mootCore

What is mootCore? And what isn't it?

mootCore is a package containing the class MootQuery and some utility classes. MootQuery provides all the database queries to the MOOT (configuration and, someday,also calibration) metadata database needed by Online and Offline applications. Applications which add or modify database entries also need the services provided by another package, **MOOT**. Gleam only needs read access so only **mootCore** is of interest here. Supported queries include

```
/**
   Store a list of structures in @info argument concerning configs
   which satisfy cuts on status, instrument and mode.  They're sorted
   in ascending order by key.  Return value is count of Configs
   satisfying the cuts.
 */
```

```
unsigned getConfigInfo(std::vector<ConfigInfo>& info,
                       const std::string& status="CREATED",
                       const std::string& instr="LAT",
                       const std::string& mode="");
```

```
/** Given fmx logical id for a latc master, look up associated
    parameter files (latc source) and return a little structure
    of information about each.
 */
```

```
bool getParmsFromMaster(unsigned fmxMasterKey,
                        std::vector<ParmOffline>& parms);
```

and many more.

mootCore is not a Gaudi package (can't be since it's used by Online); it knows nothing about TDS, conversion services, etc.

OktoberTest

The immediate goal is to make the LATC GEM configuration available to Gleam in a manner somewhat similar to the way calibrations are currently handled - merely "somewhat" because the configuration information itself will not be stored in the TDS. Instead, a little struct will be stored in the TDS for each LATC source file, one member of which will be the file path. The typical client will be interested in multiple source files even if, from its point of view, it is interested in only "one thing". That's because LATC source must be partitioned in a way which satisfies both application and hardware constraints.

Status

Some time ago Martin wrote the code which knows how to parse the LATC source files used for GEM configuration.

mootCore could be used as is. Or it's possible another simple query or two will be needed so that clients will be able to get what they need without hard-coding things they should not be hard-coding.

The largely-missing piece is a Gaudi service which will store information about the LATC configuration files appropriate for the current event in the "calibration" TDS and update it transparently, just as is currently done for calibrations.

Miscellany

Since **mootCore** is used by Online as well as Offline, it needs to have a python interface. The simplest way to do this was to make use of **SwigPolicy**, as is done by various packages in ScienceTools. **SwigPolicy** in turn uses interface packages **python** and **swig**. So to incorporate **mootCore** into GlastRelease we'll need to add those interface packages as well and the Installer will have to know to download the corresponding external libraries.

While mucking around in **CalibSvc**, I added another abstract interface, ICalibPathSvc, which can be used to get the TDS path strings for calibrations. The old method, which depends on making data in the **CalibData** shareable public, has caused all kinds of grief on Windows. It's still available, but those who use it are urged to switch over to the new scheme. See examples of use in CalibSvc/src/test