EDA095
HTML

Pierre Nugues

Lund University
http://cs.lth.se/pierre_nugues/

April 13, 2016

Covers: Chapter 8, pages 248-266, *Java Network Programming*, 3rd ed., Elliotte Rusty Harold

HTML

HTML is the page description language used by the web.

Derives from SGML

Very messy:

- Sloppy syntax
- Many versions, many flavors

There are tons of sites where to learn it

We will review basic Java tools to analyze HTML pages to display and parse HTML

Displaying HTML Tags

Most Swing components understand HTML tags as JLabel:

```
public class SimpleGUI {
  public static void main(String[] args) {
    JFrame frame = new JFrame();
    JLabel label = new JLabel("<html>
      <p>Hello! This is a multiline label with <b>bold</b>
      and <i>italic</i> text<hr></p></html>");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(label);
    frame.setSize(300, 300);
    frame.setVisible(true);
  }
}// SimpleGUI.java
```

Displaying HTML Pages: JEditorPane

```
public static void main(String[] args) {
  JFrame f = new JFrame("LTH");
  f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE)
  JEditorPane jep = new JEditorPane();
  jep.setEditable(false);
  try {
    jep.setPage("http://cs.lth.se");
  } catch (IOException ex) {
    jep.setContentType("text/html");
    jep.setText("<html>Could not load http://? </html>");
  }
  JScrollPane scrollPane = new JScrollPane(jep);
  f.setContentPane(scrollPane);
  f.setSize(512, 342);
  f.setVisible(true);
} //LTHHomePage.java
```

Parsing HTML Pages

Java has a class to implement a primitive HTML parser:

`HTMLEditorKit.Parser` in the `javax.swing.html.text` package

It is an inner class of `HTMLEditorKit`

`Parser` is an abstract class that is instantiated using an obscure and complex sequence of operations. See textbook, 3rd edition, page 248.

The code is idiosyncratic: hardwired and impossible to explain

We get the parser using the `getParser()` method that needs to be made public

There are alternative HTML parsers in Java such as jsoup.org.

Making getParser() Public

Java Network Programming, 3rd ed., chapter 8, page 250

```
import javax.swing.text.html.HTMLEditorKit;

public class ParserGetter extends HTMLEditorKit {
    // purely to make this method public
    public HTMLEditorKit.Parser getParser(){
        return super.getParser();
    }
}
```

We can then parse a page using the HTMLEditorKit.Parser method:

```
parse(Reader r, HTMLEditorKit.ParserCallback cb,
  boolean ignoreCharSet)
```

cb uses the parsing results and reacts to HTML tags, text, or comments.

Parsing HTML

You create an event-driven HTML analyzer using the class HTMLEditorKit.ParserCallback where you overload the undocumented methods you need:

```
void handleText(char[] data, int pos) //TagStripper.java
void handleStartTag(HTML.Tag t, MutableAttributeSet a,
  int pos)
void handleEndTag(HTML.Tag t, int pos)
void handleSimpleTag(HTML.Tag t, MutableAttributeSet a,
  int pos)
void handleComment(char[] data, int pos)
void handleEndOfLineString(String eol)
void handleError(String errorMsg, int pos)
```

Printing the Text of a Web Page

TagStripper.java by Elliotte Rusty Harold, *Java Network Programming*, 3rd ed., page 251.

```
public class TagStripper extends HTMLEditorKit.ParserCallback
{
  private Writer out;
  public TagStripper(Writer out) {
    this.out = out;
  }
  public void handleText(char[] text, int position) {
    try {
      out.write(text);
      out.flush();
    } catch (IOException ex) {
      System.err.println(ex);
    }
  }
}
```

Printing the Text of a Web Page

```
  public static void main(String[] args) {
    ParserGetter kit = new ParserGetter();
    HTMLEditorKit.Parser parser = kit.getParser();
    HTMLEditorKit.ParserCallback callback =
      new TagStripper(new OutputStreamWriter(System.out));
    try {
      URL url = new URL("http://cs.lth.se/EDA095/");
      InputStream in =
        new BufferedInputStream(url.openStream());
      InputStreamReader r = new InputStreamReader(in);
      parser.parse(r, callback, true);
    } catch (IOException ex) {
      ex.printStackTrace();
      System.err.println(ex);
    }
  }
}
```

Parsing Tags

HTML tags:

```
static HTML.Tag.A
static HTML.Tag.ADDRESS
static HTML.Tag.APPLET
static HTML.Tag.AREA
static HTML.Tag.B
static HTML.Tag.H1
```

etc.

Formatting instructions:

```
boolean breaksFlow()
boolean isBlock()
boolean isPreformatted()
```

Outlining the Titles of a Web Page

The HTML source:

```
<h1>Text header 1</h1>
<h2>Text header 21</h2>
<h3>Text header 3</h3>
<h2>Text header 21</h2>
```

will be displayed as:

```
Text header 1
   Text header 21
      Text header 3
   Text header 21
```

Outlining the Titles of a Web Page

```
public class OutlinerSimple extends HTMLEditorKit.
    ParserCallback {
  private Writer out;
  private boolean inHeader = false;
  private int level = 0;
  public void handleStartTag(HTML.Tag tag,
      MutableAttributeSet attributes, int position) {
    if (tag == HTML.Tag.H1 || tag == HTML.Tag.H2 ||
        tag == HTML.Tag.H3 || tag == HTML.Tag.H4) {
      inHeader = true;
      if (tag == HTML.Tag.H1) level = 1;
      if (tag == HTML.Tag.H2) level = 2;
      if (tag == HTML.Tag.H3) level = 3;
      if (tag == HTML.Tag.H4) level = 4;
    }
  }
```

Outlining the Titles of a Web Page

```
public void handleEndTag(HTML.Tag tag, int position) {
  if (tag == HTML.Tag.H1 || tag == HTML.Tag.H2 ||
      tag == HTML.Tag.H3  || tag == HTML.Tag.H4) {
    inHeader = false;
    level = 0;
  }
}
public void handleText(char[] text, int position) {
  if (inHeader == true) {
    if (level == 2) System.out.print("\t");
    if (level == 3) System.out.print("\t\t");
    if (level == 4) System.out.print("\t\t\t");
    System.out.println(text);
  }
}
```

Outlining the Titles of a Web Page

```
  public static void main(String[] args) {
    ParserGetter kit = new ParserGetter();
    HTMLEditorKit.Parser parser = kit.getParser();
    HTMLEditorKit.ParserCallback callback =
      new OutlinerSimple();
    try {
      URL url = new URL("http://cs.lth.se/EDA095/");
      InputStream in =
        new BufferedInputStream(url.openStream());
      InputStreamReader r = new InputStreamReader(in);
      parser.parse(r, callback, true);
    } catch (IOException ex) {
      ex.printStackTrace();
      System.err.println(ex);
    }
  }
} // OutlinerSimple.java
```

Printing the Links of a Web Page

HTML links have the form:

```
<a href="blackhole.html">click me!</a>
```

where

- `A` is called the tag or element in XML
- `HREF` is an attribute: `HTML.Attribute.HREF`

HTML frames have the form: `<frame src="myframe.html"/>`
HTML images have the form: `<img src="mypics.jpg"/>`
To build absolute URLs from relative URLs, we extract the `BASE` tag from the current web page and its `HREF` attribute, if it exists, as in

```
<base href="http://cs.lth.se/" />
```

otherwise we use the address of the page.

Printing the Links of a Web Page

We extract them using the code:

```
public void handleStartTag(HTML.Tag tag,
    MutableAttributeSet a, int position) {
  if(tag == HTML.Tag.A) {
    String href =
      (String) a.getAttribute(HTML.Attribute.HREF);
    System.out.println("Link: " + href);
  }
}
```

Printing the Links of a Web Page

```
public void handleSimpleTag(HTML.Tag tag,
    MutableAttributeSet a, int pos) {
  if (tag == HTML.Tag.BASE) {
    String href =
      (String) a.getAttribute(HTML.Attribute.HREF);
    baseURL = href;
    System.out.println("Base URL: " + href);
  }
  if(tag == HTML.Tag.IMG) {
    String href = (String) a.getAttribute(HTML.Attribute.SRC);
    System.out.println("Image: " + href);
  }
  if(tag == HTML.Tag.FRAME) {
    String href = (String) a.getAttribute(HTML.Attribute.SRC);
    System.out.println("Frame: " + href);
  }
} // LinkGetter.java
```

Absolute Links

Some links are absolute, while others are relative.
We can use this piece of code to create absolute links in `handleStartTag()`:

```
try {
  if (!new URI(href).isAbsolute()) {
    System.out.println(
     "\tAbsolute link: " + new URL(new URL(baseURL), href));
  }
} catch (Exception e) { }
```

or just

```
new URL(new URL(baseURL), href)
```

jsoup

jsoup.org is a very popular HTML parser, probably easier to use
It requires an external library

```
public class First {
    void parse() {
        String html = "<html><head><title>First parse</title><
                + "<body><p>Parsed HTML into a doc.</p></body>
        Document doc = Jsoup.parse(html);
        System.out.println(doc);
    }

    public static void main(String[] args) {
        First first = new First();
        first.parse();
    }
}
```

Extracting links with jsoup

```
void parse() throws IOException {
    URL url = new URL("http://cs.lth.se/eda095/");
    InputStream is = url.openStream();
    Document doc = Jsoup.parse(is, "UTF-8", "http://cs.lth.se/
    Elements base = doc.getElementsByTag("base");
    System.out.println("Base : " + base);
    Elements links = doc.getElementsByTag("a");
    for (Element link : links) {
        String linkHref = link.attr("href");
        String linkAbsHref = link.attr("abs:href");
        String linkText = link.text();
        System.out.println("href: " + linkHref + "abshref: "
            + linkAbsHref + " text: " + linkText);
    }
    is.close();
}
```

Extracting Text with jsoup

```
public String html2text(String html) {
    return Jsoup.parse(html).text();
}
```

Extracting Headers with jsoup

```
public void outline(String html) {
    Document doc = Jsoup.parse(html);
    Elements els = doc.select("h1, h2, h3, h4, h5, h6");
    for (Element el : els) {
        //Last char
        Integer last = new Integer((el.tagName()).
            substring(el.tagName().length() - 1));
        String prefix = "";
        for (int i = 0; i < last; i++) {
            prefix += '\t';
        }
        System.out.println(prefix + el.text());
    }
}
```