GEOTOOLS OPEN SOURCE GML PARSING

Contact: David Zwiers
Refractions Research Inc.
Suite 400 – 1207 Douglas Street
Victoria, BC, V8W 2E7 Canada
E-mail: dzwiers@refractions.net
Phone: (250) 383-3022
Fax: (250) 383-2140

GOAL

The goal of this project is to provide a Web Feature Server (WFS) client library for the open source community. This client should be capable of reading and writing to an Open GIS Consortium (OGC) compliant WFS, including transaction and locking support.

PROBLEM

WFS responses consist of GML documents, which can be both very large, and very structurally complex. Building a parser that is both high performance (for large documents) and extremely flexible (for arbitrary user-defined GML documents) is a difficult balancing act.

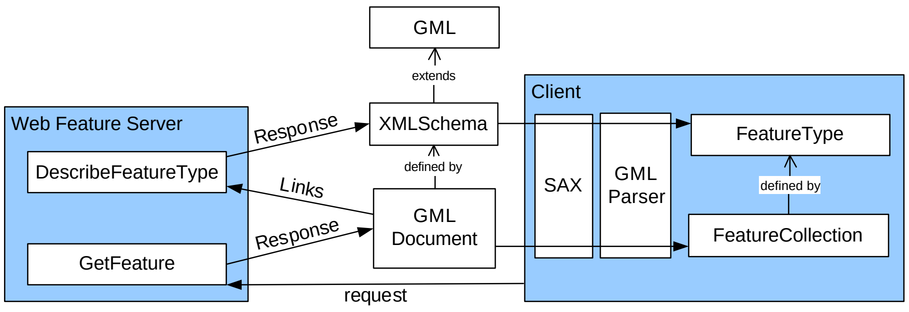


Figure 1: Web Feature Server Workflow

GML documents include links to "schema" documents, which provide a description of the GML document structure. The schema documents can themselves link to other schema documents, which describe sub-structures within the document.

Processing complex XML structures like GML is usually done with a "memory based" programming model, in which the entire document is parsed into memory structures, then post-processed into the final output objects. Processing large XML documents is usually done with an "event based" programming model, in which the parser is aware of the document structure in advance, and can build the final output objects on the fly as the document is read element by element.

The WFS client library needs a parser than can handle both cases at once – large and complex GML documents. The parser must be able to operate in real time, against any WFS server, against data sets of any size.

SOLUTION

The solution is an extremely general XML parser, capable of consuming any XML schema and document, but with hooks for previously "known" schemas, such as the GML schema and WFS schema. For unknown portions of the document, the parser can use a memory-based approach, while for known portions, the parser can use a high-performance event-based approach.

Parsing GML documents is really a special case of parsing an XML document where some types of elements (such as geometries) are handled in a special way. We use the XML inheritance structure as a guide to how each document element should be parsed.

PARSING THE SCHEMA

For example, when looking at the first Schema fragment below we can see the declared inheritance in the `park_Type` definition "`base=gml:AbstractFeatureType`". This inheritance information allows the parser to identify the park element in the XML example as a "`Feature`". When the parser processes the GML document containing the first XML fragment, the park elements can be parsed as Feature Objects.

Sample XML Fragment 1	Sample Schema Fragment 1
`<park>`	`<element name="park" type="myns:park_Type"`
`  <gml:boundedBy>`	`    substitutionGroup="gml:_Feature"/>`
`    <gml:Box srsName="EPSG:42304">`	`<complexType name="park_Type">`
`      <gml:coordinates>`	`  <complexContent>`
`        245524.015625,3585946.750000`	`    <extension base="gml:AbstractFeatureType">`
`        504494.156250,3830124.250000`	`      <sequence>`
`      </gml:coordinates>`	`        <element ref="gml:polygonProperty" minOccurs="0"/>`
`    </gml:Box>`	`        <element name="AREA" type="string"/>`
`  <gml:polygonProperty>`	`        …`
`  …`	`      </sequence>`
`  </ gml:polygonProperty>`	`    </extension>`
`  < AREA >40< AREA >`	`  </complexContent>`
`…`	`</complexType>`
`</park>`	

To understand the complete inheritance tree for a particular element, we need to parse the XML Schema. WFS users extend GML types, such as Feature, to include their own additional data. Our parser follows the XML schema inheritance tree until it finds elements it can map to a "well known" type.

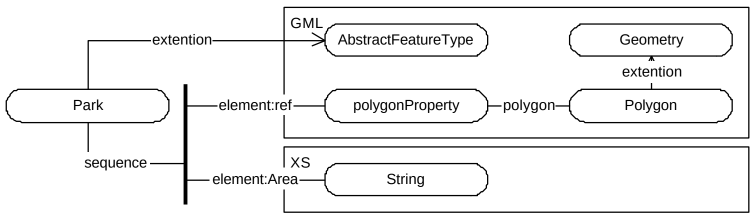


Figure 2: XML Schema Extension

XML schemas can include links to other schemas which define the structure of sub-components of the XML document. For example, in the fragments below, the opening tag for an XML document refers to two schemas: the WFS specification schema; and a user defined schema. The user defined schema in turn refers to the GML schema.

Sample XML Fragment 2	Sample Schema Fragment 2

<wfs:FeatureCollection
xsi:schemaLocation="http://www.opengis.net/wfs
wfs/1.0.0/WFS-basic.xsd http://www.ttt.org/myns
myns.xsd">
	<schema targetNamespace="http://www.ttt.org/myns"
	elementFormDefault="qualified" version="0.1">
	<import namespace="http://www.opengis.net/gml"
	schemaLocation="gml/2.1.1/feature.xsd"/>

PROCESSING THE DOCUMENT

Our solution to processing these nested structures quickly is to provide a pre-parsed version of "well known" schemas to the parser, which may include functionality to parse "well known" elements. This approach allows us to provide high performance functionality at any level of the XML inheritance tree.

When our parser processes an XML document, it follows the XML schema inheritance tree upwards until it finds a pre-defined processor capable of handling the data, or until it reaches the root of the inheritance tree. When the parser reaches the root of the inheritance tree without finding a pre-defined processor, the data is encoded using a default memory-bound algorithm.

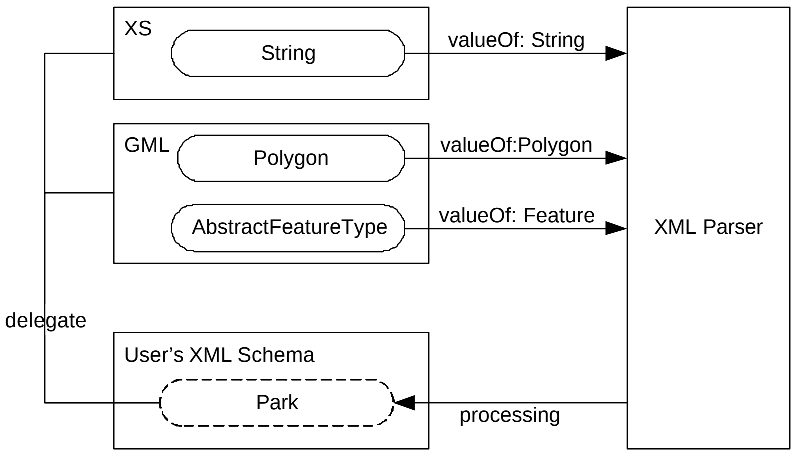


Figure 3: GML Parsing

CURRENT CAPABILITY

Currently we have processing extensions for GML 2.0, Xlink and XML Schema SimpleType Schemas. The next phase of this project add a WFS 1.0.0 Schema extension added to the list of "well known" structures. The GML processing extension avoids the use of in-memory collections, thereby allowing very large datasets to be processed on the fly.

In the future, other extensions can be easily added to include user defined schemas, or new standard schemas, providing additional validation and performance optimizations.

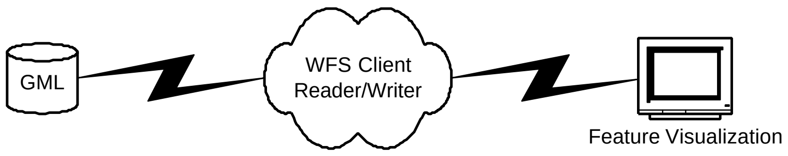
RELATED WORK

Another Open Source GML processing project, GML4J, also provides a GML 2.0 parser, based on JDOM, a memory dependant XML parser. GML4J was written and released by Galdos Systems Inc in 2002. GML4J is an interpreter for the resulting memory model of the JDOM parser, and as such may validate when JDOM validates, and is also capable of parsing GML fragments (GML without namespace declarations).

	GeoTools	**GML4J**
Scaleable	✓	
Customizable	✓	
Validating	✓	✓
Parses Fragments		✓

RESULTS

This project has already produced an Extensible XML Parser, with GML extensions. The GML extensions are capable of reading features as they are available without a large memory overhead, removing the need for all the features to reside in memory.



REFERENCES

uDig – http://udig.refractions.net
GeoTools – http://geotools.org
GML 2 Specification – http://www.opengis.org/docs/02-069.pdf
WFS Specification – http://www.opengis.org/docs/02-058.pdf
GML4J – http://gml4j.sourceforge.net/