Table of Contents

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

Chapter 8. HTML in Swing

As anyone who has ever tried to write code to read HTML can tell you, it's a painful experience. The problem is that although there is an HTML specification, no web designer or browser vendor actually follows it. And the specification itself is extremely loose. Element names may be uppercase, lowercase, or mixed case. Attribute values may or may not be quoted. If they are quoted, either single or double quotes may be used. The < sign may be escaped as `<` or it may just be left raw in the file. The `<P>` tag may be used to begin or end a paragraph. Closing `</P>`, `</LI>`, and `</TD>` tags may or may not be used. Tags may or may not overlap. There are just too many different ways of doing the same thing to make parsing HTML an easy task. In fact, the difficulties encountered in parsing real-world HTML were one of the prime motivators for the invention of the much stricter XML, in which what is and is not allowed is precisely specified and all browsers are strictly prohibited from accepting documents that don't measure up to the standard (as opposed to HTML, where most browsers try to fix up bad HTML, thereby leading to the proliferation of nonconformant HTML on the Web, which all browsers must then try to parse).

Fortunately, as of JFC 1.1.1 (included in Java 1.2.2 and later), Sun provides classes for basic HTML parsing and display that shield Java programmers from most of the tribulations of working with raw HTML. The `javax.swing.text.html.parser` package can read HTML documents in more or less their full, nonstandard atrocity, while the `javax.swing.text.html` package can render basic HTML in JFC-based applications.

8.1. HTML on Components

Most text-based Swing components, such as labels, buttons, menu items, tabbed panes, and tool tips, can have their text specified as HTML. The component will display it appropriately. If you want the label on a `JButton` to include bold, italic, and plain text, the simplest way is to write the label in HTML directly in the source code like this:

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

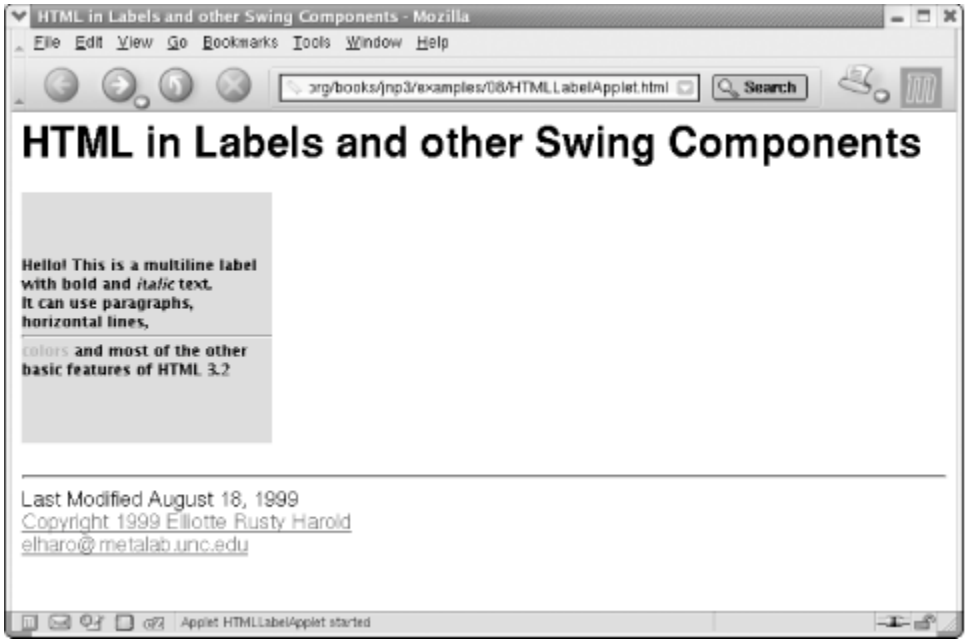
```
JButton jb = new JButton("<html><b><i>Hello World!</i></b></html>");
```

The same technique works for JFC-based labels, menu items, tabbed panes, and tool tips. Example 8-1 and Figure 8-1 show an applet with a multiline `JLabel` that uses HTML.

Example 8-1. Including HTML in a JLabel

```
import javax.swing.*;

public class HTMLLabelApplet extends JApplet {

  public void init( ) {

    JLabel theText = new JLabel(
     "<html>Hello! This is a multiline label with <b>bold</b> "
     + "and <i>italic</i> text. <P> "
     + "It can use paragraphs, horizontal lines, <hr> "
     + "<font color=red>colors</font> "
     + "and most of the other basic features of HTML 3.2</html>");

    this.getContentPane( ).add(theText);

  }

}
```

Figure 8-1. An HTML label



You can actually go pretty far with this. Almost all HTML tags are supported, at least partially, including `IMG` and the various table tags. The only completely unsupported HTML 3.2 tags

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

are <APPLET>, <PARAM>, <MAP>, <AREA>, <LINK>, <SCRIPT>, and <STYLE>. The various frame tags (technically not part of HTML 3.2, though widely used and implemented) are also unsupported. In addition, the various new tags introduced in HTML 4.0 such as BDO, BUTTON, LEGEND, and TFOOT, are unsupported.

Furthermore, there are some limitations on other common tags. First of all, relative URLs in attribute values are not resolved because there's no page for them to be relative to. This most commonly affects the SRC attribute of the IMG element. The simplest way around this is to store the images in the same JAR archive as the applet or application and load them from an absolute *jar* URL. Links will appear as blue underlined text as most users are accustomed to, but nothing happens when you click on one. Forms are rendered, but users can't type input or submit them. Some CSS Level 1 properties such as font-size are supported through the style attribute, but STYLE tags and external stylesheets are not. In brief, the HTML support is limited to static text and images. After all, we're only talking about labels, menu items, and other simple components.

8.2. JEditorPane

If you need a more interactive, complete implementation of HTML 3.2, you can use a javax.swing.JEditorPane. This class provides an even more complete HTML 3.2 renderer that can handle frames, forms, hyperlinks, and parts of CSS Level 1. The JEditorPane class also supports plain text and basic RTF, though the emphasis in this book will be on using it to display HTML.

JEditorPane supports HTML in a fairly intuitive way. You simply feed its constructor a URL or a large string containing HTML, then display it like any other component. There are four constructors in this class:

```
public JEditorPane( )
public JEditorPane(URL initialPage) throws IOException
public JEditorPane(String url)  throws IOException
public JEditorPane(String mimeType, String text)
```

The noargs constructor simply creates a JEditorPane with no initial data. You can change this later with the setPage() or setText( ) methods:

```
public void setPage(URL page) throws IOException
public void setPage(String url) throws IOException
public void setText(String html)
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

Example 8-2 shows how to use this constructor to display a web page. `JEditorPane` is placed inside a `JScrollPane` to add scrollbars; JFrame provides a home for the `JScrollPane`. Figure 8-2 shows this program displaying the O'Reilly home page.

Example 8-2. Using a JEditorPane to display a web page

```
import javax.swing.text.*;
import javax.swing.*;
import java.io.*;
import java.awt.*;

public class OReillyHomePage {

  public static void main(String[] args) {

     JEditorPane jep = new JEditorPane( );
     jep.setEditable(false);

     try {
       jep.setPage("http://www.oreilly.com");
     }
     catch (IOException ex) {
       jep.setContentType("text/html");
       jep.setText("<html>Could not load http://www.oreilly.com </html>");
     }

     JScrollPane scrollPane = new JScrollPane(jep);
     JFrame f = new JFrame("O'Reilly & Associates");
     f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
     f.setContentPane(scrollPane);
     f.setSize(512, 342);
     f.show( );

  }

}
```

Figure 8-2. The O'Reilly home page shown in a JEditorPane

Figure 8-2 shows how good (or bad) Swing really is at displaying HTML. On the whole, it correctly renders this page containing tables, images, links, colors, fonts, and more with almost no effort from the programmer. However, it has some trouble with table widths, and there are a number of artifacts I can't explain. Generally, the simpler and more basic the page, the better Swing renders it.

What is missing, though, is precisely what's not obvious from this static image: the activity. The links are blue and underlined, but clicking on one won't change the page that's displayed. JavaScript and applets will not run. Shockwave animations and QuickTime movies won't play. Password-protected web pages will be off-limits because there's no way to provide a username and password. You can add all this yourself, but it will require extra code to recognize the relevant parts of the HTML and behave accordingly. Different active content requires different levels of support. Supporting hypertext linking, for example, is fairly straightforward, as we'll explore later. Applets aren't that hard to add either, mostly requiring you to simply parse the HTML to find the tags and parameters and provide an instance of the `AppletContext` interface. Adding JavaScript is only a little harder, provided that someone has already written a JavaScript interpreter you can use. Fortunately, the Mozilla Project has written the Open Source Rhino (http://www.mozilla.org/rhino/) JavaScript interpreter, which you can use in your own work. Apple's QuickTime for Java (http://www.apple.com/quicktime/qtjava/) makes QuickTime support almost a no-brainer on Mac and Windows. (A Unix version is sorely lacking, though.) I'm not going to discuss all (or even most) of these in this chapter or this book. Nonetheless, they're available if you need them.

The second `JEditorPane` constructor accepts a `URL` object as an argument. It connects to the specified URL, downloads the page it finds, and attempts to display it. If this attempt fails, an `IOException` is thrown. For example:

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
JFrame f = new JFrame("O'Reilly & Associates");

try {
  URL u = new URL("http://www.oreilly.com");
  JEditorPane jep = new JEditorPane(u);
  jep.setEditable(false);
  JScrollPane scrollPane = new JScrollPane(jep);
  f.setContentPane(scrollPane);
}
catch (IOException ex) {
  f.getContentPane( ).add(
   new Label("Could not load http://www.oreilly.com"));
}

f.setSize(512, 342);
f.show( );
```

The third `JEditorPane` constructor is almost identical to the second except that it takes a `String` form of the URL rather than a URL object as an argument. One of the `IOExceptions` it can throw is a `MalformedURLException` if it doesn't recognize the protocol. Otherwise, its behavior is the same as the second constructor. For example:

```
try {
  JEditorPane jep = new JEditorPane("http://www.oreilly.com");
  jep.setEditable(false);
  JScrollPane scrollPane = new JScrollPane(jep);
  f.setContentPane(scrollPane);
}
catch (IOException ex) {
  f.getContentPane( ).add(
   new Label("Could not load http://www.oreilly.com"));
}
```

Neither of these constructors requires you to call `setText( )` or `setPage( )`, since that information is provided in the constructor. However, you can still call these methods to change the page or text that's displayed.

8.2.1. Constructing HTML User Interfaces on the Fly

The fourth `JEditorPane` constructor does not connect to a URL. Rather, it gets its data directly from the second argument. The MIME type of the data is determined by the first argument. For example:

```
JEditorPane jep = new JEditorPane("text/html",
 "<html><h1>Hello World!</h1> <h2>Goodbye World!</h2></html>");
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

This may be useful when you want to display HTML created programmatically or read from somewhere other than a URL. Example 8-3 shows a program that calculates the first 50 Fibonacci numbers and then displays them in an HTML ordered list. Figure 8-3 shows the output.

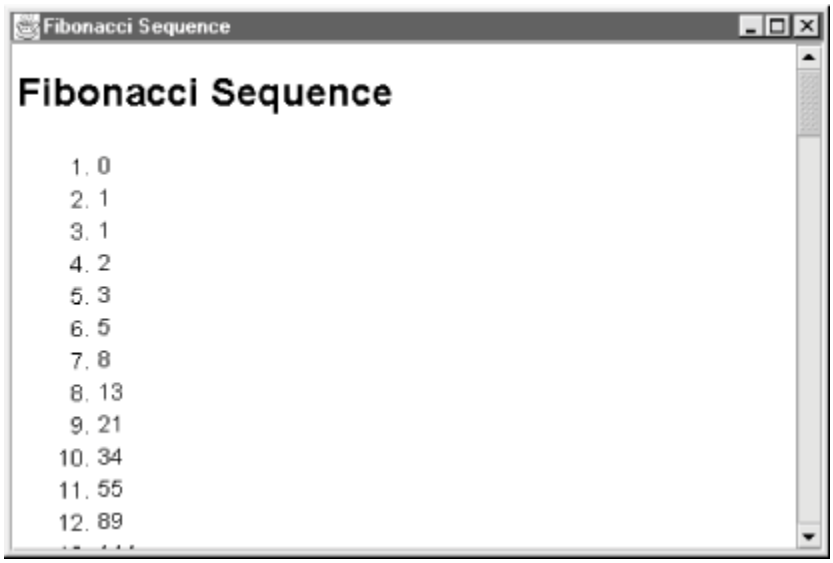
Example 8-3. Fibonacci sequence displayed in HTML

```
import javax.swing.text.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.awt.*;

public class Fibonacci {

 public static void main(String[] args) {

     StringBuffer result =
      new StringBuffer("<html><body><h1>Fibonacci Sequence</h1><ol>");

     long f1 = 0;
     long f2 = 1;

     for (int i = 0; i < 50; i++) {
       result.append("<li>");
       result.append(f1);
       long temp = f2;
       f2 = f1 + f2;
       f1 = temp;
     }

     result.append("</ol></body></html>");

     JEditorPane jep = new JEditorPane("text/html", result.toString( ));
     jep.setEditable(false);

     JScrollPane scrollPane = new JScrollPane(jep);
     JFrame f = new JFrame("Fibonacci Sequence");
     f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
     f.setContentPane(scrollPane);
     f.setSize(512, 342);
     EventQueue.invokeLater(new FrameShower(f));

  }

  // This inner class avoids a really obscure race condition.
  // See http://java.sun.com/developer/JDCTechTips/2003/tt1208.html#1
  private static class FrameShower implements Runnable {

    private final Frame frame;

    FrameShower(Frame frame) {
      this.frame = frame;
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
        }

        public void run( ) {
         frame.setVisible(true);
        }

      }

    }
```

Figure 8-3. The Fibonacci sequence displayed as HTML using a JEditorPane



The significance of this should be apparent. Your programs now have access to a very powerful styled text engine. That the format used on the backend is HTML is a nice fringe benefit. It means you can use a familiar, easy-to-write format to create a user interface that uses styled text. You don't have quite all the power of QuarkXPress here (nor should you, since HTML doesn't have it), but this is more than adequate for 99% of text display needs, whether those needs are simple program output, help files, database reports, or something more complex.

8.2.2. Handling Hyperlinks

When the user clicks on a link in a noneditable `JEditorPane`, the pane fires a `HyperlinkEvent`. As you might guess, this is responded to by any registered `HyperlinkListener` objects. This follows the same variation of the Observer design pattern used for AWT events and JavaBeans. The

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

`javax.swing.event.HyperlinkListener` interface defines a single method, `hyperlinkUpdate( )`:

```
public void hyperlinkUpdate(HyperlinkEvent evt)
```

Inside this method, you'll place the code that responds to the `HyperlinkEvent`. The `HyperlinkEvent` object passed to this method contains the URL of the event, which is returned by its `getURL( )` method:

```
public URL getURL( )
```

`HyperlinkEvent`s are fired not just when the user clicks the link, but also when the mouse enters or exits the link area. Thus, you'll want to check the type of the event before changing the page with the `getEventType()` method:

```
public HyperlinkEvent.EventType getEventType( )
```

This will return one of the three mnemonic constants `HyperlinkEvent.EventType.EXITED`, `HyperlinkEvent.EventTypeENTERED`, or `HyperlinkEvent.EventType.ACTIVATED`. These are not numbers but static instances of the `EventType` inner class in the `HyperlinkEvent` class. Using these instead of integer constants allows for more careful compile-time type checking.

Example 8-4 is an implementation of the `HyperLinkListener` interface that checks the event fired and, if it's an activated event, switches to the page in the link. A reference to the `JEditorPane` is stored in a private field in the class so that a callback to make the switch can be made.

Example 8-4. A basic HyperlinkListener class

```
import javax.swing.*;
import javax.swing.event.*;

public class LinkFollower implements HyperlinkListener {

  private JEditorPane pane;

  public LinkFollower(JEditorPane pane) {
    this.pane = pane;
  }

  public void hyperlinkUpdate(HyperlinkEvent evt) {
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
        if (evt.getEventType( ) == HyperlinkEvent.EventType.ACTIVATED) {
          try {
            pane.setPage(evt.getURL( ));
          }
          catch (Exception ex) {
            pane.setText("<html>Could not load " + evt.getURL( ) + "</html>");
          }
        }

      }

    }
```

Example 8-5 is a very simple web browser. It registers an instance of the `LinkFollower` class of Example 8-4 to handle any `HyperlinkEvent`s. It doesn't have a Back button, a Location bar, bookmarks, or any frills at all. But it does let you surf the Web by following links. The remaining aspects of the user interface you'd want in a real browser are mostly just exercises in GUI programming, so I'll omit them. But it really is amazing just how easy Swing makes it to write a web browser.

Example 8-5. SimpleWebBrowser

```
import javax.swing.text.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.awt.*;

public class SimpleWebBrowser {

  public static void main(String[] args) {

    // get the first URL
    String initialPage = "http://www.cafeaulait.org/";
    if (args.length > 0) initialPage = args[0];

    // set up the editor pane
    JEditorPane jep = new JEditorPane( );
    jep.setEditable(false);
    jep.addHyperlinkListener(new LinkFollower(jep));

    try {
      jep.setPage(initialPage);
    }
    catch (IOException ex) {
      System.err.println("Usage: java SimpleWebBrowser url");
      System.err.println(ex);
      System.exit(-1);
    }
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
        // set up the window
        JScrollPane scrollPane = new JScrollPane(jep);
        JFrame f = new JFrame("Simple Web Browser");
        f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        f.setContentPane(scrollPane);
        f.setSize(512, 342);
        EventQueue.invokeLater(new FrameShower(f));

      }

      // Helps avoid a really obscure deadlock condition.
      // See http://java.sun.com/developer/JDCTechTips/2003/tt1208.html#1
      private static class FrameShower implements Runnable {

        private final Frame frame;

        FrameShower(Frame frame) {
          this.frame = frame;
        }

        public void run( ) {
         frame.setVisible(true);
        }

      }

    }
```

The one thing this browser doesn't do is follow links to named anchors inside the body of a particular HTML page. There is a protected `scrollToReference( )` method in `JEditorPane` that can find the specified named anchor in the currently displayed HTML document and reposition the pane at that point; you can use this method to add this functionality if you so desire:

```
protected void scrollToReference(String reference)
```

8.2.3. Reading HTML Directly

The `JEditorPane` class mostly assumes that you're going to provide either a `URL` or the string form of a URL and let it handle all the details of retrieving the data from the network. However, it contains one method that allows you to read HTML directly from an input stream. That method is `read( )`:

```
public void read(InputStream in, Object document) throws IOException
```

This method may be useful if you need to read HTML from a chain of filter streams; for instance, unzipping it before you read it. It could also be used when you need to perform

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

some custom handshaking with the server, such as providing a username and password, rather than simply letting the default connection take place.

The first argument is the stream from which the HTML will be read. The second argument should be an instance of `javax.swing.text.html.HTMLDocument`. (You can use another type, but if you do, the `JEditorPane` will treat the stream as plain text rather than HTML.) Although you could use the `HTMLDocument( )` noargs constructor to create the `HTMLDocument` object, the document it creates is missing a lot of style details. You're better off letting a `javax.swing.text.html.HTMLEditorKit` create the document for you. You get an `HTMLEditorKit` by passing the MIME type you want to edit (text/html in this case) to the `JEditorPane` `getEditorKitForContentType( )` method like this:

```
EditorKit htmlKit = jep.getEditorKitForContentType("text/html");
```

Finally, before reading from the stream, you have to use the `JEditorPane`'s `setEditorKit( )` method to install a `javax.swing.text.html.HTMLEditorKit`. For example:

```
jep.setEditorKit(htmlKit);
```

This code fragment uses these techniques to load the web page at http://www.elharo.com. Here the stream comes from a URL anyway, so this is really more trouble than it's worth compared to the alternative. However, this approach would also allow you to read from a gzipped file, a file on the local drive, data written by another thread, a byte array, or anything else you can hook a stream to:

```
JEditorPane jep = new JEditorPane( );
jep.setEditable(false);
EditorKit htmlKit = jep.getEditorKitForContentType("text/html");
HTMLDocument doc = (HTMLDocument) htmlKit.createDefaultDocument( );
jep.setEditorKit(htmlKit);

try {
  URL u = new URL("http://www.elharo.com");
  InputStream in = u.openStream( );
  jep.read(in, doc);
}
catch (IOException ex) {
  System.err.println(ex);
}

JScrollPane scrollPane = new JScrollPane(jep);
JFrame f = new JFrame("Macfaq");
f.setContentPane(scrollPane);
f.setSize(512, 342);
EventQueue.invokeLater(new FrameShower(f));
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

This would also be useful if you need to interpose your program in the stream to perform some sort of filtering. For example, you might want to remove `IMG` tags from the file before displaying it. The methods of the next section can help you do this.

8.3. Parsing HTML

Sometimes you want to read HTML, looking for information without actually displaying it on the screen. For instance, more than one author I know has written a "book ticker" program to track the hour-by-hour progress of their books in the Amazon.com bestseller list. The hardest part of this program isn't retrieving the HTML. It's reading through the HTML to find the one line that contains the book's ranking. As another example, consider a Web Whacker-style program that downloads a web site or part thereof to a local PC with all links intact. Downloading the files once you have the URLs is easy. But reading through the document to find the URLs of the linked pages is considerably more complex.

Both of these examples are parsing problems. While parsing a clearly defined language that doesn't allow syntax errors, such as Java or XML, is relatively straightforward, parsing a flexible language that attempts to recover from errors, like HTML, is extremely difficult. It's easier to write in HTML than it is to write in a strict language like XML, but it's much harder to read such a language. Ease of use for the page author has been favored at the cost of ease of development for the programmer.

Fortunately, the `javax.swing.text.html` and `javax.swing.text.html.parser` packages include classes that do most of the hard work for you. They're primarily intended for the internal use of the `JEditorPane` class discussed in the last section. Consequently, they can be a little tricky to get at. The constructors are often not public or hidden inside inner classes, and the classes themselves aren't very well documented. But once you've seen a few examples, they aren't hard to use.

8.3.1. HTMLEditorKit.Parser

The main HTML parsing class is the inner class `javax.swing.html.HTMLEditorKit.Parser`:

```
public abstract static class HTMLEditorKit.Parser extends Object
```

Since this is an abstract class, the actual parsing work is performed by an instance of its concrete subclass `javax.swing.text.html.parser.ParserDelegator`:

```
public class ParserDelegator extends HTMLEditorKit.Parser
```

An instance of this class reads an HTML document from a `Reader`. It looks for five things in the document: start-tags, end-tags, empty-element tags, text, and comments. That covers all the important parts of a common HTML file. (Document type declarations and processing instructions are omitted, but they're rare and not very important in most HTML files, even when they are included.) Every time the parser sees one of these five items, it invokes the corresponding callback method in a particular instance of the `javax.swing.text.html.HTMLEditorKit.ParserCallback` class. To parse an HTML file, you write a subclass of `HTMLEditorKit.ParserCallback` that responds to text and tags as you desire. Then you pass an instance of your subclass to the `HTMLEditorKit.Parser`'s `parse( )` method, along with the `Reader` from which the HTML will be read:

```
public void parse(Reader in, HTMLEditorKit.ParserCallback callback,
 boolean ignoreCharacterSet) throws IOException
```

The third argument indicates whether you want to be notified of the character set of the document, assuming one is found in a `META` tag in the HTML header. This will normally be true. If it's false, then the parser will throw a `javax.swing.text.ChangedCharSetException` when a `META` tag in the HTML header is used to change the character set. This would give you an opportunity to switch to a different `Reader` that understands that character set and reparse the document (this time, setting `ignoreCharSet` to true since you already know the character set).

`parse( )` is the only public method in the `HTMLEditorKit.Parser` class. All the work is handled inside the callback methods in the `HTMLEditorKit.ParserCallback` subclass. The `parse( )` method simply reads from the `Reader in` until it's read the entire document. Every time it sees a tag, comment, or block of text, it invokes the corresponding callback method in the `HTMLEditorKit.ParserCallback` instance. If the `Reader` throws an `IOException`, that exception is passed along. Since neither the `HTMLEditorKit.Parser` nor the `HTMLEditorKit.ParserCallback` instance is specific to one reader, it can be used to parse multiple files simply by invoking `parse( )` multiple times. If you do this, your `HTMLEditorKit.ParserCallback` class must be fully thread-safe, because parsing takes place in a separate thread and the `parse( )` method normally returns before parsing is complete.

Before you can do any of this, however, you have to get your hands on an instance of the `HTMLEditorKit.Parser` class, and that's harder than it should be. `HTMLEditorKit.Parser` is an abstract class, so it can't be instantiated directly. Its subclass, `javax.swing.text.html.parser.ParserDelegator`, is concrete.

However, before you can use it, you have to configure it with a DTD, using the protected static methods `ParserDelegator.setDefaultDTD( )` and `ParserDelegator.createDTD( )`:

```
protected static void setDefaultDTD( )
protected static DTD createDTD(DTD dtd, String name)
```

So to create a `ParserDelegator`, you first need to have an instance of `javax.swing.text.html.parser.DTD`. This class represents a Standardized General Markup Language (SGML) document type definition. The `DTD` class has a protected constructor and many protected methods that subclasses can use to build a DTD from scratch, but this is an API that only an SGML expert could be expected to use. The normal way DTDs are created is by reading the text form of a standard DTD published by someone like the W3C. You should be able to get a DTD for HTML by using the `DTDParser` class to parse the W3C's published HTML DTD. Unfortunately, the `DTDParser` class isn't included in the published Swing API, so you can't. Thus, you're going to need to go through the back door to create an `HTMLEditorKit.Parser` instance. What we'll do is use the `HTMLEditorKit.Parser.getParser( )` method instead, which ultimately returns a `ParserDelegator` after properly initializing the DTD for HTML 3.2:

```
protected HTMLEditorKit.Parser getParser( )
```

Since this method is protected, we'll simply subclass `HTMLEditorKit` and override it with a public version, as Example 8-6 demonstrates.

Example 8-6. This subclass just makes the getParser( ) method public

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

public class ParserGetter extends HTMLEditorKit {

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

}
```

Now that you've got a way to get a parser, you're ready to parse some documents. This is accomplished through the `parse( )` method of `HTMLEditorKit.Parser`:

```
public abstract void parse(Reader input, HTMLEditorKit.ParserCallback
 callback, boolean ignoreCharSet) throws IOException
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

The `Reader` is straightforward. Simply chain an `InputStreamReader` to the stream reading the HTML document, probably one returned by the `openStream()` method of `java.net.URL`. For the third argument, you can pass true to ignore encoding issues (this generally works only if you're pretty sure you're dealing with ASCII text) or false if you want to receive a `ChangedCharSetException` when the document has a `META` tag indicating the character set. The second argument is where the action is. You're going to write a subclass of `HTMLEditorKit.ParserCallback` that is notified of every start-tag, end-tag, empty-element tag, text, comment, and error that the parser encounters.

8.3.2. HTMLEditorKit.ParserCallback

The `ParserCallback` class is a public inner class inside `javax.swing.text.html.HTMLEditorKit`:

```
public static class HTMLEditorKit.ParserCallback extends Object
```

It has a single, public noargs constructor:

```
public HTMLEditorKit.ParserCallback( )
```

However, you probably won't use this directly because the standard implementation of this class does nothing. It exists to be subclassed. It has six callback methods that do nothing. You will override these methods to respond to specific items seen in the input stream as the document is parsed:

```
public void handleText(char[] text, int position)
public void handleComment(char[] text, int position)
public void handleStartTag(HTML.Tag tag,
 MutableAttributeSet attributes, int position)
public void handleEndTag(HTML.Tag tag, int position)
public void handleSimpleTag(HTML.Tag tag,
 MutableAttributeSet attributes, int position)
public void handleError(String errorMessage, int position)
```

There's also a `flush( )` method you use to perform any final cleanup. The parser invokes this method once after it's finished parsing the document:

```
public void flush( ) throws BadLocationException
```

Let's begin with a simple example. Suppose you want to write a program that strips out all the tags and comments from an HTML document and leaves only the text. You would write a subclass of `HTMLEditorKit.ParserCallback` that overrides the `handleText( )`

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

method to write the text on a `Writer`. You would leave the other methods alone. Example 8-7 demonstrates.

Example 8-7. TagStripper

```
import javax.swing.text.html.*;
import java.io.*;


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);
    }
  }

}
```

Now let's suppose you want to use this class to actually strip the tags from a URL. You begin by retrieving a parser using Example 8-5s `ParserGetter` class:

```
ParserGetter kit = new ParserGetter( );
HTMLEditorKit.Parser parser = kit.getParser( );
```

Next, construct an instance of your callback class like this:

```
HTMLEditorKit.ParserCallback callback
 = new TagStripper(new OutputStreamWriter(System.out));
```

Then you get a stream you can read the HTML document from. For example:

```
try {
  URL u = new URL("http://www.oreilly.com");
  InputStream in = new BufferedInputStream(u.openStream( ));
  InputStreamReader r = new InputStreamReader(in);
```

Finally, you pass the `Reader` and the `HTMLEditorKit.ParserCallback` to the `HTMLEditorKit.Parser`'s `parse( )` method, like this:

```
      parser.parse(r, callback, false);
    }
    catch (IOException ex) {
      System.err.println(ex);
    }
```

There are a couple of details about the parsing process that are not obvious. First, the parser parses in a separate thread. Therefore, you should not assume that the document has been parsed when the `parse( )` method returns. If you're using the same `HTMLEditorKit.ParserCallback` object for two separate parses, you need to make all your callback methods thread-safe.

Second, the parser actually skips some of the data in the input. In particular, it normalizes and strips whitespace. If the input document contains seven spaces in a row, the parser will convert that to a single space. Carriage returns, linefeeds, and tabs are all converted to a single space, so you lose line breaks. Furthermore, most text elements are stripped of *all* leading and trailing whitespace. Elements that contain nothing but space are eliminated completely. Thus, suppose the input document contains this content:

```
<H1> Here's   the   Title </H1>

<P> Here's the text </P>
```

What actually comes out of the tag stripper is:

```
Here's the TitleHere's the text
```

The single exception is the `PRE` element, which maintains all whitespace in its contents unedited. Short of implementing your own parser, I don't know of any way to retain all the stripped space. But you can include the minimum necessary line breaks and whitespace by looking at the tags as well as the text. Generally, you expect a single break in HTML when you see one of these tags:

```
<BR>
<LI>
<TR>
```

You expect a double break (paragraph break) when you see one of these tags:

```
<P>
</H1> </H2> </H3> </H4> </H5> </H6>
<HR>
<DIV>
</UL> </OL> </DL>
```

To include line breaks in the output, you have to look at each tag as it's processed and determine whether it falls in one of these sets. This is straightforward because the first argument passed to each of the tag callback methods is an `HTML.Tag` object.

8.3.3. HTML.Tag

`Tag` is a public inner class in the `javax.swing.text.html.HTML` class.

```
public static class HTML.Tag extends Object
```

It has these four methods:

```
public boolean isBlock( )
public boolean breaksFlow( )
public boolean isPreformatted( )
public String  toString( )
```

The `breaksFlow( )` method returns true if the tag should cause a single line break. The `isBlock()` method returns true if the tag should cause a double line break. The `isPreformatted()` method returns true if the tag indicates that whitespace should be preserved. This makes it easy to provide the necessary breaks in the output.

Chances are you'll see more tags than you'd expect when you parse a file. The parser inserts missing closing tags. In other words, if a document contains only a `<P>` tag, then the parser will report both the `<P>` start-tag and the implied `</P>` end-tag at the appropriate points in the document. Example 8-8 is a program that does the best job yet of converting HTML to pure text. It looks for the empty and end-tags, explicit or implied, and, if the tag indicates that line breaks are called for, inserts the necessary number of line breaks.

Example 8-8. LineBreakingTagStripper

```
import javax.swing.text.*;
import javax.swing.text.html.*;
import javax.swing.text.html.parser.*;
import java.io.*;
import java.net.*;

public class LineBreakingTagStripper
 extends HTMLEditorKit.ParserCallback {

  private Writer out;
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
    private String lineSeparator;

    public LineBreakingTagStripper(Writer out) {
      this(out, System.getProperty("line.separator", "\r\n"));
    }

    public LineBreakingTagStripper(Writer out, String lineSeparator) {
      this.out = out;
      this.lineSeparator = lineSeparator;
    }

    public void handleText(char[] text, int position) {
      try {
        out.write(text);
        out.flush( );
      }
      catch (IOException ex) {
        System.err.println(ex);
      }
    }

    public void handleEndTag(HTML.Tag tag, int position) {

      try {
        if (tag.isBlock( )) {
          out.write(lineSeparator);
          out.write(lineSeparator);
        }
        else if (tag.breaksFlow( )) {
          out.write(lineSeparator);
        }
      }
      catch (IOException ex) {
        System.err.println(ex);
      }

    }
    public void handleSimpleTag(HTML.Tag tag,
     MutableAttributeSet attributes, int position) {

      try {
        if (tag.isBlock( )) {
          out.write(lineSeparator);
          out.write(lineSeparator);
        }
        else if (tag.breaksFlow( )) {
          out.write(lineSeparator);
        }
        else {
          out.write(' ');
        }
      }
      catch (IOException ex) {
        System.err.println(ex);
      }

    }

  }
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

Most of the time, of course, you want to know considerably more than whether a tag breaks a line. You want to know what tag it is, and behave accordingly. For instance, if you were writing a full-blown HTML-to-TeX or HTML-to-RTF converter, you'd want to handle each tag differently. You test the type of tag by comparing it against these 73 mnemonic constants from the `HTML.Tag` class:

HTML.Tag.A	HTML.Tag.FRAMESET	HTML.Tag.PARAM
HTML.Tag.ADDRESS	HTML.Tag.H1	HTML.Tag.PRE
HTML.Tag.APPLET	HTML.Tag.H2	HTML.Tag.SAMP
HTML.Tag.AREA	HTML.Tag.H3	HTML.Tag.SCRIPT
HTML.Tag.B	HTML.Tag.H4	HTML.Tag.SELECT
HTML.Tag.BASE	HTML.Tag.H5	HTML.Tag.SMALL
HTML.Tag.BASEFONT	HTML.Tag.H6	HTML.Tag.STRIKE
HTML.Tag.BIG	HTML.Tag.HEAD	HTML.Tag.S
HTML.Tag.BLOCKQUOTE	HTML.Tag.HR	HTML.Tag.STRONG
HTML.Tag.BODY	HTML.Tag.HTML	HTML.Tag.STYLE
HTML.Tag.BR	HTML.Tag.I	HTML.Tag.SUB
HTML.Tag.CAPTION	HTML.Tag.IMG	HTML.Tag.SUP
HTML.Tag.CENTER	HTML.Tag.INPUT	HTML.Tag.TABLE
HTML.Tag.CITE	HTML.Tag.ISINDEX	HTML.Tag.TD
HTML.Tag.CODE	HTML.Tag.KBD	HTML.Tag.TEXTAREA

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

HTML.Tag.DD	HTML.Tag.LI	HTML.Tag.TH
HTML.Tag.DFN	HTML.Tag.LINK	HTML.Tag.TR
HTML.Tag.DIR	HTML.Tag.MAP	HTML.Tag.TT
HTML.Tag.DIV	HTML.Tag.MENU	HTML.Tag.U
HTML.Tag.DL	HTML.Tag.META	HTML.Tag.UL
HTML.Tag.DT	HTML.Tag.NOFRAMES	HTML.Tag.VAR
HTML.Tag.EM	HTML.Tag.OBJECT	HTML.Tag.IMPLIED
HTML.Tag.FONT	HTML.Tag.OL	HTML.Tag.COMMENT
HTML.Tag.FORM	HTML.Tag.OPTION	
HTML.Tag.FRAME	HTML.Tag.P	

These are not `int` constants. They are object constants to allow compile-time type checking. You saw this trick once before in the `javax.swing.event.HyperlinkEvent` class. All `HTML.Tag` elements passed to your callback methods by the `HTMLEditorKit.Parser` will be one of these 73 constants. They are not just the *same as* these 73 objects; they *are* these 73 objects. There are exactly 73 objects in this class; no more, no less. You can test against them with == rather than `equals( )`.

For example, let's suppose you need a program that outlines HTML pages by extracting their `H1` through `H6` headings while ignoring the rest of the document. It organizes the outline as nested lists in which each `H1` heading is at the top level, each `H2` heading is one level deep, and so on. You would write an `HTMLEditorKit.ParserCallback` subclass that extracted the contents of all `H1`, `H2`, `H3`, `H4`, `H5`, and `H6` elements while ignoring all others, as Example 8-9 demonstrates.

Example 8-9. Outliner

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
import javax.swing.text.*;
import javax.swing.text.html.*;
import javax.swing.text.html.parser.*;
import java.io.*;
import java.net.*;
import java.util.*;

public class Outliner extends HTMLEditorKit.ParserCallback {

  private Writer out;
  private int level = 0;
  private boolean inHeader=false;
  private static String lineSeparator
    = System.getProperty("line.separator", "\r\n");

  public Outliner(Writer out) {
    this.out = out;
  }

  public void handleStartTag(HTML.Tag tag,
   MutableAttributeSet attributes, int position) {

    int newLevel = 0;
    if (tag == HTML.Tag.H1) newLevel = 1;
    else if (tag == HTML.Tag.H2) newLevel = 2;
    else if (tag == HTML.Tag.H3) newLevel = 3;
    else if (tag == HTML.Tag.H4) newLevel = 4;
    else if (tag == HTML.Tag.H5) newLevel = 5;
    else if (tag == HTML.Tag.H6) newLevel = 6;
    else return;

    this.inHeader = true;
    try {
      if (newLevel > this.level) {
        for (int i =0; i < newLevel-this.level; i++) {
          out.write("<ul>" + lineSeparator + "<li>");
        }
      }
      else if (newLevel < this.level) {
        for (int i =0; i < this.level-newLevel; i++) {
          out.write(lineSeparator + "</ul>" + lineSeparator);
        }
        out.write(lineSeparator + "<li>");
      }
      else {
        out.write(lineSeparator + "<li>");
      }
      this.level = newLevel;
      out.flush( );
    }
    catch (IOException ex) {
      System.err.println(ex);
    }

  }

  public void handleEndTag(HTML.Tag tag, int position) {
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
      if (tag == HTML.Tag.H1 || tag == HTML.Tag.H2
       || tag == HTML.Tag.H3 || tag == HTML.Tag.H4
       || tag == HTML.Tag.H5 || tag == HTML.Tag.H6) {
        inHeader = false;
      }

      // work around bug in the parser that fails to call flush
      if (tag == HTML.Tag.HTML) this.flush( );

    }


    public void handleText(char[] text, int position) {

      if (inHeader) {
        try {
          out.write(text);
          out.flush( );
        }
        catch (IOException ex) {
          System.err.println(ex);
        }
      }

    }

    public void flush( ) {
      try {
        while (this.level-- > 0) {
          out.write(lineSeparator + "</ul>");
        }
        out.flush( );
      }
      catch (IOException e) {
        System.err.println(e);
      }
    }

    private static void parse(URL url, String encoding) throws IOException {
        ParserGetter kit = new ParserGetter( );
        HTMLEditorKit.Parser parser = kit.getParser( );
        InputStream in = url.openStream( );
        InputStreamReader r = new InputStreamReader(in, encoding);
        HTMLEditorKit.ParserCallback callback = new Outliner
         (new OutputStreamWriter(System.out));
        parser.parse(r, callback, true);
    }

    public static void main(String[] args) {

      ParserGetter kit = new ParserGetter( );
      HTMLEditorKit.Parser parser = kit.getParser( );

      String encoding = "ISO-8859-1";
      URL url = null;
      try {
        url = new URL(args[0]);
        InputStream in = url.openStream( );
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
          InputStreamReader r = new InputStreamReader(in, encoding);
          // parse once just to detect the encoding
          HTMLEditorKit.ParserCallback doNothing
           = new HTMLEditorKit.ParserCallback( );
          parser.parse(r, doNothing, false);
        }
        catch (MalformedURLException ex) {
          System.out.println("Usage: java Outliner url");
          return;
        }
        catch (ChangedCharSetException ex) {
          String mimeType = ex.getCharSetSpec( );
          encoding = mimeType.substring(mimeType.indexOf("=") + 1).trim( );
        }
        catch (IOException ex) {
          System.err.println(ex);
        }
        catch (ArrayIndexOutOfBoundsException ex) {
          System.out.println("Usage: java Outliner url");
          return;
        }

        try {
          parse(url, encoding);
        }
        catch(IOException ex) {
          System.err.println(ex);
        }

      }

    }
```

When a heading start-tag is encountered by the `handleStartTag( )` method, the necessary number of `<ul>`, `</ul>`, and `<li>` tags are emitted. Furthermore, the `inHeading` flag is set to true so that the `handleText( )` method will know to output the contents of the heading. All start-tags except the six levels of headers are simply ignored. The `handleEndTag( )` method likewise considers heading tags only by comparing the tag it receives with the seven tags it's interested in. If it sees a heading tag, it sets the `inHeading` flag to false again so that body text won't be emitted by the `handleText( )` method. If it sees the end of the document via an `</html>` tag, it flushes out the document. Otherwise, it does nothing. The end result is a nicely formatted group of nested, unordered lists that outlines the document. For example, here's the output of running it against http://www.cafeconleche.org:

```
% java Outliner http://www.cafeconleche.org/
<ul>
<li> Cafe con Leche XML News and Resources<ul>
<li>Quote of the Day
<li>Today's News
<li>Recommended Reading
<li>Recent News<ul>
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
<li>XML Overview
<li>Tutorials
<li>Projects
<li>Seminar Notes
<li>Random Notes
<li>Specifications
<li>Books
<li>XML Resources
<li>Development Tools<ul>
<li>Validating Parsers
<li>Non-validating Parsers
<li>Online Validators and Syntax Checkers
<li>Formatting Engines
<li>Browsers
<li>Class Libraries
<li>Editors
<li>XML Applications
<li>External Sites
</ul>
</ul>
</ul>
</ul>
```

8.3.4. Attributes

When processing an HTML file, you often need to look at the attributes as well as the tags. The second argument to the `handleStartTag( )` and `handleSimpleTag( )` callback methods is an instance of the `javax.swing.text.MutableAttributeSet` class. This object allows you to see what attributes are attached to a particular tag. `MutableAttributeSet` is a subinterface of the `javax.swing.text.AttributeSet` interface:

```
public abstract interface MutableAttributeSet extends AttributeSet
```

Both `AttributeSet` and `MutableAttributeSet` represent a collection of attributes on an HTML tag. The difference is that the `MutableAttributeSet` interface declares methods to add attributes to, remove attributes from, and inspect the attributes in the set. The attributes themselves are represented as pairs of `java.lang.Object` objects, one for the name of the attribute and one for the value. The `AttributeSet` interface declares these methods:

```
public int          getAttributeCount( )
public boolean      isDefined(Object name)
public boolean      containsAttribute(Object name, Object value)
public boolean      containsAttributes(AttributeSet attributes)
public boolean      isEqual(AttributeSet attributes)
public AttributeSet copyAttributes( )
public Enumeration  getAttributeNames( )
public Object       getAttribute(Object name)
public AttributeSet getResolveParent( )
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

Most of these methods are self-explanatory. The `getAttributeCount( )` method returns the number of attributes in the set. The `isDefined( )` method returns true if an attribute with the specified name is in the set, false otherwise. The `containsAttribute(Object name, Object value)` method returns true if an attribute with the given name and value is in the set. The `containsAttributes(AttributeSet attributes)` method returns true if all the attributes in the specified set are in this set with the same values; in other words, if the argument is a subset of the set on which this method is invoked. The `isEqual()` method returns true if the invoking `AttributeSet` is the same as the argument. The `copyAttributes( )` method returns a clone of the current `AttributeSet`. The `getAttributeNames( )` method returns a `java.util.Enumeration` of all the names of the attributes in the set. Once you know the name of one of the elements of the set, the `getAttribute( )` method returns its value. Finally, the `getResolveParent( )` method returns the parent `AttributeSet`, which will be searched for attributes that are not found in the current set. For example, given an `AttributeSet`, this method prints the attributes in name=value format:

```
private void listAttributes(AttributeSet attributes) {
  Enumeration e = attributes.getAttributeNames( );
  while (e.hasMoreElements( )) {
    Object name = e.nextElement( );
    Object value = attributes.getAttribute(name);
    System.out.println(name + "=" + value);
  }
}
```

Although the argument and return types of these methods are mostly declared in terms of `java.lang.Object`, in practice, all values are instances of `java.lang.String`, while all names are instances of the public inner class `javax.swing.text.html.HTML.Attribute`. Just as the `HTML.Tag` class predefines 73 HTML tags and uses a private constructor to prevent the creation of others, so too does the `HTML.Attribute` class predefine 80 standard HTML attributes (`HTML.Attribute.ACTION`, `HTML.Attribute.ALIGN`, `HTML.Attribute.ALINK`, `HTML.Attribute.ALT`, etc.) and prohibits the construction of others via a nonpublic constructor. Generally, this isn't an issue, since you mostly use `getAttribute( )`, `containsAttribute()`, and so forth only with names returned by `getAttributeNames( )`. The 80 predefined attributes are:

`HTML.Attribute.ACTION`	`HTML.Attribute.DUMMY`	`HTML.Attribute.PROMPT`
`HTML.Attribute.ALIGN`	`HTML.Attribute.ENCTYPE`	`HTML.Attribute.REL`

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

HTML.Attribute.ALINK	HTML.Attribute.ENDTAG	HTML.Attribute.REV
HTML.Attribute.ALT	HTML.Attribute.FACE	HTML.Attribute.ROWS
HTML.Attribute.ARCHIVE	HTML.Attribute.FRAMEBORDER	HTML.Attribute.ROWSPAN
HTML.Attribute.BACKGROUND	HTML.Attribute.HALIGN	HTML.Attribute. SCROLLING
HTML.Attribute.BGCOLOR	HTML.Attribute.HEIGHT	HTML.Attribute.SELECTED
HTML.Attribute.BORDER	HTML.Attribute.HREF	HTML.Attribute.SHAPE
HTML.Attribute. CELLPADDING	HTML.Attribute.HSPACE	HTML.Attribute.SHAPES
HTML.Attribute. CELLSPACING	HTML.Attribute.HTTPEQUIV	HTML.Attribute.SIZE
HTML.Attribute.CHECKED	HTML.Attribute.ID	HTML.Attribute.SRC
HTML.Attribute.CLASS	HTML.Attribute.ISMAP	HTML.Attribute.STANDBY
HTML.Attribute.CLASSID	HTML.Attribute.LANG	HTML.Attribute.START
HTML.Attribute.CLEAR	HTML.Attribute.LANGUAGE	HTML.Attribute.STYLE
HTML.Attribute.CODE	HTML.Attribute.LINK	HTML.Attribute.TARGET
HTML.Attribute.CODEBASE	HTML.Attribute.LOWSRC	HTML.Attribute.TEXT
HTML.Attribute.CODETYPE	HTML.Attribute. MARGINHEIGHT	HTML.Attribute.TITLE
HTML.Attribute.COLOR	HTML.Attribute.MARGINWIDTH	HTML.Attribute.TYPE
HTML.Attribute.COLS	HTML.Attribute.MAXLENGTH	HTML.Attribute.USEMAP

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

HTML.Attribute.COLSPAN	HTML.Attribute.METHOD	HTML.Attribute.VALIGN
HTML.Attribute.COMMENT	HTML.Attribute.MULTIPLE	HTML.Attribute.VALUE
HTML.Attribute.COMPACT	HTML.Attribute.N	HTML.Attribute. VALUETYPE
HTML.Attribute.CONTENT	HTML.Attribute.NAME	HTML.Attribute.VERSION
HTML.Attribute.COORDS	HTML.Attribute.NOHREF	HTML.Attribute.VLINK
HTML.Attribute.DATA	HTML.Attribute.NORESIZE	HTML.Attribute.VSPACE
HTML.Attribute.DECLARE	HTML.Attribute.NOSHADE	HTML.Attribute.WIDTH
HTML.Attribute.DIR	HTML.Attribute.NOWRAP	

The `MutableAttributeSet` interface adds six methods to add attributes to and remove attributes from the set:

```
public void                addAttribute(Object name, Object value)
public void addAttributes(AttributeSet attributes)
public void removeAttribute(Object name)
public void removeAttributes(Enumeration names)
public void removeAttributes(AttributeSet attributes)
public void setResolveParent(AttributeSet parent)
```

Again, the values are strings and the names are `HTML.Attribute` objects.

One possible use for all these methods is to modify documents before saving or displaying them. For example, most web browsers let you save a page on your hard drive as either HTML or text. However, both these formats lose track of images and relative links. The problem is that most pages are full of relative URLs, and these all break when you move the page to your local machine. Example 8-10 is an application called `PageSaver` that downloads a web page to a local hard drive while keeping all links intact by rewriting all relative URLs as absolute URLs.

The `PageSaver` class reads a series of URLs from the command line. It opens each one in turn and parses it. Every tag, text block, comment, and attribute is copied into a local file. However, all link attributes, such as `SRC`, `LOWSRC`, `CODEBASE`, and `HREF`, are remapped to an absolute URL. Note particularly the extensive use to which the `URL` and

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

`javax.swing.text` classes were put; `PageSaver` could be rewritten with string replacements, but that would be considerably more complicated.

Example 8-10. PageSaver

```
import javax.swing.text.*;
import javax.swing.text.html.*;
import javax.swing.text.html.parser.*;
import java.io.*;
import java.net.*;
import java.util.*;

public class PageSaver extends HTMLEditorKit.ParserCallback {

  private Writer out;
  private URL base;

  public PageSaver(Writer out, URL base) {
    this.out = out;
    this.base = base;
  }

  public void handleStartTag(HTML.Tag tag,
   MutableAttributeSet attributes, int position) {
    try {
      out.write("<" + tag);
      this.writeAttributes(attributes);
      // for the <APPLET> tag we may have to add a codebase attribute
      if (tag == HTML.Tag.APPLET
       && attributes.getAttribute(HTML.Attribute.CODEBASE) == null) {
        String codebase = base.toString( );
        if (codebase.endsWith(".htm") || codebase.endsWith(".html")) {
          codebase = codebase.substring(0, codebase.lastIndexOf('/'));
        }
        out.write(" codebase=\"" + codebase + "\"");
      }
      out.write(">");
      out.flush( );
    }
    catch (IOException ex) {
      System.err.println(ex);
      e.printStackTrace( );
    }
  }

  public void handleEndTag(HTML.Tag tag, int position) {
    try {
      out.write("</" + tag + ">");
      out.flush( );
    }
    catch (IOException ex) {
      System.err.println(ex);
    }
  }
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
    private void writeAttributes(AttributeSet attributes)
     throws IOException {

      Enumeration e = attributes.getAttributeNames( );
      while (e.hasMoreElements( )) {
        Object name = e.nextElement( );
        String value = (String) attributes.getAttribute(name);
        try {
          if (name == HTML.Attribute.HREF || name == HTML.Attribute.SRC
           || name == HTML.Attribute.LOWSRC
           || name == HTML.Attribute.CODEBASE ) {
            URL u = new URL(base, value);
            out.write(" " + name + "=\"" + u + "\"");
          }
          else {
            out.write(" " + name + "=\"" + value + "\"");
          }
        }
        catch (MalformedURLException ex) {
          System.err.println(ex);
          System.err.println(base);
          System.err.println(value);
          ex.printStackTrace( );
        }
      }
    }

    public void handleComment(char[] text, int position) {

      try {
        out.write("<!-- ");
        out.write(text);
        out.write(" -->");
        out.flush( );
      }
      catch (IOException ex) {
        System.err.println(ex);
      }

    }

    public void handleText(char[] text, int position) {

      try {
        out.write(text);
        out.flush( );
      }
      catch (IOException ex) {
        System.err.println(ex);
        e.printStackTrace( );
      }

    }

    public void handleSimpleTag(HTML.Tag tag,
     MutableAttributeSet attributes, int position) {
      try {
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
            out.write("<" + tag);
            this.writeAttributes(attributes);
            out.write(">");
          }
          catch (IOException ex) {
            System.err.println(ex);
            e.printStackTrace( );
          }
        }

        public static void main(String[] args) {

          for (int i = 0; i < args.length; i++) {

            ParserGetter kit = new ParserGetter( );
            HTMLEditorKit.Parser parser = kit.getParser( );

            try {
              URL u = new URL(args[i]);
              InputStream in = u.openStream( );
              InputStreamReader r = new InputStreamReader(in);
              String remoteFileName = u.getFile( );
              if (remoteFileName.endsWith("/")) {
                remoteFileName += "index.html";
              }
              if (remoteFileName.startsWith("/")) {
                remoteFileName = remoteFileName.substring(1);
              }
              File localDirectory = new File(u.getHost( ));
              while (remoteFileName.indexOf('/') > -1) {
                String part = remoteFileName.substring(0, remoteFileName.
                                                              indexOf('/'));
                remoteFileName =
                      remoteFileName.substring(remoteFileName.indexOf('/')+1);
                localDirectory = new File(localDirectory, part);
              }
              if (localDirectory.mkdirs( )) {
                File output = new File(localDirectory, remoteFileName);
                FileWriter out = new FileWriter(output);
                HTMLEditorKit.ParserCallback callback = new PageSaver(out, u);
                parser.parse(r, callback, false);
              }
            }
            catch (IOException ex) {
              System.err.println(ex);
              e.printStackTrace( );
            }

          }

        }

      }
```

The `handleEndTag( )`, `handleText()`, and `handleComment( )` methods simply copy their content from the input into the output. The `handleStartTag( )` and `handleSimpleTag( )` methods write their respective tags onto the output but also invoke

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

the private `writeAttributes( )` method. This method loops through the attributes in the set and mostly just copies them onto the output. However, for a few select attributes, such as `SRC` and `HREF`, which typically have URL values, it rewrites the values as absolute URLs. Finally, the `main( )` method reads URLs from the command line, calculates reasonable names and directories for corresponding local files, and starts a new `PageSaver` for each URL.

In the first edition of this book, I included a similar program that downloaded the raw HTML using the `URL` class and parsed it manually. That program was about a third longer than this one and much less robust. For instance, it did not support frames or the `LOWSRC` attributes of `IMG` tags. It went to great effort to handle both quoted and unquoted attribute values and still didn't recognize attribute values enclosed in single quotes. By contrast, this program needs only one extra line of code to support each additional attribute. It is much more robust, much easier to understand (since there's not a lot of detailed string manipulation), and much easier to extend.

This is just one example of the various HTML filters that the `javax.swing.text.html` package makes easy to write. You could, for example, write a filter that pretty-prints the HTML by indenting the different levels of tags. You could write a program to convert HTML to TeX, XML, RTF, or many other formats. You could write a program that spiders a web site, downloading all linked pages—and this is just the beginning. All of these programs are much easier to write because Swing provides a simple-to-use HTML parser. All you have to do is respond to the individual elements and attributes that the parser discovers in the HTML document. The more difficult problem of parsing the document is removed.

8.4. Cookies

Cookies are an atrocious hack perpetrated on the browsing world by Netscape. They are completely contrary to the web architecture. They attempt to graft state onto the deliberately stateless HTTP protocol. Statelessness in HTTP was not a mistake or a design flaw. It was a deliberate design decision that helped the Web scale to the enormous size it's reached today.

On the server side, cookies are never necessary and always a bad idea. There is always a cleaner, simpler, more scalable solution that does not involve cookies. Sadly, a lot of server-side developers don't know this and go blindly forward developing web sites that require client-side developers to support cookies.

Prior to Java 1.5, cookies can be supported only by direct manipulation of the HTTP header. When a server sets a cookie, it includes a Set-Cookie field like this one in the HTTP header:

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
Set-Cookie: user=elharo
```

This sends the browser a cookie with the name "user" and the value "elharo". The value of this field is limited to the printable ASCII characters (because HTTP header fields are limited to the printable ASCII characters). Furthermore, the names may not contain commas, semicolons, or whitespace.

A later version of the spec, RFC 2965, uses a Set-Cookie2 HTTP header instead. The most obvious difference is that this version of the cookie spec requires a version attribute after the name=value pair, like so:

```
Set-Cookie2: user=elharo; Version=1
```

The Version attribute simply indicates the version of the cookie spec in use. Version 1 and the unmarked original version zero are the only ones currently defined. Some servers will send both Set-Cookie and Set-Cookie2 headers. If so, the value in Set-Cookie2 takes precedence if a client understands both. Set-Cookie2 also allows cookie values to be quoted so they can contain internal whitespace. For example, this sets the cookie with the name food and the value "chocolate ice cream".

```
Set-Cookie2: food="chocolate ice cream"; Version=1
```

The quotes are just delimiters. They are not part of the attribute value. However, the attribute values are still limited to printable ASCII characters.

When requesting a document from the same server, the client echoes that cookie back in a Cookie header field in the request it sends to the server:

```
Cookie: user=elharo
```

If the original cookie was set by Set-Cookie2, this begins with a $Version attribute:

```
Cookie: $Version=1;user=elharo
```

The $ sign helps distinguish between cookie attributes and the main cookie name=value pair.

The client's job is simply to keep track of all the cookies it's been sent, and send the right ones back to the original servers at the right time. However, this is a little more complicated because cookies can have attributes identifying the expiration date, path, domain, port, version, and security options.

For example, by default a cookie applies to the server it came from. If a cookie is originally set by *www.foo.example.com*, the browser will only send the cookie back to *www.foo.example.com*. However, a site can also indicate that a cookie applies within an entire subdomain, not just at the original server. For example, this request sets a user cookie for the *entire.foo.example.com* domain:

```
Set-Cookie: user=elharo;Domain=.foo.example.com
```

The browser will echo this cookie back not just to *www.foo.example.com* but also to *lothar.foo.example.com*, *eliza.foo.example.com*, *enoch.foo.example.com*, and any other host somewhere in the *foo.example.com* domain. However, a server can only set cookies for domains it immediately belongs to. *www.foo.example.com* cannot set a cookie for www.oreilly.com, example.com, or .com, no matter how it sets the domain. (In practice, there have been a number of holes and workarounds for this, with severe negative impacts on user privacy.)

If the cookie was set by Set-Cookie2, the client will include the domain that was originally set, like so:

```
Cookie: $Version=1; user=elharo;  $Domain=.foo.example.com
```

However, if it's a version zero cookie, the domain is not echoed back.

Beyond domains, cookies are scoped by path, so they're used for some directories on the server, but not all. The default scope is the original URL and any subdirectories. For instance, if a cookie is set for the URL http://www.cafeconleche.org/XOM/, the cookie also applies in http://www.cafeconleche.org/XOM/apidocs/, but not in http://www.cafeconleche.org/slides/ or http://www.cafeconleche.org/. However, the default scope can be changed using a `Path` attribute in the cookie. For example, this next response sends the browser a cookie with the name "user" and the value "elharo" that applies only within the server's */restricted* subtree, not on the rest of the site:

```
Set-Cookie: user=elharo; $Version=1;Path=/restricted
```

When requesting a document in the subtree */restricted* from the same server, the client echoes that cookie back. However, it does not use the cookie in other directories on the site. Again, if and only if the cookie was originally set with Set-Cookie2, the client will include the Path that was originally set, like so:

```
Cookie: user=elharo; $Version=1;$Path=/restricted
```

A cookie can set include both a domain and a path. For instance, this cookie applies in the */restricted* path on any servers within the *example.com* domain:

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
Set-Cookie2: $Version=1;user=elharo; $Path=/restricted;$Domain=.example.com
```

The order of the different cookie attributes doesn't matter, as long as they're all separated by semicolons and the cookie's own name and value come first. However, this isn't true when the client is sending the cookie back to the server. In this case, the path must precede the domain, like so:

```
Cookie: $Version=1;user=elharo; $Path=/restricted;$Domain=.foo.example.com
```

A version zero cookie can be set to expire at a certain point in time by setting the `expires` attribute to a date in the form Wdy, DD-Mon-YYYY HH:MM:SS GMT. Weekday and month are given as three-letter abbreviations. The rest are numeric, padded with initial zeros if necessary. In the pattern language used by `java.text.SimpleDateFormat`, this is `"E, dd-MMM-yyyy k:m:s 'GMT'"`. For instance, this cookie expires at 3:23 P.M. on December 21, 2005:

```
Set-Cookie: user=elharo; expires=Wed, 21-Dec-2005 15:23:00 GMT
```

The browser should remove this cookie from its cache after that date has passed.

Set-Cookie2 use a Max-Age attribute that sets the cookie to expire after a certain number of seconds have passed instead of at a specific moment. For instance, this cookie expires one hour (3,600 seconds) after it's first set:

```
Set-Cookie2: user="elharo"; $Version=1;Max-Age=3600
```

The browser should remove this cookie from its cache after this amount of time has elapsed.

Because cookies can contain sensitive information such as passwords and session keys, some cookie transactions should be secure. Exactly what secure means in this context is not specified. Most of the time, it means using HTTPS instead of HTTP, but whatever it means, each cookie can have the a secure attribute with no value, like so:

```
Set-Cookie: key=etrogl7*;Domain=.foo.example.com; secure
```

Browsers are supposed to refuse to send such cookies over insecure channels.

Finally, in addition to path, domain, and time, version 1 cookies can be scoped by port. This isn't common, but clients are required to support it. The `Port` attribute contains a quoted list of whitespace-separated port numbers to which the cookie applies:

```
Set-Cookie2: $Version=1;user=elharo; $Path=/restricted;$Port="8080 8000"
```

For the response, the order is always path, domain, and port, like so:

```
Cookie: $Version=1;user=elharo;  $Path=/restricted; $Domain=.foo.example.com;
$Port="8080 8000"
```

Multiple cookies can be set in one request by separating the name-value pairs with commas. For example, this `Set-Cookie` header assigns the cookie named user the value "elharo" and the cookie named `zip` the value "10003":

```
Set-Cookie: user=elharo, zip=10003
```

Each cookie set in this way can also contain attributes. For example, this `Set-Cookie` header scopes the `user` cookie to the path `/restricted` and the `zip` cookie to the path `/weather`:

```
Set-Cookie: user=elharo; path=/restricted, zip=10003; path=/weather
```

I've left out a couple of less important details like comments that don't matter much in practice. If you're interested, complete details are available in RFC 2965, *HTTP State Management Mechanism*.

That's how cookies work behind the scenes. In theory, this is all transparent to the user. In practice, the most sophisticated users routinely disable, filter, or inspect cookies to protect their privacy and security so cookies are not guaranteed to work.

Let's wrap this all up in a neat class called `Cookie`, shown in Example 8-12, with appropriate getter methods for the relevant properties and a factory method that parses HTTP header fields that set cookies. We'll need this in a minute because even as of Java 1.5 there's nothing like this in the standard JDK.

Example 8-11. A cookie class

```
package com.macfaq.http;

import java.net.URI;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Cookie {
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
    private String  version = "0";
    private String  name;
    private String  value;
    private URI   uri;
    private String  domain;
    private Date  expires;
    private String  path;
    private boolean secure = false;

    private static DateFormat expiresFormat
      = new SimpleDateFormat("E, dd-MMM-yyyy k:m:s 'GMT'");

    // prevent instantiation
    private Cookie( ) {}

    public static Cookie bake(String header, URI uri)
      throws CookieException {

      try {
        String[] attributes = header.split(";");
        String nameValue = attributes[0];
        Cookie cookie = new Cookie( );
        cookie.uri = uri;
        cookie.name = nameValue.substring(0, nameValue.indexOf('='));
        cookie.value = nameValue.substring(nameValue.indexOf('=')+1);
        cookie.path = "/";
        cookie.domain = uri.getHost( );

        if (attributes[attributes.length-1].trim( ).equals("secure")) {
          cookie.secure = true;
        }

        for (int i=1; i < attributes.length; i++) {
          nameValue = attributes[i].trim( );
          int equals = nameValue.indexOf('=');
          if (equals == -1) continue;
          String attributeName = nameValue.substring(0, equals);
          String attributeValue = nameValue.substring(equals+1);
          if (attributeName.equalsIgnoreCase("domain")) {
            String uriDomain = uri.getHost( );
            if (uriDomain.equals(attributeValue)) {
              cookie.domain = attributeValue;
            }
            else {
              if (!attributeValue.startsWith(".")) {
                attributeValue = "." + attributeValue;
              }
              uriDomain = uriDomain.substring(uriDomain.indexOf('.'));
              if (!uriDomain.equals(attributeValue)) {
                throw new CookieException(
                 "Server tried to set cookie in another domain");
              }
              cookie.domain = attributeValue;
            }
          }
          else if (attributeName.equalsIgnoreCase("path")) {
            cookie.path = attributeValue;
          }
```

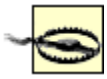

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
            else if (attributeName.equalsIgnoreCase("expires")) {
              cookie.expires = expiresFormat.parse(attributeValue);
            }
            else if (attributeName.equalsIgnoreCase("Version")) {
              if (!"1".equals(attributeValue)) {
                throw new CookieException("Unexpected version " + attributeValue);
              }
              cookie.version = attributeValue;
            }
          }

          return cookie;
        }
        catch (Exception ex) {
          // ParseException, StringIndexOutOfBoundsException etc.
          throw new CookieException(ex);
        }

      }

      public boolean isExpired( ) {
        if (expires == null) return false;
        Date now = new Date( );
        return now.after(expires);
      }

      public String getName( ) {
        return name;
      }

      public boolean isSecure( ) {
        return secure;
      }

      public URI getURI( ) {
        return uri;
      }

      public String getVersion( ) {
        return version;
      }

      // should this cookie be sent when retrieving the specified URI?
      public boolean matches(URI u) {

        if (isExpired( )) return false;

        String path = u.getPath( );
        if (path == null) path = "/";

        if (path.startsWith(this.path)) return true;

        return false;
      }

      public String toExternalForm( ) {
        StringBuffer result = new StringBuffer(name);
        result.append("=");
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
        result.append(value);
        if ("1".equals(version)) {
           result.append(" Version=1");
        }
        return result.toString( );
      }
    }
```

Prior to Java 1.5, the only way to support cookies is by direct inspection of the relevant HTTP headers. The `URL` class does not support this, but the `URLConnection` class introduced in Chapter 15 does. Java 1.5 adds a new `java.net.CookieHandler` class that makes this process somewhat easier. You provide a subclass of this abstract class where Java will store all cookies retrieved through the HTTP protocol handler. Once you've done this, when you access an HTTP server through a `URL` object and the server sends a cookie, Java automatically puts it in the system default cookie handler. When the same VM instance goes back to that server, it sends the cookie.



I'm writing this section based on betas of Java 1.5. While the information about how cookies are handled in HTTP should be accurate, it's entirely possible a few of the Java details may change by the time Java 1.5 is released. Be sure to compare what you read here with the latest documentation from Sun.

The `CookieHandler` class is summarized in Example 8-12. As you can see, there are two abstract methods to implement, `get( )` and `put()`. When Java loads a URL from a server that sets a cookie, it passes the URI it was loading and the complete HTTP headers of the server response to the `put( )` method. The handler can parse the details out of these headers and store them somewhere. When Java tries to load an HTTP URL from a server, it passes the URL and the request HTTP header to the `get( )` method to see if there are any cookies in the store for that URL. Sadly, you have to implement the parsing and storage code yourself. `CookieHandler` is an abstract class that does not do this for you, even though it's pretty standard stuff.

Example 8-12. CookieHandler

```
package java.net;

public abstract class CookieHandler {
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
  public CookieHandler( )

  public abstract Map<String,List<String>> get(
   URI uri, Map<String,List<String>> requestHeaders)
   throws IOException
  public abstract void put(
   URI uri, Map<String,List<String>> responseHeaders)
   throws IOException

  public static CookieHandler getDefault( )
  public static void           setDefault(CookieHandler handler)

}
```

A subclass is most easily implemented by delegating the hard work to the Java Collections API, as Example 8-13 demonstrates. Since `CookieHandler` is only available in Java 1.5 anyway, I took the opportunity to show off some new features of Java 1.5, including generic types and enhanced `for` loops. This implementation limits itself to version 0 cookies, which are far and away the most common kind you'll find in practice. If version 1 cookies ever achieve broad adoption, it should be easy to extend these classes to support them.

Example 8-13. A CookieHandler implemented on top of the Java Collections API

```
package com.macfaq.http;

import java.io.IOException;
import java.net.*;
import java.util.*;

public class CookieStore extends CookieHandler {

  private List<Cookie> store = new ArrayList<Cookie>( );

  public Map<String,List<String>> get(URI uri,
   Map<String,List<String>> requestHeaders)
   throws IOException {

      Map<String,List<String>> result = new HashMap<String,List<String>>( );
      StringBuffer cookies = new StringBuffer( );
      for (Cookie cookie : store) {
        if (cookie.isExpired( )) {
          store.remove(cookie);
        }
        else if (cookie.matches(uri)) {
              if (cookies.length( ) != 0) cookies.append(", ");
              cookies.append(cookie.toExternalForm( ));
        }
      }
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

```
            if (cookies.length( ) > 0) {
              List<String> temp = new ArrayList<String>(1);
              temp.add(cookies.toString( ));
              result.put("Cookie", temp);
            }

            return result;

        }

        public void put(URI uri, Map<String,List<String>> responseHeaders)
         throws IOException {

          List<String> setCookies = responseHeaders.get("Set-Cookie");
          for (String next : setCookies) {
            try {
              Cookie cookie = Cookie.bake(next, uri);
              // Is a cookie with this name and URI already in the list?
              // If so, we replace it
              for (Cookie existingCookie : store) {
                if (cookie.getURI( ).equals(existingCookie.getURI( )) &&
                  cookie.getName( ).equals(existingCookie.getName( ))) {
                    store.remove(existingCookie);
                    break;
                }
              }
              store.add(cookie);
            }
            catch (CookieException ex) {
              // Server sent malformed header;
              // log and ignore
              System.err.println(ex);
            }
          }

        }

      }
```

When storing a cookie, the `responseHeaders` argument to the `put( )` method contains the complete HTTP response header sent by the server. From this you need to extract any header fields that set cookies (basically, just `Set-Cookie` and `Set-Cookie2`). The key to this map is the field name (`Set-Cookie` or `Set-Cookie2`). The value of the map entry is a list of cookies set in that field. Each separate cookie is a separate member of the list. That is, Java does divide the header field value along the commas to split up several cookies and pass them in each as a separate entry in the list.

In the other direction, when getting a cookie it's necessary to consider not only the URI but the path for which the cookie is valid. Here, the path is delegated to the `Cookie` class itself via the `matches()` method. This is hardly the most efficient implementation possible. For each cookie, the store does a linear search through all available cookies. A more intelligent implementation would index the list by URIs and domains, but for simple purposes this solution suffices without being overly complex. A more serious limitation is that this store is

Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004

not persistent. It lasts only until the driving program exits. Most web browsers would want to store the cookies in a file so they could be reloaded when the browser was relaunched. Nonetheless, this class is sufficient to add basic cookie support to the simple web browser. All that's required is to add this one line at the beginning of the `main( )` method in Example 8-5:

```
CookieHandler.setDefault(new com.macfaq.http.CookieStore( ));
```


Java Network Programming, 3rd Edition By Elliotte Rusty Harold ISBN: 0596007213 Publisher: O'Reilly
Print Publication Date: 10/1/2004