ObsPy: A Python Toolbox for Seismology
A Webservice Perspective

Moritz Beyreuther & Robert Barsch

Department of Earth and Environmental Sciences (Geophysics)
Ludwig-Maximilians-Universität München

RapidSeis Liverpool
2009-11-12

ObsPy: A Python Toolbox for Seismology

Simplify Python programming for seismologists. Functionality include:

- obspy.gse2 - GSE2 read and write support (CM6)
- obspy.sac - SAC read and write support
- obspy.mseed - MiniSEED read and write support
- obspy.xseed - Converter from Dataless SEED to XML-SEED and RESP files
- obspy.seishub - SeisHub client
- obspy. arclink - ArcLink/WebDC request client
- obspy. fissures - DHI/Fissures request client (experimental)
- ...

Why Python?

- Free and Open Source
  - No licence for every process necessary
  - Available for third world countries
- Interpreter Language (steep learning curve)
  - Python shell for interactive learning
  - Easy debugging
  - Fast prototyping
- Direct access to existing shared libraries (C & Fortran)
- Large collection of open source scientific modules (matrix based)
- "Batteries included", large standard library

Key Features Ensuring Continuation of ObsPy

- Test-driven development (TDD), currently 177 unit tests
- Modular structure
- Reliance of well-known third-party tools (numpy, scipy, matplotlib)
- Reusing well established existing code, e.g. libmseed, GSE_UTI
- Platform independence (Win, Mac, Linux) and tested
- Free, open source (and available from the very beginning)
- Automatic generated API documentation
- Community webpage http://www.obspy.org containing: tutorials, installation instructions, complete source code, ...

Object Structure

- stream object consists of multiple trace objects
- trace = stream[i] object for one contiguous data block
- trace.data contains data as C style contiguous memory block, easy passing to C and Fortran libraries
- trace.stats contains meta information as dictionary
  - trace.stats.starttime contains starttime datetime object
  - trace.stats.sampling_rate contains the sampling rate
  - ...

ArcLink

ArcLink, a distributed data request protocol for accessing archived waveform data

```
from obspy.arclink import Client
from obspy.core import UTCDateTime

t = UTCDateTime("2009-08-24 00:20:03")
client = Client(host="webdc.eu", port=18001)

data = []
for station in ["RJOB","RNON","MANZ","ROTZ"]:
    data += client.getWaveform("BW",station,"","EHZ", t, t+30)
```

ArcLink, available Methods

Access station data

- client .getNetworks
- client .getPAZ
- client .saveResponse

Access waveform data

- client .getWaveform
- client .saveWaveform

SeisHub

SeisHub, a open source, modular, multi-component XML database with access to observational infrastructure (REST based)

```
from obspy.seishub import Client
from obspy.core import UTCDateTime

t = UTCDateTime("2009-08-24 00:20:03")
client = Client(host="http://teide.geophysik.uni-muenchen.de
    :8080")

data = []
for station in ["RJOB","RNON","MANZ","ROTZ"]:
    data += client.waveform.getWaveform("BW",station,"", "EHZ",
        t, t+30)
```

Event object

- client .event. getList
- client .event.getXMLResource

Station object

- client . station . getList
- client . station .getPAZ
- client . station .getXMLResource

Waveform object

- client .waveform.getChannelIds
- client .waveform.getNetworkIds
- client .waveform.getLatency
- client .waveform. getStationIds
- client .waveform.getLocationIds
- client .waveform.getWaveform

DHI/Fissures

DHI, a Data Handling Interface allowing users to access seismic data and meta data from IRIS DMC and other participating institutions (CORBA based)

```
from obspy.fissures import Client
from obspy.core import UTCDateTime

t = UTCDateTime("2009-08-24 00:20:03")
client = Client()

data = []
for station in ["BRNL", "PMG", "MORC", "DSB"]:
    data += client.getWaveform("GE",station,"", "EHZ", t, t+30)
```

DHI/Fissures, structure

Event CORBA object

- "/edu/iris/dmc","EventDC"

Network CORBA object

- "/edu/iris/dmc","NetworkDC"

Waveform CORBA object

- "/edu/iris/dmc","DataCenter"

XML-SEED

Introduced by Tsuboi, Tromp and Komatitsch (2004)

- Converter from:
  - Dataless SEED to XML-SEED and vice versa
  - Dataless SEED to RESP files
- Tested against complete:
  - ORFEUS Dataless SEED archive
  - IRIS (US) Dataless SEED archive
  - ArcLink requests

```
from obspy.xseed import Parser

sp = Parser()
sp.read("data/dataless/bw/dataless.seed.BW_MANZ")
sp.writeRESP(folder="BW_MANZ", zipped=False)
sp.writeXSEED("dataless.seed.BW_MANZ.xml")
```

XML-SEED

```
000001V 010009402.3121970,001,00:00:00.0000~2038,001,00:00:00.0000~
2009,037,04:32:41.0000~BayernNetz~~0110032002RJOB 000003RJOB 000008
...

<?xml version='1.0' encoding='utf-8'?>
<xseed version="1.0">
  <volume_index_control_header>
    <volume_identifier blockette="010">
      <version_of_format>2.4</version_of_format>
      <logical_record_length>12</logical_record_length>
      <beginning_time>1970-01-01T00:00:00</beginning_time>
      <end_time>2038-01-01T00:00:00</end_time>
      <volume_time>2009-02-06T04:32:41</volume_time>
      <originating_organization>BayernNetz</
          originating_organization>
...
```

Open Questions

Question we like to discuss/resolve in this workshop

- Structure for storing response information, XML?
- Structure for additional station information?
- Status QuakeML?