NAME

DbException – Exception class for Db activity

SYNOPSIS

```
#include <db_cxx.h>

DbException::DbException(int err);
DbException::DbException(const char *description);
DbException::DbException(const char *prefix, int err);
DbException::DbException(const char *prefix1,
        const char *prefix2, int err);

const int
DbException::get_errno();

virtual const char *
DbException::what() const;
```

DESCRIPTION

The DB library is a family of classes that provides a modular programming interface to transactions and record-oriented file access. The library includes support for transactions, locking, logging and file page caching, as well as various indexed access methods. Many of the classes (e.g., the file page caching class) are useful independent of the other DB classes, although some classes are explicitly based on other classes (e.g., transactions and logging). For a general description of the DB package, see *db_intro*(3).

This manual page describes the DbException class and how it is used by the various Db* classes.

Most methods in the Db classes return an int but also throw an exception. This allows for two different error behaviors, that are known as *error models*. By default, the error model is configured to throw an exception whenever a serious error occurs. This generally allows for cleaner logic for transaction processing, as a try block can surround a single transaction. Alternatively, the error model can be set to not throw exceptions, and instead request the individual function to return an error code. The error model can be changed using the *DbEnv::set_error_model* method, see *DbEnv*(3).

A DbException object contains an informational string and an errno. The errno can be obtained using *DbException::get_errno*(). The informational string can be obtained using *DbException::what*().

We expect in the future that this class will inherit from the standard class exception, but certain language implementation bugs currently prevent this on some platforms.

Some methods may return non-zero values without issuing an exception. This occurs in situations that are not normally considered an error, but when some informational status is returned. For example, *Db::get* returns DB_KEYNOTFOUND when a requested key does not appear in the database.

SEE ALSO

db_archive(1), *db_checkpoint*(1), *db_deadlock*(1), *db_dump*(1), *db_load*(1), *db_recover*(1), *db_stat*(1), *db_intro*(3), *db_internal*(3), *db_thread*(3), *Db*(3), *Dbc*(3), *DbEnv*(3), *DbException*(3), *DbInfo*(3), *DbLock*(3), *DbLockTab*(3), *DbLog*(3), *DbLsn*(3), *DbMpool*(3), *DbMpoolFile*(3), *Dbt*(3), *DbTxn*(3), *DbTxnMgr*(3)