CJRLock

Inherits From: Object

Conforms To: CJRLock

Declared In: CJRLock.h

Requires: ThreadedApp

Class Description

A CJRLock is used to protect shared pieces of data or regions of code in a multi-threaded Application that uses the ThreadedApp class. It provides similar functionality to NeXT's NXLock (NS3.3) and NSLock (OpenStep), except that it detects if it being called from the main AppKit thread. If it is, it does not block waiting for a lock, but rather starts a modal event loop looking for EV_UNLOCK events (NX_APPDEFINED events). This keeps the AppKit thread alive and responsive to events.

CJRLocks are used in the same way as NXLocks and NSLocks. However, they cannot be used from fast callback methods (see ThreadedApp).

Instance Variables

None declared in this class.

Method Types

Acquire or release a lock - lock
- unlock

Instance Methods

lock

- **lock**

Waits until a lock isn't in use and acquires it using **mutex_lock()**. If called from the main AppKit thread, another strategy is used to acquire the lock to avoid blocking the main thread's event handling. First, **mutex_try_lock()** attempts to acquire the lock. If it fails, a modal event loop is started and waits for EV_UNLOCK events. When one is received, **mutex_try_lock()** attempts to acquire the lock again. This repeats until a lock is acquired. This behaves like a blocking lock, but method *callbacks* are permitted to continue (see ThreadedApp). Other events remain on the queue during the "block".

unlock

- **unlock**

Releases the lock, using **mutex_unlock()**. If the main thread is waiting for EV_UNLOCK events, post an EV_UNLOCK event to the application's event queue.