Source Code Highlight Filter

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
8.6.5	20 May 2011		

Contents

The AsciiDoc distribution includes a source code syntax highlight filter (`source-highlight-filter.conf`).

1 HTML Outputs

The highlighter uses GNU source-highlight to highlight *html4* and *xhtml11* outputs. You also have the option of using the Pygments syntax highlighter for *xhtml11* outputs.

To use Pygments you need to define an AsciiDoc attribute named *pygments* (either from the command-line or in the global `asciidoc.conf` configuration file) and you will also need to have Pygments installed and the *pygmentize* command in your PATH.

- You can customize Pygments CSS styles by editing `./stylesheets/pygments.css`.

- To make Pygments your default highlighter put the following line your `~/.asciidoc/asciidoc.conf` file:

```
pygments=
```

- The AsciiDoc *encoding* attribute is passed to Pygments as a `-O` command-line option.

2 DocBook Outputs

DocBook outputs are highlighted by toolchains that have `programlisting` element highlight support, for example *dblatex*.

3 Examples

3.1 Source code paragraphs

The `source` paragraph style will highlight a paragraph of source code. These three code paragraphs:

```
[source,python]
if n < 0: print 'Hello World!'

:language: python

[source]
if n < 0: print 'Hello World!'

[source,ruby,numbered]
[true, false].cycle([0, 1, 2, 3, 4]) do |a, b|
    puts "#{a.inspect} => #{b.inspect}"
```

Render this highlighted source code:

```
if n < 0: print 'Hello World!'
```

```
if n < 0: print 'Hello World!'
```

```
1  [true, false].cycle([0, 1, 2, 3, 4]) do |a, b|
2      puts "#{a.inspect} => #{b.inspect}"
```

3.2   Unnumbered source code listing

This source-highlight filtered block:

```
[source,python]
----------------------------------------------------------------------
''' A multi-line
    comment.'''
def sub_word(mo):
    ''' Single line comment.'''
    word = mo.group('word')   # Inline comment
    if word in keywords[language]:
        return quote + word + quote
    else:
        return word
----------------------------------------------------------------------
```

Renders this highlighted source code:

```
''' A multi-line
    comment.'''
def sub_word(mo):
    ''' Single line comment.'''
    word = mo.group('word')      # Inline comment
    if word in keywords[language]:
        return quote + word + quote
    else:
        return word
```

3.3   Numbered source code listing with callouts

This source-highlight filtered block:

```
[source,ruby,numbered]
----------------------------------------------------------------------
#
# Useful Ruby base class extensions.
#

class Array

  # Execute a block passing it corresponding items in
  # +self+ and +other_array+.
  # If self has less items than other_array it is repeated.

  def cycle(other_array)  # :yields: item, other_item
    other_array.each_with_index do |item, index|
      yield(self[index % self.length], item)
    end
  end

end

if $0 == __FILE__                                     <1>
  # Array#cycle test
  # true => 0
  # false => 1
  # true => 2
  # false => 3
  # true => 4
```

```
    puts 'Array#cycle test'                                <2>
    [true, false].cycle([0, 1, 2, 3, 4]) do |a, b|
      puts "#{a.inspect} => #{b.inspect}"
    end
  end
  ----------------------------------------------------------------------

  <1> First callout.
  <2> Second callout.
```

Renders this highlighted source code:

```
1   #
2   # Useful Ruby base class extensions.
3   #
4
5   class Array
6
7     # Execute a block passing it corresponding items in
8     # +self+ and +other_array+.
9     # If self has less items than other_array it is repeated.
10
11    def cycle(other_array)  # :yields: item, other_item
12      other_array.each_with_index do |item, index|
13        yield(self[index % self.length], item)
14      end
15    end
16
17  end
18
19  if $0 == __FILE__                                         ❶
20    # Array#cycle test
21    # true => 0
22    # false => 1
23    # true => 2
24    # false => 3
25    # true => 4
26    puts 'Array#cycle test'                                ❷
27    [true, false].cycle([0, 1, 2, 3, 4]) do |a, b|
28      puts "#{a.inspect} => #{b.inspect}"
29    end
30  end
```

❶   First callout.

❷   Second callout.

Tip

- If the source *language* attribute has been set (using an *AttributeEntry* or from the command-line) you don't have to specify it in each source code block.

- You may need to place callout markers inside source code comments to ensure they are not misinterpreted and mangled by the highlighter.

---

4 Installation

4.1 HTML

If you want to syntax highlight AsciiDoc HTML outputs (`html4` and `xhtml11` backends) you need to install GNU source-highlight or Pygments (most distributions have these packages).

4.2 DocBook

AsciiDoc encloses the source code in a DocBook *programlisting* element and leaves source code highlighting to the DocBook toolchain (dblatex has a particularly nice programlisting highlighter). The DocBook programlisting element is assigned two attributes:

1. The *language* attribute is set to the AsciiDoc *language* attribute.

2. The *linenumbering* attribute is set to the AsciiDoc *src_numbered* attribute (*numbered* or *unnumbered*).

4.3 Testing

Test the filter by converting the test file to HTML with AsciiDoc:

```
$ asciidoc -v ./filters/source/source-highlight-filter-test.txt
$ firefox ./filters/source/source-highlight-filter-test.html &
```