Subject: CodeEditor with user-definable syntax highlighting
Posted by dolik.rce on Sun, 24 Jun 2012 08:01:00 GMT
View Forum Message <> Reply to Message

Few days ago, I thought that it might be fun to reimplement CodeEditor to support user-defined highlighting as an excersise to avoid my brain slowly rotting from all that Java code I have to write at work I even started working on it in my free time, and here is a little tasting of the work done so far:

The image shows CodeEditor using the highlight definition file specifying highlighting of higlighting definition files to highlight highlight definition file specifying highlighting of higlighting definition files. Pretty meta, right?

It is based on pcre and caching. Basic highlighting works just fine, some of the higher level features (scope highlighting, matching brackets, ...) are not yet implemented and some are using the old CodeEditor code. The goal is to allow loading user specified highlighters, both at compile time (by include) and at run-time. It requires some pcre skill (or copy&paste skill) to write one, but otherwise it is very simple format as you can see above - it takes only 16 lines to describe C++ This file imitates current CodeEditor:Quote:HL_PATTERN(INK_OPERATOR,
"[\\Q!+-*/\%&|=[]:?<>.~^\\E]")
HL_PATTERN(INK_KEYWORD, " \\b(__(asm|cdecl|declspec|except|fastcall|finally|inline|int
(16|32|64|8)|leave|stdcall|try|uuidof)|auto|bool|break|case|
catch|char|class|const|const_cast|continue|default|delete|dl
l(ex|im)port|do|double|dynamic_cast|else|enum|explicit|exter
n|false|float|for|force_inline|friend|goto|if|inline|int|lon
g|mutable|namespace|new|operator|private|protected|public|re
gister|reinterpret_cast|return|short|signed|sizeof|static|st
atic_cast|struct|switch|template|this|thread|throw|true|try|
typedef|typeid|typename|union|unsigned|using|virtual|void|vo latile|while)\\b ")
HL_PATTERN(INK_UPPER, "\\b[A-Z][A-Z_0-9]*\\b")
HL_PATTERN(INK_UPP, "\\bupp\\b")
HL_PATTERN(INK_MACRO, " ^[[:space:]]*#[[:space:]]*(define|include(_next)?|(un|ifn?)d
ef|endif|el(if|se)|if|warning|error|pragma|using)\\b ")
HL_PATTERN(INK_UPPMACROS, "
\\b(CLASSNAME|(THIS|PTE)BACK([1-4]?)|QUOTE|X?ASSERT|X?NEVER|
X?CHECK|NAMESPACE_UPP|END_UPP_NAMESPACE|NEVER_|ASSERT_)\\b ")
HL_PATTERN(INK_UPPLOGS, "
\\b([DLR]?DUMP(C|CC|CC|HEX|M)?|[DLR]?LOG(F|HEX|BEGIN|END|BLO
CK|HEXDUMP|SRCPOS)?|[DLR]?TIMING|DEBUGCODE|RQUOTE)\\b ")
HL_PATTERN(INK_CONST_INT, "\\b[0-9]+\\b")
HL_PATTERN(INK_CONST_HEX, "\\b0(x|X)[[:xdigit:]]+\\b")
HL_PATTERN(INK_CONST_OCT, "\\b0[0-7]+\\b")
HL_PATTERN(INK_CONST_FLOAT, "\\b[0-9]+\\.[0-9]*\\b")
HL_PATTERN(INK_CONST_STRINGOP, "TODO")
HL_PATTERN(INK_CONST_STRING, "[\"](\\.|[^\"])*[\"]")
HL_PATTERN(INK_COMMENT, "//.*")
HL_PATTERN_ML(INK_COMMENT, "/*", "*/")
HL_SCOPE("[{]", "[}]")

Best regards,
Honza

PS: Reposted from this thread.
