

Python code-binding for VOEvent

Roy Williams
California Institute of Technology

VOEventLib is arranged as a package within a package. The outer level is the code binding technology used to build a parser from a documented XML schema, and the inner level is the actual parser, with a utility package, documentation, example code, and example events.

Parsing, Modifying, and Building VOEvents

VOEventLib is a reference implementation and parser for the VOEvent2 XML specification, now under discussion at the IVOA (Sept 2010). For more information about VOEvent, see VOEvent in wikipedia [1], or read the the VOEvent wiki at IVOA[2] or the draft specification of VOEvent 2.0 [3]. There is also the Skyalert web application [4] for publishing and dissemination of real-time astronomical events. First the parser itself (inner package). Use this command to get the software:

```
svn checkout https://ligo-vcs.phys.uwm.edu/svn/VOEventLib/lib
```

Once you have this directory tree, put the lib directory in your PYTHONPATH, then run and modify the example programs. The get and set methods of VOEventLib are all defined in the detailed documentation, which is too long to publish here, so we will discuss some of the example programs.

```
# parse the event from the file name
v = parse(filename)

# set the ivorn to something else
v.set_ivorn('ivo://sample/survey#123')

# set the author
v.get_who().get_Author().set_contactName(['Mickey Mouse'])

# look for a specific param in the event
param = findParam(v, '', 'EVENT_ID')
if param:
    print "old EVENT_ID is %s" % param.get_value()
# change the value of the param
    param.set_value(873646)

# output to standard output
v.export(sys.stdout, 0)
```

The event is parsed from a sample event in the given file, then the IVORN changed using the set method. The contact name for the event is set – note that the schema allows multiple contact names, and therefore the argument of `set_contactName` is a list. Then a specific parameter is found and changed, then the modified event output. Other methods allow construction and modification of `<Table>` data, a new feature of the VOEvent 2.0 specification.

Automatic Code Generation from XML Schema

The other package contains the engine used to gbuild the VOEvent parser directly from the XML schema, and may be of more general use than just VOEvent. It uses a pure python package called generateDS to make the python code, together with Sphinx to autogenerate package documentation from the documented schema.

The package generateDS generates Python data structures (for example, class definitions) from an XML Schema document. These data structures represent the elements in an XML document described by the XML Schema. It also generates parsers that load an XML document into those data structures, and

copies schema annotation into documentation of the python code (docstrings). This package was written by Dave Kuhlman. For more information see the generateDS web page.

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license. It uses the embedded documentation in the Python code (docutils) to build the HTML documentation. For more information see the Sphinx web page.

Use this command to get this code-generation part of the software, and the VOEvent implementation that is built with it:

```
svn checkout https://ligo-vcs.phys.uwm.edu/svn/VOEventLib
```

The schema generates the code automatically

```
<xs:complexType name="Param">
  <xs:annotation>
    <xs:documentation> what/Param definition. A Param has name, value, ucd, unit, dataType;
    and may have Description and Reference.</xs:documentation>
  </xs:annotation>
  <xs:choice maxOccurs="unbounded">
    <xs:element name="Description" type="xs:string"/>
    <xs:element name="Reference" type="Reference"/>
  </xs:choice>
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="ucd" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="unit" type="xs:string"/>
  <xs:attribute name="dataType" type="dataType" default="string"/>
</xs:complexType>
```

generates the methods and documentation shown below. Notice that the documentation element from the XML schema is correctly copied into the docstrings of the python code, and from there to the automated documentation of those methods.

```
class VOEvent.Param():
```

```
What/Param definition. A Param has name, value, ucd, unit, dataType; and may have Description and Reference.
```

```
add_Description(value)¶
add_Reference(value)¶
get_Description()¶
get_Reference()¶
get_dataType()¶
get_name()¶
get_ucd()¶
get_unit()¶
get_value()¶
insert_Description(index, value)¶
insert_Reference(index, value)¶
set_Description(Description)¶
set_Reference(Reference)¶
set_dataType(dataType)¶
set_name(name)¶
set_ucd(ucd)¶
set_unit(unit)¶
set_value(value)¶
```