

reStructuredText Support in Trac

Trac supports using *reStructuredText* (RST) as an alternative to wiki markup in any context [WikiFormatting](#) is used.

From the reStructuredText webpage:

"reStructuredText is an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system. It is useful for in-line program documentation (such as Python docstrings), for quickly creating simple web pages, and for standalone documents. reStructuredText is designed for extensibility for specific application domains. "

If you want a file from your Subversion repository be displayed as reStructuredText in Trac's source browser, set `text/x-rst` as value for the Subversion property `svn:mime-type`. See [this example](#).

Requirements

Note that to activate RST support in Trac, the python docutils package must be installed. If not already available on your operating system, you can download it at the [RST Website](#).

Install docutils using `easy_install docutils`. Do not use the package manager of your OS (e.g. `apt-get install python-docutils`), because Trac will not find docutils then.

More information on RST

- reStructuredText Website -- [?http://docutils.sourceforge.net/rst.html](http://docutils.sourceforge.net/rst.html)
- RST Quick Reference -- [?http://docutils.sourceforge.net/docs/rst/quickref.html](http://docutils.sourceforge.net/docs/rst/quickref.html)

Using RST in Trac

To specify that a block of text should be parsed using RST, use the `rst` processor.

TracLinks in reStructuredText

- Trac provides a custom RST directive `trac::` to allow [TracLinks](#) from within RST text.

Wiki Markup	Display
<pre>{{{ #!rst This is a reference to a ticket .. a ticket trac:: #42 }}}</pre>	This is a reference to #42

- Trac allows an even easier way of creating [TracLinks](#) in RST, using the custom `:trac:` role.

Wiki Markup	Display
<pre>{{{ #!rst This is a reference to ticket `#12`:trac: To learn how to use Trac, see `TracGuide`:trac: }}}</pre>	This is a reference to ticket #12 To learn how to use Trac, see TracGuide

For a complete example of all uses of the `:trac:` role, please see [WikiRestructuredTextLinks](#).

Syntax highlighting in reStructuredText

There is a directive for doing [TracSyntaxColoring](#) in RST as well. The directive is called `code-block`

Wiki Markup

Display

```
{{{
#!rst

.. code-block:: python
    class Test:
        def TestFunction(self):
            pass

def TestFunction(self):
    pass
}}}
```

Note the need to indent the code at least one character after the `.. code-block` directive.

Wiki Macros in reStructuredText

For doing [Wiki Macros](#) in RST you use the same directive as for syntax highlighting i.e `code-block`.

Wiki Markup

Display

```
{{{
#!rst

.. code-block:: RecentChanges
    Trac, 3
}}}
```

11/30/16

- [TracRepositoryAdmin](#)
- [TracTickets](#)
- [TracPlugins](#)

Or a more concise Wiki Macro like syntax is also available, using the `:code-block:` role:

Wiki Markup

Display

```
{{{
#!rst

:code-block:`RecentChanges:Trac, 3`
}}}
```

11/30/16

- [TracRepositoryAdmin](#)
- [TracTickets](#)
- [TracPlugins](#)

Bigger RST Example

The example below should be mostly self-explanatory:

Wiki Markup

Display

```
{{{
#!rst
FooBar Header
=====
reStructuredText is nice. It has its own webpage_.
A table:
=====  =====  =====
Inputs      Output
}}}
```

reStructuredText is **nice**. It has its own [webpage](#).

A table:

```

-----
  A      B      A or B
=====
False  False  False
True   False  True
False  True   True
True   True   True
=====

```

RST TracLinks

See also ticket `#42`:trac:.

```

.. _webpage: http://docutils.sourceforge.net/rst.html
}}}

```

Inputs		Output
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

RST TracLinks

See also ticket [#42](#).

See also: [WikiRestructuredTextLinks](#), [WikiProcessors](#), [WikiFormatting](#)