Archived:Storing application settings in Symbian C++

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Overview

The following example shows how to handle application settings. Settings are stored into the private folder of the application.

MMP file

The following capabilities and libraries are required:

```
CAPABILITY NONE
LIBRARY efsrv.lib
LIBRARY  estor.lib
```

Header file, CSettingsData.h

Settings data class for storing/reading settings data from the stream.

```
#include <e32base.h>
class CSettingsData : public CBase
{
public:
    static CSettingsData* NewL();
    static CSettingsData* NewLC();
    virtual ~CSettingsData();

    // Loads itself from stream.
    void LoadL(RReadStream& aStream);

    // Saves itself to stream.
    void SaveL(RWriteStream& aStream) const;

    // Using default values (e.g. in first time)
    void SetDefaultValues();

    inline TBuf<100>& Text() { return iText; }
    inline TInt& Enumeration() { return iEnumeration; }

private:
    CSettingsData();
    void ConstructL();

public:
    TBuf<100> iText; //Data for text field
    TInt iEnumeration; // Data for enumerated text field
```

Source file, CSettingsData.cpp

```
CSettingsData *CSettingsData::NewL()
    {
    CSettingsData *self = CSettingsData::NewLC();
    CleanupStack::Pop(self);
    return self;
    }

CSettingsData *CSettingsData::NewLC()
    {
    CSettingsData *self = new (ELeave) CSettingsData();
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
    }

CSettingsData::~CSettingsData()
    {
    }

CSettingsData::CSettingsData()
    {
    }

void CSettingsData::SetDefaultValues()
    {
    // TODO: Set default values of your setting
    iEnumeration = 1000;
    }

void CSettingsData::ConstructL()
    {
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
// Reading setting data from stream
void CSettingsData::LoadL(RReadStream& aStream)
    {
    aStream >> iText;
    iEnumeration = aStream.ReadInt32L();
    }

// Storing setting data into stream
void CSettingsData::SaveL(RWriteStream& aStream) const
    {
    aStream << iText;
    aStream.WriteInt32L(iEnumeration);
    }
```

Header file, CMyAppUi.h

Application class that reads settings on construction and stores them on exit.

```
#include <aknviewappui.h>
#include "CSettingsData.h"

class CMyAppUi : public CAknViewAppUi
    {
public:
    void ConstructL();
    CSettingExampleAppUi();
    ~CSettingExampleAppUi();

public: // from CEikAppUi
    void HandleCommandL(TInt aCommand);

private:
    void InternalizeSettingsDataL();
    void ExternalizeSettingsDataL() const;

private:
    CSettingsData* iData;   // Pointer to settings data
    TFileName iSettingsFile;
    };
```

Source file, CMyAppUi.cpp

```
void CMyAppUi::ConstructL()
    {
    BaseConstructL(EAknEnableSkin);

    // Create private path
    RFs& fsSession = iEikonEnv->FsSession();
    User::LeaveIfError(fsSession.CreatePrivatePath( EDriveC ) );
    User::LeaveIfError(fsSession.PrivatePath(iSettingsFile));
    iSettingsFile += KSettingsFile;

    // Construct the data object the settings list will use
    iData = CSettingsData::NewL();
    // Read settings from stream
    InternalizeSettingsDataL();

    // TODO: Construct and activate the app view as well
    }

// Handle menu commands
void CMyAppUi::HandleCommandL(TInt aCommand)
    {
    switch(aCommand)
        {
        case EEikCmdExit:
        case EAknSoftkeyExit:
            {
            // Store setting data on application exit
            ExternalizeSettingsDataL();
            Exit();
            break;
            }
        default:
            break;
        }
    }

// Load settings
void CMyAppUi::InternalizeSettingsDataL()
    {
    RFs& fs = iEikonEnv->FsSession();
    RFileReadStream readStream;
    TInt error = readStream.Open(fs, iSettingsFile, EFileRead);
    TInt internalizationError = KErrNone;
    // If file existed, try to read settings.
    if (error == KErrNone)
        {
        TRAP(internalizationError, iData->LoadL(readStream);)
        }
    else
        {
        // Use default values in first time when no
        // setting file exists
        iData->SetDefaultValues();
        }
    readStream.Release();

    // Reading failed, settings file might be corrupted.
    if (internalizationError != KErrNone)
        {
        User::LeaveIfError(fs.Delete(iSettingsFile));
        }
    }

// Save settings
void CMyAppUi::ExternalizeSettingsDataL() const
    {
    RFs& fs = iEikonEnv->FsSession();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    RFileWriteStream writeStream;
    TInt error = writeStream.Open(fs, iSettingsFile, EFileWrite);
    // Setting file did not exist, create one.
    if (error != KErrNone)
        {
        User::LeaveIfError(writeStream.Create(fs, iSettingsFile, EFileWrite));
        }
    writeStream.PushL();
    iData->SaveL(writeStream);
    writeStream.CommitL();
    writeStream.Pop();
    writeStream.Release();
    }
```

Streaming

Some examples of data streaming (reading):

```
    // String
    TBuf<100> iText;
    aStream >> iText;

    // Integer
    TInt iVolume;
    iVolume = aStream.ReadInt32L();

    // Boolean
    TBool iBinary;
    iBinary = aStream.ReadInt16L();

    // Ip address
    TInetAddr iIpAddress;
    iIpAddress.SetAddress(aStream.ReadUint32L());

    // Time
    TTime iTime;
    TInt64 time;
    aStream >> time;
    iTime = TTime(time);
```

Postconditions

Application loads settings on startup and stores them on exit.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.