

Re: SAX PARSING DESIGN PATTERN

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-03/msg03273.html>

- *From:* Lew <lew@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 28 Mar 2007 19:15:04 -0400
-

javerra wrote:

Hello all. Just looking for everyones thoughts on this.

I am parsing out an xml document using a sax parser. I create a new object representing my XML at the root in the StartElement [sic] method of my record handler and fill the object with the appropriate data at the EndElement [sic] method.

This works fine for for XML structures that contain 0 or 1 child elements. My question is for XML elements that can have children with more that one occurrence what would be the best way to design my objects. Is an inner class appropriate for this scenario? This has to be a fairly common thing encountered every day but all of the SAX examples I've see have only worked with very small generic XML documents.

Your thoughts are appreciated.

In the class that implements the parser element for a given tag I include a reference to the parent parser element object. So if <address> is a valid subtag for both <person> and <organization>, then the AddressHandler has an instance variable (inherited from AbstractHandler) called "encloser" or "parent" or similar that points, respectively, to the current enclosing PersonHandler or OrganizationHandler.

The parsing loop will retrieve a handler for the current tag during startElement() (typically using a Map<String, Class<AbstractHandler>>) and set its "parent" instance variable to the (erstwhile) current AbstractHandler before pointing currentHandler at the new one. At endElement() you emit whatever the element has, point the currentHandler to the parent object and let the (erstwhile) current handler become eligible for GC.

It's a linked list!

By the way, spelling counts. Or does your SAX parser actually have StartElement() and EndElement() methods? I don't think so. For one thing, if it did it would not be compatible with Sun's (or Apache's) parser interfaces, nor the Sun naming conventions.

-- Lew

.