XML, HTML and SQL

Human-Machine Dialogue

by

Bernhard Friedrichs, Metronix, Germany

bernhard.friedrichs@coopertools.com

Introduction

Data storage, handling and interchange has become a major problem in geophysics. ASCII files such as calibration data or the EDI format are still in use but they do not provide the capabilities of automatedo data storage and access. Binary formats such as NetCDF perform well in mass data storage but proprietary software is needed to access the data. The XML specification can solve most of these problems.

ASCII, Binary and XML Storage

The XML data format has become popular in the recent years. It solves the data storage and data access problem in an easy manner.

An simple example illustrates this:

```
>HEAD
  DATAID=079
  ACQBY=UNKNOWN
  LAT=..52:18:34
  LONG=  12:16:54
>=DEFINEMEAS
  MAXCHAN=7
  MAXRUN=999
  MAXMEAS=9999
```

The **ASCII** storage format uses cryptic separators to define keys and value. Without any knowledge one can not differentiate between main sections, subsections (>, >=, = ??).

In addition, ASCII files are read and written by “hand-made” parsers which lead to slight changes (spaces, tabulators, line feed) over the time and make the data file unreadable over long terms.

```
struct status {
      bool on;
      int gain;
};
----------------------------
struct status {
      bool on;
      double voltage;
      int gain;
};
```

The most efficient and compact format is the **binary** format.

The programmer is forced to use *exactly* the same structure as proposed. The binary format is used for mass data storage but the data description is generally incomplete. To change the data description the binary structure must be extended and the software must be re-written and re-compiled in order to read the new files.

This leads either to a minor acceptance or to a split of data description and data storage.

```
<caldata chopper="on">
  <c1 unit="Hz">1.00000000e-01</c1>
  <c2 unit="V/(nT*Hz)">2.05e-01</c2>
  <c3 unit="deg">8.85960000e+01</c3>
</caldata>
<caldata chopper="on">
  <c1 unit="Hz">1.25890000e-01</c1>
  <c2 unit="V/(nT*Hz)">2.027e-01</c2>
  <c3 unit="deg">8.82250000e+01</c3>
</caldata>
```

The **XML** format solves most of the problems mentioned above.

The `<key>value</key>` syntax solves the data storage as well as the data organization into sections and subsections.

A validating parser checks the consistency of the file automatically.

The outstanding feature of XML parsers is that the file can be extended any time (for example `<c4>value</c4>`) – the parser will always read and write the *complete* file – even though it may contain new sections which were not used before.

Therefore XML is used for extensible data storage where users or institutes can use the base functionality of XML files and software but can integrate their own specific information additionally.

Data-Mining

Instead of writing a manual, the XML specification allows the user to write a declaration of the data being used.

```
<DataDescription>
 ......
 <DataField name="c2"
   optype="continuous"
   dataType="double">
   <Unit unit="V/(nT*Hz)"/>
   <Unit unit="V/nT"/>
   <Unit unit="V"/>
   <Unit unit="re"/>
 </DataField>  ........
</DataDescription>
```

Inside the DataDescription section we can define

a) a valid data type (here double)

b) the option (here continuous for repeating entries)

c) and valid units / strings used for the attributes of the c2 field (e.g. V, re)

With this additional information the validity of the keys can be proofed and during data input the data can be transformed into valid units.

In this example a software developer will prepare his code for four different units in column c2.

Data Access

```
 $xml = simplexml_load_file($file);
}
 echo "cal  chopper on\n";
 foreach ($xml->caldata as $data) {
   if  ("on" == $data['chopper'] )
   echo
     $data->c1 . "  "
   . $data->c2 . "  "
   . $data->c3 . "\n";
   }
}
```

With the XML libraries – supplied for most of the programming languages – the data access becomes easy. To read the calibration data from the example above a few lines of code are only needed.

The data read and data write functionality is stable and complete and the data interchange becomes easy.

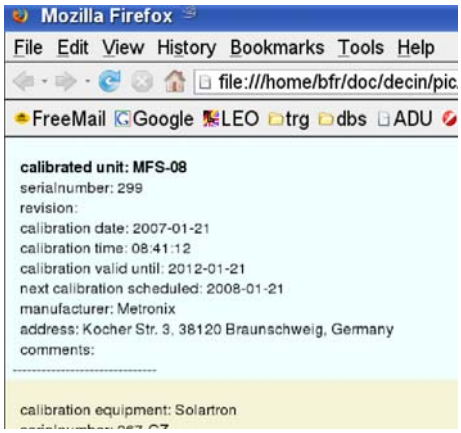
Data Storage

id	ci	ci_serial_number	ci_date	caldata
1	MFS-06	12	2007-08-08	<?xml version="1.0" encoding="utf-8"?> <?xml-style...
8	MFS-06	122	2007-09-10	<?xml version="1.0" encoding="utf-8"?> <?xml-styl...

Storing XML data on the file system leads to the problem of the remote access and, to a greater extend, of the indexing of these data files.

XML data can however be stored in SQL databases; a subset of information is used as an index (here the calibrated item, serial number and calibration date). Instead of scanning the XML data the SQL database selects the data by your pre-defined search criteria much faster. Hence that the column headers must have the same name as the keys inside the XML file.

Visualization



Together with a corresponding CSS (cascading style sheet) XML data can be visualized with standard browsers.

No proprietary software needs to be installed in this case.

In addition the CSS can be easily translated into other languages without changing the XML keys.

Conclusion

Even though XML standards are still not available in geophysics, more and more groups are in the developing phase.

Well designed XML data files with a few lines of data description can be easily accessed and the interchange is safe.

The complete specification of the Metronix calibration files can be found at www.metronix.de in the geophysics section.