TagSoup: A SAX parser in Java for nasty, ugly HTML

John Cowan (*cowan@ccil.org*)

Copyright

Where Is Tag Soup?

- On the World Wide Web
  - About 2 gigadocs
- On about 10,000 corporate intranets
  - Unknown number
- The results of "Save As HTML..."
  - Unknown number

Why Process Tag Soup?

- Programs come and go, data is forever
  - That includes ugly legacy data
  - Almost all data is legacy data at any given time
- Tools improve all the time
  - Parsing legacy data lets us use our modern tools

Using TagSoup

- The Parser class is the main entry point to TagSoup
- To parse HTML:
  - Create an instance of Parser
  - Provide your own SAX2 ContentHandler
  - Provide an InputSource referring to the HTML
  - And parse()!
- Everything else is just details

Guarantees

- Start-tags and end-tags are matched
- Illegal characters in element and attribute names are suppressed
- Windows-specific characters (0x80-0x9F) are mapped to Unicode, independently of character encoding
- Attribute defaulting and normalization are performed

Non-Guarantees

- The generated events may not correspond to any particular XHTML DTD -- no validation is performed

Standards Support

- Supports all HTML 4.01 elements, attributes, and entities, plus a bunch of IE-specific and NS4-specific ones
- Unknown elements are assumed EMPTY
- Unknown attributes are assumed CDATA
- All names are lowercased, as in XHTML

TagSoup Features

- Conforms to SAX2 parsing interface
- Processes HTML as it is, not as it should be
- Written in Java
- Parser plus command-line tool
- "Just Keep On Truckin'"
- Roll your own
- Academic Free License or GPL, your choice

Why SAX2?

- The de-facto standard for streaming XML
- Lightweight and easy to implement
- Suitable for any document size
- May be more difficult for programmers to use than DOM, JDOM, or XOM
- SAX-to-DOM/JDOM/XOM converters readily available

TagSoup SAX2 Support

- Generates the full range of SAX2 events
- Fakes namespaces
- Supports all standard SAX2 features, mostly with fixed values
- Supports a few crucial TagSoup-specific properties

HTML In The Wild

- Tags don't necessarily nest properly
- DOCTYPE declarations are often missing or incorrect
- End tags are often left out
- Start tags are often left out
- Quotation marks around attribute values are often missing

Java

- SAX was originally defined for Java and is still supported best there
- Java has used Unicode since 1.1
- TagSoup doesn't depend on Java-specific features, so it should be easy to port
- TagSoup was developed under Java 1.4, but works fine down to Java 1.1 with a patch or two

A Parser, Not An Application

- Does what is necessary to ensure correct syntax and bare HTML semantics
- Meant to be embedded in XML applications to let them process any HTML as if it were well-formed XML
- Can also be used as a command-line tool to preprocess HTML before passing it to XML applications

TagSoup vs. Tidy

- TagSoup is not a substitute for the HTML Tidy program, which actually cleans up HTML files, converts markup to CSS, etc.
- Tidy comes in both C and Java flavors, but the Java version tends to lag the C version
- TagSoup is more robust than Tidy
  – Will not loop even on highly malformed input

"Just Keep On Truckin'"

- Never throws any syntax errors of any kind
- Never gives up until the whole input has been read, no matter what (unless there is an IOException)
- Only throws SAXExceptions if you misuse features or properties

Academic Free License 3.0

- A cleaned-up BSD license
- TagSoup can be used in Open Source or proprietary software
- If you sue using this or a related license for patent infringement, your license terminates automatically
- I dual-license for GNU GPL compatibility
- *http://opensource.org/licenses/afl-3.0.txt*

How TagSoup Works

The HTML Scanner

- DOCTYPE declarations are ignored completely
- Consequently, external DTDs are not read
- Comments and processing instructions (ending in >, not `?>`) are passed through to the application
- Entity references are expanded or turned into text

Element Rectification

- Rectification takes the incoming stream of start-tags, end-tags, and character data and makes it well-structured
- TagSoup is essentially an HTML scanner plus a schema-driven element rectifier
- TagSoup uses its own schema language compiled into Schema objects

TagSoup Schemas

- Element type: name, alternative content models, and *parent element types* (a new concept)
- Content model: a list of element types (no ordering or quantity rules)
- Attribute: name, type, and default value
- Entity: name and character code
- One schema, one namespace name

Parent Element Types

- Parent element types represent the most *conservative* possible parent of an element
- The schema gives a parent element type for each element type:
  - The parent of `BODY` is `HTML`
  - The parent of `LI` is `UL`
  - The parent of `#PCDATA` is `BODY`

End-Tag Omission

- When the start-tag of an element that can't be a child of the current element is seen, the current element is closed
- This is done recursively until the new start-tag is allowed
- `<P>This is a <A>link<P>More` becomes `<P>This is a <A>link</A></P><P>More`

Start-Tag Omission

- A start-tag that can't be contained in *any* open element implies that some start-tags are missing
- TagSoup supplies a start-tag for the parent element type (recursively if necessary)
- A document that starts with character data will supply `<HTML><BODY>` first

Explicit End-Tags

- An end-tag with no corresponding start-tag is ignored
- An end-tag with a corresponding start-tag closes all descendant elements:
- `<P>This is <I>`*`weird`*`</P>`
  becomes `<P>This is <I>`*`weird`*`</I></P>`
- TagSoup supports XML empty tags and even SGML minimized end-tags (`</>`)

Restartable Tags

- If we need to close an element that is known to be *restartable*, it will be opened again as soon as possible (but not until we are inside an element where it's legal)
- `<P><FONT>a<P>b</FONT>`
  becomes `<P><FONT>a</FONT></P>`
  `<P><FONT>b</FONT>`
- Attributes are copied (except `id` and `name`)
- Restartability is recorded in the schema

Uncloseable Tags

- `FORM` and `TABLE` need special handling, and are marked as *uncloseable* in the schema
- If a form begins inside a table and extends past the end of it, we shouldn't force the form to close, as that will leave some UI elements orphaned
- Restarting the form doesn't work either, as that breaks the form into two forms
- Instead, TagSoup postpones the `</table>` till after the `</form>` is seen

Non-Nesting Tags

- All these features work together to ensure that improperly nested tags get repaired *with the correct semantics*
- `<B>`**`bold`**` <I>`***`bold italics`***` </B>`*`italics`*` </I>normal` becomes `<B>`**`bold`**` <I>`***`bold italics`***` </I></B><I>`*`italics`*` </I>normal`

Embedded Scripts and Stylesheets

- The `SCRIPT` and `STYLE` elements are supported by the schema
- Markup inside them is taken literally with the exception of the matching end-tag

Attribute Cleanup

- Attributes with no values are expanded: bare `compact` becomes `compact=”compact”`
- Attribute values with no whitespace do not require quotation marks
- TagSoup will get confused if an attribute value does contain whitespace but has no quotation marks

A Few Other Points

Roll Your Own

SAX2 properties let you:

- Specify your own scanner object (if your surface syntax is not HTML)
- Specify your own schema object (if your elements, content models, attributes, and entities are not HTML)
- Specify your own auto-detector object (if you know how to recognize encodings)

Schema Changes

- New elements, content models, attributes, and entities can be encoded
- Every element must have a content model and a parent element type
- Entities must resolve to a single character
- You can clone the standard HTML schema, or start from an empty schema and build up

How Big Is TagSoup?

- 7 classes: `Parser`, `Schema`, `HTMLSchema`, `HTMLScanner`, `Element`, `ElementType`, a private copy of `AttributesImpl`
- 4 interfaces: `Scanner`, `ScanHandler`, `AutoDetector`, `HTMLModels`
- 3 debug classes: `PYXScanner`, `PYXWriter`, a modified version of `XMLWriter`
- About 4600 lines of Java + 2000 lines of XSLT, Ant, RELAX NG, etc.
- JAR archive is about 48K

TagSoup Schema Language

- TSSL is a special-purpose schema language for describing HTML (or any other such language) to TagSoup
- A RELAX NG schema for TSSL is provided
- An XSLT script converts the TSSL into tables for the Java `HTMLSchema` and `HTMLModel` classes

State Table Markup Language

- STML is another special-purpose language for describing arbitrary state machines
- TagSoup's lexical analyzer is written in STML
- A RELAX NG schema for TSSL is provided
- An XSLT script converts the STML into tables for the Java `HTMLScanner` class

PYX Format

- PYX format is a linearized syntax for XML, almost the same as SGML ESIS format
- TagSoup provides support for PYX format on input and output
- Mostly for debugging, but may be useful for people using PYX format
- *http://www.xml.com/pub/a/2000/03/15/feature/*

Character Encodings

- Character encodings specified in an `InputSource` are believed
- By default, the platform default encoding is assumed (Latin-1 or Windows 1252, typically)
- You can plug in an `AutoDetector` class that knows how to guess the encoding of HTML to be parsed

Improvements

- Suggestions and patches are welcome
- Subscribe to `tagsoup-friends@yahoogroups.com`

More Information

`http://tagsoup.info`