RDoc Documentation

Generated July 20, 2011
by the PDF LaTeX generator for RDoc

= PDF Generator for RDoc (based on LaTeX)
This is something *intersting*.

1 Classes and modules overview

2 Methods overview

3 Class and module reference

Class Object

Method list

Constants

PROJECT_TITLE	"LaTeX-PDF...

Public instance methods

Module **RDoc**

Method list

Constants

Public instance methods

Class RDoc::Alias

Method list

#latexized_new_name, #latexized_old_name

Constants

Public class methods

#latexized_new_name

```
latexized_new_name()
```

#latexized_old_name

```
latexized_old_name()
```

Public instance methods

Class RDoc::CodeObject

This file does some monkey patches on RDoc's classes to make it easier to get a LaTeX-conforming represantation that is unique across the whole documentation in a way that it can be used as cross-references for `\label` and `\ref` things. <foo>abc</foo>.

Method list

#latex_description, #latexized_name

Constants

`LATEX_FORMATTER`	`RDoc::Mark...`

Public class methods

#latex_description

```
latex_description()
```

#latexized_name

```
latexized_name()
```

Takes this CodeObject's name and puts it into #latexize.

Public instance methods

Class RDoc::Constant

Method list

#latex_label, #latexized_value

Constants

LATEX_VALUE_LENGTH	10

Public class methods

#latex_label

```
latex_label()
```

#latexized_value

```
latexized_value()
```

Shortens the value to LATEX_VALUE_LENGTH characters (plus ellipsis ...) and escapes all LaTeX control characters.

Public instance methods

Class RDoc::Context

Method list

#latex_label, #latexized_full_name

Constants

Public class methods

#latex_label

```
latex_label()
```

Returns a (hopefully) unique, LaTeX-conforming label string that can be used for cross-references.

#latexized_full_name

```
latexized_full_name()
```

Returns this class'/module's full lexicographical name in a LaTeX-parsable way.

Public instance methods

Module RDoc::Markup

Method list

Constants

Public instance methods

Class RDoc::Markup::ToLaTeX

This is an RDoc Converter/Formatter that turns the RDoc markup into LaTeX code. It's intended for use with the RDoc::Generator::PDF_LaTeX class, but if you like you can use it on it's own (but note this class absolutely depends on RDoc's parser). To use it, you first have to instanciate this class, and then call the #convert method on it with the text you want to convert:

```
f = RDoc::Markup::ToLaTeX.new
f.convert("A *bold* and +typed+ text.")
```

Should result in:

```
A \textbf{bold} and \texttt{typed} text.
```

If for any reason you want to just escape LaTeX control characters, you may do so by calling the #escape method. See it's documentation for an example. This is **bold**.

Method list

#accept_blank_line, #accept_heading, #accept_list_end, #accept_list_item_end, #accept_list_item_start, #accept_list_start, #accept_paragraph, #accept_raw, #accept_rule, #accept_verbatim, #end_accepting, #escape, #init_tags, ::new, #start_accepting, #to_latex

Constants

LATEX_HEADINGS	[nil, ...	LaTeX heading commands. 0 is nil as there is no zeroth heading.
LATEX_SPECIAL_CHARS	{ /\\/ =>...	Characters that need to be escaped for LaTeX and their corresponding escape sequences.
LIST_TYPE2LATEX	{ :BULLET ...	Maps RDoc's list types to the corresponding LaTeX ones. TODO: There are some missing here!

Public class methods

#accept_blank_line

```
accept_blank_line(line)
```

Adds \\, a line break.

#accept_heading

```
accept_heading(head)
```

Adds a fitting \section, \subsection, etc. for the heading.

#accept_list_end

```
accept_list_end(list)
```

Adds \end{list_type}.

#accept_list_item_end

```
accept_list_item_end(item)
```

Adds the terminating newline for an item.

#accept_list_item_start

```
accept_list_item_start(item)
```

Adds \item[label_if_necessary].

#accept_list_start

```
accept_list_start(list)
```

Adds \begin{<list type}.

#accept_paragraph

```
accept_paragraph(par)
```

Adds par's text plus newline to the result.

#accept_raw

```
accept_raw(raw)
```

Writes the raw thing as-is into the document.

#accept_rule

```
accept_rule(rule)
```

Adds a \rule. The rule's height is `rule.weight` pt, the rule's with \textwidth.

#accept_verbatim

```
accept_verbatim(ver)
```

Puts ver's text between \begin{verbatim} and \end{verbatim}

#end_accepting

```
end_accepting()
```

Last method called. Supposed to return the result string.

#escape

```
escape(str)
```

Escapes all LaTeX control characters from a string.
PARAMETER

str The string to remove the characters from.

RETURN VALUE A new string with many backslashes. :-)

EXAMPLE

```
f = RDoc::Markup::ToLaTeX.new
str = "I paid 20$ to buy the_item #15."
puts f.escape(str) #=> I paid 20\$ to buy the\_item \#15.
```

#init_tags

```
init_tags()
```

::new

```
new()
```

Instanciates this formatter.

#start_accepting

```
start_accepting()
```

First method called.

#to_latex

```
to_latex(item)
```

Converts `item` to LaTeX text.

Public instance methods

Class RDoc::MethodAttr

Method list

#latex_label, #latexized_full_name, #latexized_prefix, #latexized_prefix_name

Constants

Public class methods

#latex_label

```
latex_label()
```

Returns a (hopefully) unique, LaTeX-conforming label string that can be used for cross-references.
Return value A string object.

Examples

```
p meth.full_name   #=> "Foo#hello_world"
p meth.latex_label #=> "method-attr-Foo+hello_world

p meth.full_name   #=> "Foo::hello_world"
p meth.latex_label #=> "method-attr-Foo::hello_world
```

#latexized_full_name

```
latexized_full_name()
```

Returns this method's full lexicographical name in a LaTeX-parsable way.

#latexized_prefix

```
latexized_prefix()
```

#latexized_prefix_name

```
latexized_prefix_name()
```

Public instance methods