Archived:Using CTextResolver to resolve error texts

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

This code snippet shows how the class `CTextResolver` is used to resolve the corresponding error texts for error codes. The class `CTextResolver` has two versions of the method `ResolveErrorString()`: the first one is for normal use and the second one is for advanced use. Both versions of the method are used in this example. It is also possible to give an optional error context to map error codes to texts in a unique way. If no context is given (aContext = ECtxAutomatic), the assumption is that the error codes are unique.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY commonui.lib
```

Header file

```
#ifndef __TESTERAPPUI_H__
#define __TESTERAPPUI_H__

// INCLUDES
#include <aknappui.h>

// FORWARD DECLARATIONS
class CTextResolver;


class CTesterAppView : public CCoeControl
    {
private:
  // Functions from base classes
  void HandleCommandL(TInt aCommand);
//...
  void DoHandleCommandL(TInt aCommand);
 //...
private:
    CTextResolver* iTextResolver;
    };

#endif // __TESTERAPPUI_H__
```

Source file

(C) Copyright Nokia 2014. All rights reserved.

```
#include <textresolver.h>

void CTesterAppUi::ConstructL()
  {
  //...
   iTextResolver = CTextResolver::NewL(*iCoeEnv);
  }

CTesterAppUi::~CTesterAppUi()
  {
  //...
  delete iTextResolver;
  }

void CTesterAppUi::HandleCommandL(TInt aCommand)
  {
    switch (aCommand)
      {
      case EEikCmdExit:
      case EAknSoftkeyExit:
        Exit();
        break;

      case ECommand1:
        {
        TPtrC errorTextPtr; // pointer to returned error text

        TRAPD( errSimple, DoHandleCommandL( aCommand ) );
        if ( errSimple != KErrNone )
          {
          // resolve the given error code
          errorTextPtr.Set(iTextResolver->ResolveErrorString(errSimple));
          }
        else
          {
          // KErrNone, do something....
          }
        }
        break;

      case ECommand2:
        {
        TInt textId =0;     // ID of the returned text
        TUint flags = 0;    // priority of the returned error
        TPtrC errorTextPtr; // pointer to returned error text

        TRAPD( errAdvanced, DoHandleCommandL( aCommand ) );
        if ( errAdvanced != KErrNone)
          {
          // resolve the given error code
          errorTextPtr.Set(iTextResolver->ResolveErrorString(errAdvanced, textId,
flags));

            //the flag indicates that the error has no proper explanation
          if (flags & EErrorResBlankErrorFlag)
            {
            // do something...
```


(C) Copyright Nokia 2014. All rights reserved.

```
            }

          //the flag indicates that Text Resolver does not support the error code
          if (flags & EErrorResUnknownErrorFlag)
            {
            // do something...
            }

          //the flag is returned while processing KErrNoMemory error code
          if (flags & EErrorResOOMFlag)
            {
            // do something...
            }
          }
        else
          {
          // KErrNone do something...
          }
        }
        break;

    default:
      Panic(ETesterUi);
      break;
    }
  }

void CTesterAppUi::DoHandleCommandL(TInt aCommand)
  {
  //Do Something that might leave...
  User::Leave(KErrNotSupported );
  }
```

Postconditions

The class `CTextResolver` resolves the error text when the method called by the user leaves.

See also

Archived:Using class CErrorUI to display error notes

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.