TestNG and AspectJ Spring AOP Mix

Project documentation

User Guide

2013-05-11

Table of contents

1 APT Format

1.1 The APT format

In the following section, boxes containing text in typewriter-like font are examples of APT source. Original idea: https://github.com/akquinet/maven-site-tutorial/tree/master/regular-site

1.1.1 Document structure

A short APT document is contained in a single text file. A longer document may be contained in a ordered list of text files. For instance, first text file contains section 1, second text file contains section 2, and so on.

Note:

Splitting the APT document in several text files on a section boundary is not mandatory. The split may occur anywhere. However doing so is recommended because a text file containing a section is by itself a valid APT document.

A file contains a sequence of paragraphs and ``displays'' (non paragraphs such as tables) separated by open lines.

A paragraph is simply a sequence of consecutive text lines.

```
  First line of first paragraph.
  Second line of first paragraph.
  Third line of first paragraph.

  Line 1 of paragraph 2 (separated from first paragraph by an open line).
  Line 2 of paragraph 2.
```

The indentation of the first line of a paragraph is the main method used by an APT processor to recognize the type of the paragraph. For example, a section title must not be indented at all.

A ``plain'' paragraph must be indented by a certain amount of space. For example, a plain paragraph which is not contained in a list may be indented by two spaces.

```
My section title (not indented).

  My paragraph first line (indented by 2 spaces).
```

Indentation is not rigid. Any amount of space will do. You don't even need to use a consistent indentation all over your document. What really matters for an APT processor is whether the paragraph is not indented at all or, when inside a list, whether a paragraph is more or less indented than the first item of the list (more about this later).

```
    First paragraph has its first line indented by four
spaces. Then the author did even bother to indent the
other lines of the paragraph.

  Second paragraph contains several lines which are all
  indented by two spaces. This style is much nicer than
  the one used for the previous paragraph.
```

Note that tabs are expanded with a tab width set to 8.

1.1.2 Document elements

1.1.2.1 Block level elements

1.Title

A title is optional. If used, it must appear as the first block of the document.

```
------
Title
------
Author
------
 Date
```

A title block is indented (centering it is nicer). It begins with a line containing at least 3 dashes (`---`).

After the first `---` line, one or several consecutive lines of text (implicit line break after each line) specify the title of the document.

This text may immediately be followed by another `---` line and one or several consecutive lines of text which specifies the author of the document.

The author sub-block may optionaly be followed by a date sub-block using the same syntax.

The following example is used for a document with an title and a date but with no declared author.

```
------
Title
------
------
 Date
------
```

The last line is ignored. It is just there to make the block nicer.

1.Paragraph

Paragraphs other than the title block may appear before the first section.

```
Paragraph 1, line 1.
Paragraph 1, line 2.

Paragraph 2, line 1.
Paragraph 2, line 2.
```

Paragraphs are indented. They have already been described in the document structure section.

1.Section

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

```
Section title

* Sub-section title

** Sub-sub-section title

*** Sub-sub-sub-section title

**** Sub-sub-sub-sub-section title
```

Section titles are not indented. A sub-section title begins with one asterisk (*), a sub-sub-section title begins with two asterisks (**), and so forth up to four sub-section levels.

1.List

```
      * List item 1.

      * List item 2.

        Paragraph contained in list item 2.

            * Sub-list item 1.

            * Sub-list item 2.

      * List item 3.
```

List items are indented and begin with a asterisk (*).

Plain paragraphs more indented than the first list item are nested in that list. Displays such as tables (not indented) are always nested in the current list.

To nest a list inside a list, indent its first item more than its parent list. To end a list, add a paragraph or list item less indented than the current list.

Section titles always end a list. Displays cannot end a list but the [] pseudo-element may be used to force the end of a list.

```
      * List item 3.
        Force end of list:

      []

-------------------------------------------
Verbatim text not contained in list item 3
-------------------------------------------
```

In the previous example, without the [], the verbatim text (not indented as all displays) would have been contained in list item 3.

A single [] may be used to end several nested lists at the same time. The indentation of [] may be used to specify exactly which lists should be ended. Example:

```
      * List item 1.

      * List item 2.

            * Sub-list item 1.

            * Sub-list item 2.

            []

------------------------------------------------------------------------
Verbatim text contained in list item 2, but not in sub-list item 2
------------------------------------------------------------------------
```

There are three kind of lists, the bulleted lists we have already described, the numbered lists and the definition lists.

```
      [[1]] Numbered item 1.

                [[A]] Numbered item A.

                [[B]] Numbered item B.

      [[2]] Numbered item 2.
```

A numbered list item begins with a label beetween two square brackets. The label of the first item establishes the numbering scheme for the whole list:

[[1]]

Decimal numbering: 1, 2, 3, 4, etc.

[[a]]

Lower-alpha numbering: a, b, c, d, etc.

[[A]]

Upper-alpha numbering: A, B, C, D, etc.

[[i]]

Lower-roman numbering: i, ii, iii, iv, etc.

[[I]]

Upper-roman numbering: I, II, III, IV, etc.

The labels of the items other than the first one are ignored. It is recommended to take the time to type the correct label for each item in order to keep the APT source document readable.

```
      [Defined term 1] of definition list 2.

      [Defined term 2] of definition list 2.
```

A definition list item begins with a defined term: text between square brackets.

1.Verbatim text

```
----------------------------------------
Verbatim
         text,
                preformatted,
                                  escaped.
----------------------------------------
```

A verbatim block is not indented. It begins with a non indented line containing at least 3 dashes (`---`). It ends with a similar line.

`+--` instead of `---` draws a box around verbatim text.

Like in HTML, verbatim text is preformatted. Unlike HTML, verbatim text is escaped: inside a verbatim display, markup is not interpreted by the APT processor.

1.Figure

```
[Figure name] Figure caption
```

A figure block is not indented. It begins with the figure name between square brackets. The figure name is optionally followed by some text: the figure caption.

The figure name is the pathname of the file containing the figure but without an extension. Example: if your figure is contained in `/home/joe/docs/mylogo.jpeg`, the figure name is `/home/joe/docs/mylogo`.

If the figure name comes from a relative pathname (recommended practice) rather than from an absolute pathname, this relative pathname is taken to be relative to the directory of the current APT document (a la HTML) rather than relative to the current working directory.

Why not leave the file extension in the figure name? This is better explained by an example. You need to convert an APT document to PostScript and your figure name is `/home/joe/docs/mylogo`. A APT processor will first try to load `/home/joe/docs/mylogo.eps`. When the desired format is not found, a APT processor tries to convert one of the existing formats. In our example, the APT processor tries to convert `/home/joe/docs/mylogo.jpeg` to encapsulated PostScript.

1.Table

A table block is not indented. It begins with a non indented line containing an asterisk and at least 2 dashes (`*--`). It ends with a similar line.

The first line is not only used to recognize a table but also to specify column justification. In the following example,

- the second asterisk (`*`) is used to specify that column 1 is centered,
- the plus sign (`+`) specifies that column 2 is left aligned,
- the colon (`:`) specifies that column 3 is right aligned.

```
*----------*--------------+----------------:
| Centered | Left-aligned | Right-aligned  |
| cell 1,1 | cell 1,2     | cell 1,3       |
*----------*--------------+----------------:
| cell 2,1 | cell 2,2     | cell 2,3       |
*----------*--------------+----------------:
Table caption
```

Rows are separated by a non indented line beginning with `*--`.

An optional table caption (non indented text) may immediately follow the table.

Rows may contain single line or multiple line cells. Each line of cell text is separated from the adjacent cell by the pipe character (`|`). (`|` may be used in the cell text if quoted: `\|`.)

The last `|` is only used to make the table nicer. The first `|` is not only used to make the table nicer, but also to specify that a grid is to be drawn around table cells.

The following example shows a simple table with no grid and no caption.

```
*-----*------*
 cell | cell
*-----*------*
 cell | cell
*-----*------*
```

1.Horizontal rule

```
====================
```

A non indented line containing at least 3 equal signs (`===`).

1.Page break

```
^L
```

A non indented line containing a single form feed character (Control-L).

1.1.2.2 Text level elements

1.Font

```
  <Italic> font. <<Bold>> font. <<<Monospaced>>> font.
```

Text between < and > must be rendered in italic. Text between << and >> must be rendered in bold. Text between <<< and >>> must be rendered using a monospaced, typewriter-like font.

Font elements may appear anywhere except inside other font elements.

It is not recommended to use font elements inside titles, section titles, links and defined terms because a APT processor automatically applies appropriate font styles to these elements.

1.Anchor and link

```
  {Anchor}. Link to {{anchor}}. Link to {{http://www.pixware.fr}}.
  Link to {{{anchor}showing alternate text}}.
  Link to {{{http://www.pixware.fr}Pixware home page}}.
```

Text between curly braces (`{}`) specifies an anchor. Text between double curly braces (`{{}}`) specifies a link.

It is an error to create a link element that does not refer to an anchor of the same name. The name of an anchor/link is its text with all non alphanumeric characters stripped.

This rule does not apply to links to *external* anchors. Text beginning with `http:/`, `https:/`, `ftp:/`, `file:/`, `mailto:`, `../`, `./` (`..\` and `.\` on Windows) is recognized as an external anchor name.

When the construct **{{{** *name* **}** *text* **}}** is used, the link text *text* may differ from the link name *name*.

Anchor/link elements may appear anywhere except inside other anchor/link elements.

Section titles are implicitly defined anchors.

1.Line break

```
Force line\
break.
```

A backslash character (\) followed by a newline character.

Line breaks must not be used inside titles and tables (which are line oriented blocks with implicit line breaks).

1.Non breaking space

```
Non\ breaking\ space.
```

A backslash character (\) followed by a space character.

1.Special character

```
Escaped special characters: \~, \=, \-, \+, \*, \[, \], \<, \>, \{, \}, \\.
```

In certain contexts, these characters have a special meaning and therefore must be escaped if needed as is. They are escaped by adding a backslash in front of them. The backslash may itself be escaped by adding another backslash in front of it.

Note that an asterisk, for example, needs to be escaped only if its begins a paragraph. (* has no special meaning in the middle of a paragraph.)

```
Copyright symbol: \251, \xA9, \u00a9.
```

Latin-1 characters (whatever is the encoding of the APT document) may be specified by their codes using a backslash followed by one to three octal digits or by using the \x *NN* notation, where *NN* are two hexadecimal digits.

Unicode characters may be specified by their codes using the \u *NNNN* notation, where *NNNN* are four hexadecimal digits.

1.Comment

```
~~Commented out.
```

Text found after two tildes (~~) is ignored up to the end of line.

A line of ~ is often used to ``underline" section titles in order to make them stand out of other paragraphs.

1.1.3 The APT format at a glance

```
                                    ------
                                    Title
                                    ------
                                    Author
                                    ------
                                     Date

  Paragraph 1, line 1.
  Paragraph 1, line 2.

  Paragraph 2, line 1.
  Paragraph 2, line 2.

Section title

* Sub-section title

** Sub-sub-section title

*** Sub-sub-sub-section title

**** Sub-sub-sub-sub-section title

      * List item 1.

      * List item 2.

        Paragraph contained in list item 2.

            * Sub-list item 1.

            * Sub-list item 2.

      * List item 3.
        Force end of list:

      []

+------------------------------------------+
Verbatim text not contained in list item 3
+------------------------------------------+

      [[1]] Numbered item 1.

                [[A]] Numbered item A.

                [[B]] Numbered item B.

      [[2]] Numbered item 2.

  List numbering schemes: [[1]], [[a]], [[A]], [[i]], [[I]].

      [Defined term 1] of definition list.

      [Defined term 2] of definition list.

+--------------------------------+
Verbatim text
                        in a box
+--------------------------------+
```

2 XDoc Example

2.1 Welcome to an XDOC file!

This is some text for the xdoc file.

3 FAQ

3.1 FAQ

1. Where did Maven come from?
2. Why is Maven so wildly popular?

Where did Maven come from?

Maven was created by a group of software developers who were tired of wasting their time fiddling around with builds and wanted to get down to brass tacks and actually develop software!

[top]

Why is Maven so wildly popular?

Maven saves you so much time in your software development efforts that you will have time to learn a second language, relax ten hours a day, and train for that marathon you've always wanted to run!

[top]