Acrobat SDK CodeSnippet Architecture

Adobe Developer Support

Introduction

The CodeSnippet plug-in delivered as part of the Acrobat SDK is designed to allow developers to quickly and easily prototype Acrobat API calls. It provides infrastructure that supports these code snippets, and will eventually be used to provide a bridge between working code samples and core reference documentation.

Rationale

Adobe FrameMaker is an excellent long document processing application, however it is severely limited as an IDE. Embedding code within FrameMaker documents makes it hard to test and verify, impossible to maintain.

Acrobat Application plug-ins are relatively heavy weight. Each has a set of project files (one for each platform), plus other “baggage” (such as an “init” file). The smallest meaningful plug-in has 4 associated files (not including main.c or winproject.dsw). If a plug-in only details three or four APIs, it is an expensive way to communicate this to developers. To highlight this, consider the amount of work required to convert a project from one version of the application to another, or from one version of the IDE to another.

So on the one hand we have the ability to add code to documentation, cheap to do, expensive to verify and expensive to maintain. On the other hand we can add plug-ins to the SDK, expensive to do, cheap to verify and expensive to maintain. We need a solution that allows us to amortise the cost of writing a plug-in across many sets of API calls (demonstrations). The cost of maintaining such a solution should focus on the changes to the evolving Acrobat application, not the IDE. It should also be cheap to verify both compilation and execution correctness.

The solution is Code Snippets. These are sub-elements of a traditional Acrobat plug-in that are supported within a rich environment. The primary goal is to make a snippet as cheap to publish as possible.

The SnippetRunner Plug-in

The SnippetRunner plug-in is a project (dsp or mcp) that houses and provides infrastructure to snippets. Each snippet is free to determine what context should exist before it is called. For example, a snippet that reports the type of the current selection does not need to open a document and set a selection before reporting its type, it can assume the user has performed these tasks prior to invoking the snippet (the snippet should still check that a document has been opened, however).

The infrastructure that the SnippetRunner project provides includes:

- set of utility functions to test the state of the application (such as having a document opened) TBD
- a back stop exception handler that reports the name of the snippet that caused an exception - snippet writers do not need to consider exception handling in their entry routine. Some snippets might execute beyond the scope of the function invoked from the menu, such as callbacks or notifications. Whenever there is an as an asynchronous call back into a snippet, the function being called should handle exceptions in the usual manner.
- A macro definition that binds the snippet to the environment. This macro call details the function to call and a colon delimitered string that defines where the menu item will be placed in the UI.
- A common parameter input mechanism (a dialog and supporting utility code to convert from string to common types) that allows the snippets to be parameterised. TBD
- Automatic binding to the snippet documentation set.

Writing a Code Snippet

The simplest code snippet is a single method with the associated macro call. The code snippet should be places within a separate file, and added to the SnippetRunner project.

Ironically, I am including an example of a code snippet below:

```
void RaiseExcepSnip(void){
        ASInt32 errorCode = ASRegisterErrorString (ErrAlways,"A sample error");
        ASRaise(errorCode);
}
SNIPRUN_REGISTER(RaiseExcepSnip, "AS Layer:Exception:Raise custom")
```

I have left the #includes out for brevity, however although this snippet executes only two API calls, the work involved in creating a snippet is minimal.

Snippet Interaction

To execute a snippet, just select its menu item (so for the example above, from the “AcroSDK Snippet” menu on the toolbar, select the “AS Layer” sub-menu, then the “Exception” sub-menu before finally getting to the “Raise custom” menu-item). To see its associated documentation, ctrl-click on the menu item.