Fetching data from the database

- Add package (if necessary)
- Simple test
- Method detectors
- Method experiment_runs
- Method run_attributes
- Method calibration_runs
- Igor's script to find data type

This note is composed from Igor's e-mail from 2013-08-14 and further e-mail exchange.

Add package (if necessary)

```
  addpkg RegDB V00-01-16
  scons
```

Simple test

All methods can be tested in a single command:

```
 % python RegDB/src/experiment_info.py
```

Method `detectors`

From Python code:

```
  from RegDB import experiment_info
  detectors = experiment_info.detectors('XPP','xppa4513',1)
  print detectors

['BldEb-0|NoDevice-0',
 'EpicsArch-0|NoDevice-0',
 'NoDetector-0|Evr-0',
 'XppEndstation-0|Opal1000-2',
 'XppGon-0|Cspad-0',
 'XppSb4Pim-1|Tm6740-1']
```

Method `experiment_runs`

To get info which runs (and WHAT kind of runs) exist for the specified experiment do:

```
  runs = experiment_info.experiment_runs('XPP','xppa4513')
  print runs

[{'begin_time': 1375417636042155759L,
  'begin_time_unix': 1375417636,
  'end_time': 1375417646535192694L,
  'end_time_unix': 1375417646,
  'exper_id': 329L,
  'id': 69762L,
  'num': 1L,
  'type': 'DATA'},
 {'begin_time': 1375437784068608263L,
  'begin_time_unix': 1375437784,
  'end_time': 1375437816700510685L,
  'end_time_unix': 1375437816,
  'exper_id': 329L,
  'id': 69763L,
  'num': 2L,
  'type': 'DATA'},
  ..
{'begin_time': 1375806447772306081L,
  'begin_time_unix': 1375806447,
  'end_time': 1375806491980944918L,
  'end_time_unix': 1375806491,
  'exper_id': 329L,
  'id': 70074L,
  'num': 189L,
  'type': 'DATA'}]
```

The run number is given by key:

```
  'num': 189L
```

The last key of the dictionary:

```
  'type': 'DATA'
```

That is for normal data runs. A guess the idea is that people were able to change that to specify:

```
  'type': 'CALIB'
```

This is already supported by I don't think we have any experiment where they're using it.

Method `run_attributes`

To access tags associated with run use method **run_attributes(instr_name, exper_name, runnum)**:

```
instr_name = 'CXI'
exper_name = 'cxic0213'
runnum     = 215

for attr in experiment_info.run_attributes(instr_name, exper_name, runnum):
    print 'class:',attr['class']
    print 'name:',attr['name']
    print 'type (of the value):',attr['type']
    print 'value (optional):',attr['val']
    print 'description (optional):',attr['descr']
```


The value of an attribute is optional. If no value is set then you'll see None. The default description is the empty string

Method `calibration_runs`

Implemented since RegDB V00-01-18.

```
runs = calibration_runs(instr_name, exper_name)
for run in runs:
  print 'comment for this run:',run['comment']
  for calibtype in run['calibrations']:
    print calibtype
```

The function returns the dictionary of pairs:

```
{runnum:dict_recs}
```

where `dict_recs` is a dictionary of records:

```
{'comment':<string-of-comment>, 'calibrations':[<list-of-calibrations>]}
```

and `[<list-of-calibrations>]` consists of calibration types, for which this file can be applied.

For example:

```
{runnum:dict_recs}
{
  123  : {
    'calibrations' : ['dark', 'flat', ... ] ,
    'comment'      :   'Phillip marked this as DARK and FLAT filed run'
  } ,
  200 : {
    ..
  }
}
```

Igor's script to find data type

```
#!/usr/bin/env python
import sys
import MySQLdb as db
import RegDB.experiment_info as expinfo
if len(sys.argv) != 2:
    print "usage: <text-to-search>"
    sys.exit(1)
text2search = sys.argv[1]
conn   = db.Connect(host='psdb',db='regdb',user='regdb_reader')
cursor = conn.cursor()
cursor.execute('SELECT i.name AS `instr`, e.name as `exper` FROM instrument `i`, experiment `e` WHERE i.id=e.
instr_id')
for (instr,exper) in cursor.fetchall():
    runs = expinfo.experiment_runs(instr,exper)
    if len(runs):
        print "%s:%s (%d runs)" % (instr,exper,len(runs))
        first = runs[0] ['num']
        last  = runs[-1]['num']
        for runnum in range(first,last+1):
            try:
                for det in expinfo.detectors(instr,exper,runnum):
                    idx = det.find(text2search)
                    if idx != -1:
                        print "%s:%s:%d %s" % (instr,exper,runnum,det)
            except ValueError:
                pass
```