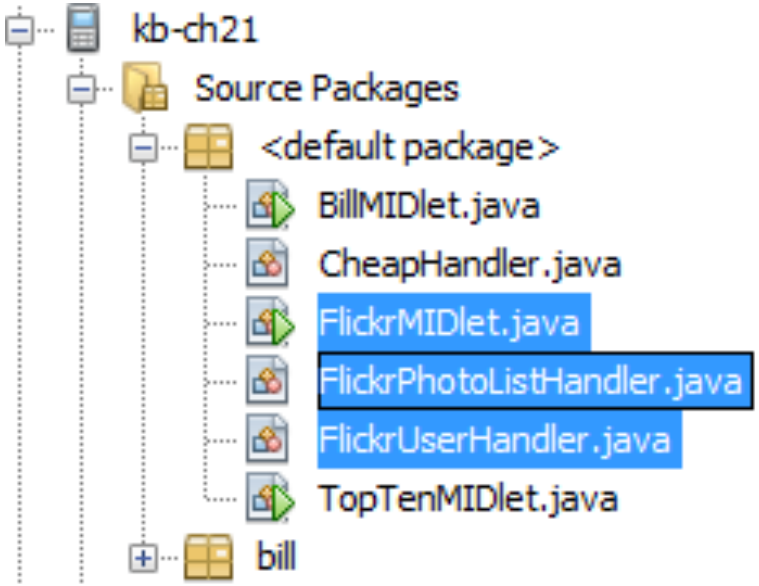
- Simple URLs
  - http://api.flickr.com/services/rest/? …
- XML responses

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat = "ok">
  <photos page = "1" …>
    <photo id = "09098098" … />
    <photo id = "09098098" … />
    …
  </photos>
</rsp>
```

- Good documentation
  - http://www.flickr.com/services/api/

Mobile client for Flickr

- Download the file **kb-ch21-nb.zip**
- Unzip and open the project **kb-ch21** from NetBeans



Mobile client for Flickr

- API for flickr:
  - http://www.flickr.com/services/api/
- Flickr API MIDlet needs a Flickr API key that you can obtain from http://www.flickr.com/services/api/misc.api_keys.html
  - Set this as value of the "flickr-apikey" attribute in the JAD of your project
- The MIDlet makes three calls into the Flickr API
  1. **flickr.people.findByUsername:** returns a flickr user id
  2. **flickr.people.getPublicPhotos:** returns a list of information about the public photos for a given flickr user id
  3. **Retrieve photos:** using the URL scheme described in the flickr API documentation

FlickrUserHandler extends CheapHandler

```
public class FlickrUserHandler extends CheapHandler {
  private String mNSID;
  public String getNSID() { return mNSID; }
  public void processStart(String tree, Hashtable a) {
    if (tree.equals("rsp|user")) {
      Enumeration keys = a.keys();
      while (keys.hasMoreElements()) {
        String name = (String)keys.nextElement();
        String value = (String)a.get(name);
        if (name.equals("nsid"))
          mNSID = value;
      }
    }
  }
}
```

flickr.people.getPublicPhotos

```
private Vector lookupPictures(String nsid)
      throws IOException, ParserConfigurationException, SAXException {

   Vector urls = null;
    String cs = kURL + "method=flickr.people.getPublicPhotos&" +
        "user_id=" + nsid + "&" + "api_key=" + mAPIKey;
    HttpConnection hc = (HttpConnection)Connector.open(cs);
    try {
      InputStream in = hc.openInputStream();
      try {
        FlickrPhotoListHandler fplh = new FlickrPhotoListHandler();
        parseToHandler(in, fplh);
        urls = fplh.getPhotoURLs();
      }
      finally { in.close(); }
    }
    finally { hc.close(); }
    return urls;
  }
```

FlickrPhotoListHandler extends CheapHandler

```
public void processStart(String tree, Hashtable a) {
    if (tree.equals("rsp|photos|photo")) {
      String id = getAttribute(a, "id");
      String secret = getAttribute(a, "secret");
      String server = getAttribute(a, "server");
      String farm = getAttribute(a, "farm");

      if (mPhotoURLs == null) mPhotoURLs = new Vector();

      String url = "http://farm" + farm +
          ".static.flickr.com/" +
          server + "/" + id + "_" + secret + "_t.jpg";
      mPhotoURLs.addElement(url);
    }
  }
```

Load a single image

```
private byte[] lookupPicture(String cs) throws
 IOException {
  byte[] raw = null;
  HttpConnection hc = HttpConnection)Connector.open(cs);
  try {
    InputStream in = hc.openInputStream();
    try { raw = readAll(in); }
    finally { in.close(); }
  }
  finally { hc.close(); }
  return raw;
}
```

References

Kicking Butt with MIDP and MSA:

- Available on Safari (Ch. 21):
  Kicking Butt with MIDP and MSA: Creating Great Mobile Applications

Book Sources:

- http://kickbutt.jonathanknudsen.com/download.html
- http://kickbutt.jonathanknudsen.com/src/kb-ch21-nb.zip

kXML2 parser:

- http://kxml.sourceforge.net
- http://www-128.ibm.com/developerworks/edu/wi-dw-wi-kxml-i.html (nice introduction)

Devices JSR support:

- http://www.dpsoftware.org/filter.php

SAX parser definition:

- http://en.wikipedia.org/wiki/Simple_API_for_XML