

Table of Contents

reStructuredText Support in Trac	3
Requirements	3
More information on RST	3
<i>Using RST in Trac</i>	3
TracLinks in reStructuredText	3
Syntax highlighting in reStructuredText	4
WikiMacros in reStructuredText	4
WikiMacro Example	5
Bigger ReST Example	5

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.*"

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](#).

More information on RST

- reStructuredText Website - <http://docutils.sourceforge.net/rst.html>
- RST Quick Reference - <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 reference-directive 'trac' to allow TracLinks from within RST text.

Example:

```
<code>
#!rst
This is a reference to |a ticket|

.. |a ticket| trac:: #42
```

```
</code>
```

For a complete example of all uses of the *trac*-directive, please see WikiRestructuredTextLinks.

- Trac allows an even easier way of creating TracLinks in RST, using the custom `:trac:` link naming scheme.

Example:

```
<code>
#!rst
This is a reference to ticket `#12`:trac:

To learn how to use Trac, see `TracGuide`:trac:
```

```
</code>
```

Syntax highlighting in reStructuredText

There is a directive for doing TracSyntaxColoring in ReST as well. The directive is called `code-block`

Example

```
<code>
#!rst

.. code-block:: python

class Test:

    def TestFunction(self):
        pass
```

```
</code>
```

Will result in the below.

```
#!rst

.. code-block:: python

class Test:

    def TestFunction(self):
        pass
```

WikiMacros in reStructuredText

For doing WikiMacros in ReST you use the same directive as for syntax highlighting i.e `code-block`. To

work you must use a version of trac that has #801 applied.

WikiMacro Example

```
<code>
#!rst

.. code-block:: HelloWorld

    Something I wanted to say

```

```
</code>
```

Will result in the below.

[I wanted to say](#))

Bigger ReST Example

The example below should be mostly self-explanatory:

```
#!html
<pre class="wiki"><code>
#!rst
FooBar Header
=====
reStructuredText is nice. It has its own webpage_.

A table:

= = =====
   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`:trac:.
```

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

```
</pre> </code>
```

Results in:

```
#!rst
FooBar Header
=====
reStructuredText is nice. It has its own webpage_.
```

A table:

```
= = =====
  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`:trac:.
```

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

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

From:

<https://wiki.linux-ottawa.org/> - **Linux-Ottawa (OCLUG) Wiki**

Permanent link:

<https://wiki.linux-ottawa.org/doku.php?id=wikirestructuredtext>

Last update: **2015/06/09 19:23**

