Subject: What do you think about this approach to making CodeEditor more user extendable?
Posted by cbpporter on Tue, 15 Dec 2015 11:07:19 GMT
View Forum Message <> Reply to Message

Hi Mirek,

In my project I require a powerful code editor and I'm using CodeEditor. While it is a very powerful class, it is not that easy to add features to it without forking it. And this is what I've done.

But I am thinking about how to make it much more extensible.

Currently, there is an enum describing a bunch of hard-coded languages and a few tables with hard-coded positions.

I'm thinking of an approach where CodeEditor has no hard coded support for languages. Instead of a single value describing the whole language, we replace it with a structure having options and an old language option acts as a particular collection of new boolean options.

I'll give two examples on how this could work.

First, a statement like:
if(highlight == HIGHLIGHT_CSS ? *p == '#' : pair == MAKELONG('0', 'x') || pair == MAKELONG('0', 'X'))

would become:
if(highlight.UseCSSHex ? *p == '#' : pair == MAKELONG('0', 'x') || pair == MAKELONG('0', 'X'))

We would register the CSS syntax with this option set to true and for other registered syntaxes to false.

Then in CSyntax I added support for #region tags, like in C#. Rather than checking if it is C# or other languages that support #region, one would check highlight.UseRegions. This option would be checked by EditorBar too in order to draw #region outlines, like for #ifdef. I implemented this using a new hardcoded language, but would like a more flexible solution.

One would register a structure with appropriate options set for a language, together with keywords and upp ids.

How does this approach sound? I already have a forked CodeEditor, so I can create an early minimal prototype if you wish.