Implementing a Singleton using Symbian CCoeStatic class

This article shows how to create a singleton using `CCoeStatic` that can be used in Symbian C++ application classes (only).

Overview

The singleton pattern is one of the best-known patterns in software engineering. Essentially, a singleton is a class which only allows a single instance of itself to be created, and usually gives simple access to that instance.

This article explains how to create a singleton that can be used in classes which use the `CCoeEnv` class. Since `CCoeEnv` is a part of the UI control framework, this approach is only available to applications (not their engines). The approach is simpler than using TLS.

Source code

CMySingleton.h

```
/**
* Example implementation of a singleton class by means of inheriting
* from CCoeStatic.
*/
class CMySingleton : public CCoeStatic
{
public: // constructors and destructor
/**
* Returns an instance of this class. When called for the first
* time, a new instance is created and returned. After that,
* calling InstanceL returns the same instance that was created
* earlier.
*
* @return A pointer to a CMySingleton object
*/
static CMySingleton* InstanceL();
private: // constructor
/**
* Default constructor is private because we are using the
* singleton design pattern.
*/
CMySingleton();
...
};
```

CMySingleton.cpp

```
static const TUid KUidMySingleton = { 0x100000E9 };

// ---------------------------------------------------------------------------
// CMySingleton::CMySingleton
// C++ default constructor. It is private because we are using the
// singleton design pattern.
// ---------------------------------------------------------------------------
CMySingleton::CMySingleton()
: CCoeStatic( KUidMySingleton )
```

```
{
}
// ---------------------------------------------------------------------------
// CMySingleton::InstanceL
// Returns an instance of this class. When called for the first time,
// a new instance is created and returned. After that, calling
// InstanceL returns the same instance that was created earlier.
// Note that the UID passed to CCoeEnv::Static needs to be unique.
// ---------------------------------------------------------------------------
CMySingleton* CMySingleton::InstanceL()
{
CMySingleton* instance = static_cast<CMySingleton*>
( CCoeEnv::Static( KUidMySingleton ) );
if ( !instance )
{
instance = new ( ELeave ) CMySingleton;
CleanupStack::PushL( instance );
instance->ConstructL();
CleanupStack::Pop();
}
return instance;
}
```