AlCThreading C++ Library

Author: alwyn.j.dippenaar@gmail.com

This project will produce the AlCThreading.dll library, which exposes the AlCThread class, which provides very similar functionality as the Thread class in java.

Table of Contents

Build:

This project will require Visual Studio 2017 to build.
This project will require all it's dependencies and their respective dependencies as well to build.

Dependencies:

This project depends on the following:

- AlCLogger
- std::mutex

Usage:

Users of this library must do the following:

- Include the <alcthreading.h> C++ header
- Link to the AlCThreading.lib C++ library.
- Derive from the AlCThread class.
  - Implement/override the following functions:
    - ***run***()

```
 //The main threadd class.
class DLLEXPORT AlCThread
{
public:
	AlCThread(AlCLogger *vLogger);
	virtual ~AlCThread();

	//Starts this thread.
	bool start();

	//THis will get invoked once the thread starts,
	//by a new thread, and not the current thread tha invoked start.
	virtual DWORD run();

	//Logging.
	AlCLogger *logger;

	//Thread id.
	DWORD threadId;
	HANDLE threadHandle;

	int stackSize;
	void setCompleted(bool newval);		// Thread safe way to set completed for thread.
	bool getCompleted();			// Thread safe way to retrieve completed for thread.

	void setStarted(bool newval);		// Thread safe way to set started for thread.
	bool getStarted();			// Thread safe way to retrieve started for thread.

	bool completed;

	//Character buffer for output.
	wchar_t tbuffer[2048];

	std::mutex threadMutex;			// Mutex to protect this thread.


	//Bool flags, for thread control.
	bool started;


};
```

Example:

Header:

```
/**
	This class represents a thread that will poll for videos to analyze.
**/
class PollerThread : public AlCThread
{
public:
	PollerThread(AlCLogger *vLogger, char *vpath);  // Constructor.
	virtual ~PollerThread();                                        // Destructor.

	char path[2048];                                                        // The input path fo poll.
	char pBuffer[2048];                                                        // The buffer for logging.

	virtual DWORD run();                                                // //The run Impl, is responsible for polling and processing videos.

};
```

Cpp:

```
/**
	The run Impl, is responsible for polling and processing videos.
**/
DWORD PollerThread::run()
{
	sprintf_s(pBuffer, "PollerThread::run:-- STARTED , path: '%s' \n\0", path);
	logger->debug(pBuffer);

 …
}
```