Dave's Development Blog
Software Development using Borland / Codegear / Embarcadero RAD Studio

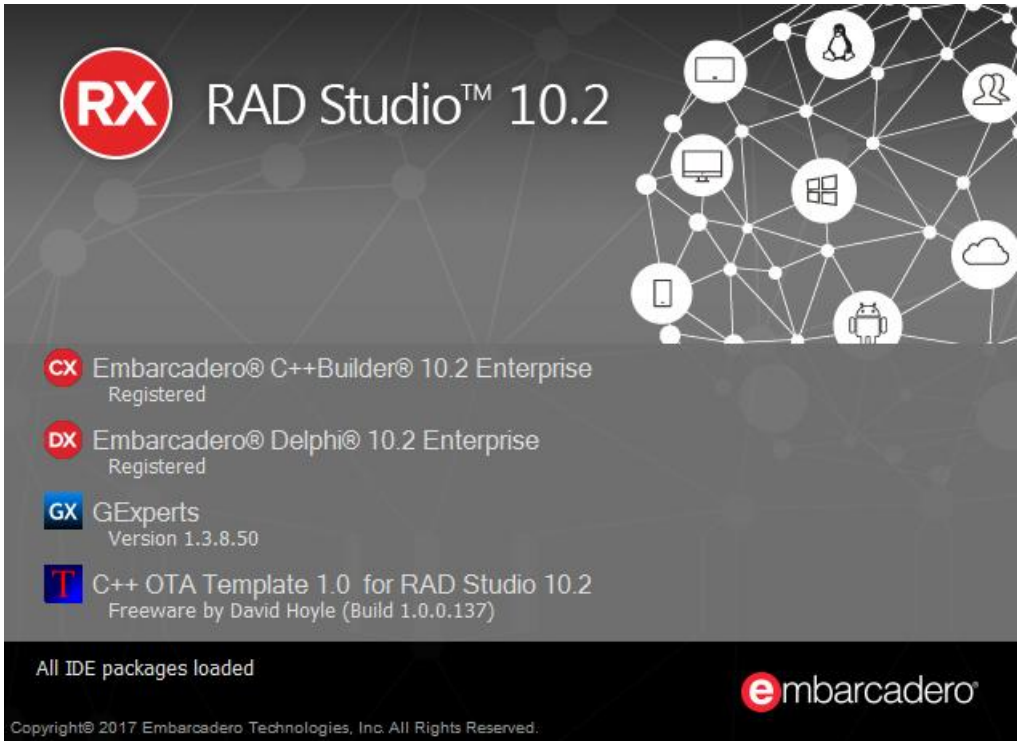
The Open Tools API using C++ Builder – A Fix

By David | July 9, 2017 0 Commen

Overview

A while ago I wrote an article (The Open Tools API using C++ Builder) on building an Open Tools API (OTA) plug-in using C++ Builder. In the article I mentioned that there was a bug that prevented a DLL from directly referencing the `DesignIDE.bpl` global variables `BorlandIDEServices` and `SplashScreenServices`.

I raises this as a bug to Embarcadero (https://quality.embarcadero.com/browse/RSP-16481) that prevented a splash screen from being used in a DLL and David Millington has responded with a workaround which I'll describe here and which works for eariler IDEs as well.



The fix

The fix require the addition of a line of code in the project CPP file and amendments to other files so I'll describe the changes to each file in the following subsections:

CPPOTATemplateXE102.cpp

There is a single change to the main project file where we add an additional line to link in the `DesignIDE.bpi` package file. This is the workaround provided by Embarcadero while they seek to fix the Tokyo compiler but this fix works with earlier compilers as well.

```
#include <vcl .h>
#include <windows .h>

#pragma hdrstop
#pragma argsused
```

```
//: @note This pragma line fixes the missing external references to the DesignIDE.BPL package.
#pragma link "DesignIDE.bpi"

int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void* lpReserved)
{
 return 1;
}
```

CPPOTATemplateMacros.h

Now we've bound the `DesignIDE.bpi` file to the DLL we no longer need the macro that binds the local IDE services reference to the `BorlandIDEServices` passed to the wizard initialization method.

```
#ifndef CPPOTATemplateMacrosH
#define CPPOTATemplateMacrosH

#include <toolsapi .hpp>

//: @note Not required any more
//: #ifdef DLL
//: #define BorlandIDEServices LocalIDEServices
//: extern _di_IBorlandIDEServices LocalIDEServices;
//: #endif

#define QUERY_INTERFACE(T, iid, obj)   \
  if ((iid) == __uuidof(T)) {          \
    *(obj) = static_cast(this);    \
    static_cast(*(obj))->AddRef(); \
    return S_OK;                       \
  }

#endif
```

CPPOTATemplatePkgDLLInit.cpp

In the main DLL code we can remove the variable declaration and assignment code which assigned the `BorlandIDEServices` reference pass to the wizard initialization function as they are not needed any more.

```
#pragma hdrstop

#include <cppotatemplatepkgdllinit .h>
#include <cppotatemplatemainwizard .h>
#pragma package(smart_init)

#ifndef DLL
// For Packages...
// We need to declare for a package a Register procedure.
// The NAMESPACE MUST BE the same name as unit Register is declared in and be lower case except
// for first letter.
namespace Cppotatemplatepkgdllinit {
  void __fastcall PACKAGE Register() {
    RegisterPackageWizard(new TCPPOTATemplateWizard("TCPPOTATemplateWizard"));
  }
}
#else
// For DLLs...
// We need to declare a local variable to accept the BorlandIDEServices reference from the
// Wizard creation method below
//: @note Not Required
//: _di_IBorlandIDEServices LocalIDEServices;
```

```
// We also need to delcare the wizard entry point that is called by the IDE on loading a DLL
extern "C" bool __stdcall __declspec(dllexport) INITWIZARD0001(
  const _di_IBorlandIDEServices Service,
  TWizardRegisterProc RegisterWizard,
  TWizardTerminateProc&)
{
  //: @note Not Required
  //: LocalIDEServices = Service; // get reference to the BorlandIDEServices
  RegisterWizard(new TCPPOTATemplateWizard("TCPPOTATemplateWizard"));
  return true;
}
#endif
```

CPPOTATemplateSplashScreen.h

Next we need to remove or comment out the `#ifndef` that surrounds the declaration of the function that installs the splash screen.

```
#ifndef CPPOTATemplateSplashScreenH
#define CPPOTATemplateSplashScreenH
#endif

//: @note Not required
//: #ifndef DLL
void __fastcall AddSplashScreen();
//: #endif
```

CPPOTATemplateSplashScreen.cpp

Next we need to remove or comment out the `#ifndef` that surrounds the function that installs the splash screen.

```
#pragma hdrstop
#include "CPPOTATemplateSplashScreen.h"
#include "windows.h";
#include "CPPOTATemplateConstants.h"

#include "SysInit.hpp"
#include <toolsapi .hpp>
#include "SysUtils.hpp"
#include "Forms.hpp"
#include "CPPOTATemplateFunctions.h"

#pragma package(smart_init)

//: @note IFNDEF not required anymore
//: #ifndef DLL
void __fastcall AddSplashScreen() {
  int iMajor;
  int iMinor;
  int iBugFix;
  int iBuild;
  HBITMAP bmSplashScreen;
  BuildNumber(iMajor, iMinor, iBugFix, iBuild);
  bmSplashScreen = LoadBitmap(HInstance, L"CPPOTATemplateSplashScreenBitMap24x24");
  _di_IOTASplashScreenServices SSServices;
  if (SplashScreenServices->Supports(SSServices)) {
    String strRev = strRevision[iBugFix];
    SSServices->AddPluginBitmap(
      Format(strSplashScreenName, ARRAYOFCONST((iMajor, iMinor, strRev, Application->Title))),
      bmSplashScreen,
      False,
      Format(strSplashScreenBuild, ARRAYOFCONST((iMajor, iMinor, iBugFix, iBuild)))
    );
```

```
    Sleep(5000); //: @debug Here to pause splash screen to check icon
  }
}
//: #endif
```

CPPOTATemplateMainWizard.cpp

The final change to make is to comment out or remove the `#ifndef` that surrounds the call to install the splash screen which is contained within the main wizard class's constructor.

```
__fastcall TCPPOTATemplateWizard::TCPPOTATemplateWizard(String strObjectName) :
  TDGHNotifierObject(strObjectName) {
  //: @note Not required now
  //: #ifndef DLL
  AddSplashScreen();
  //: #endif
  FAboutBoxPlugin = AddAboutBoxPlugin();
  FIDENotifier = AddIDENotifier();
  FAppOptions = new TCPPOTATemplateOptions();
  FIDEOptions = AddOptionsFrameToIDE(FAppOptions);
  FTimerCounter = 0;
  FAutoSaveTimer = new TTimer(NULL);
  FAutoSaveTimer->Interval = 1000;
  FAutoSaveTimer->OnTimer = AutoSaveTimerEvent;
  FAutoSaveTimer->Enabled = true;
}
```

Hopefully the above changes are straight forward and allow you to either update my existing code (although you can download the changes from the C++ OTA Template page) or update your own code.

Related posts:

1. The Open Tools API using C++ Builder (15.9)
2. DUnit Testing in C++ Builder (10.6)
3. Conditional Compilation of Open Tools API experts (6.7)
4. Finding Open Tools API Interfaces (6.2)
5. Chapter 9: Aboutbox Plugins and Splash Screens (6.1)

Category: C++ Builder C++ OTA Template Tags: IOTASplashScreenServices, _di_IOTASplashScreenServices