

Wikiprint Book

Title: reStructuredText Support in Trac

Subject: eDokumenty - elektroniczny system obiegu dokumentów, workflow i CRM - WikiRestructuredText

Version: 1

Date: 04/21/26 13:43:28

Table of Contents

<i>reStructuredText Support in Trac</i>	3
<i>Requirements</i>	3
<i>More information on RST</i>	3
<i>Using RST in Trac</i>	3
<i>TracLinks in reStructuredText</i>	3
<i>Syntax highlighting in reStructuredText</i>	3
<i>Wiki Macros in reStructuredText</i>	4
<i>Wiki Macro Example</i>	4
<i>05/22/09</i>	4
<i>Bigger ReST Example</i>	4
<i>Foobar Header</i>	5
<i>RST TracLinks</i>	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:

```

{{{
#!rst
This is a reference to |a ticket|

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

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:

```

{{{
#!rst
This is a reference to ticket `#12`:trac:

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

Syntax highlighting in reStructuredText

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

Example

```

{{{
#!rst

.. code-block:: python
```

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

Will result in the below.

```
class Test:
    def TestFunction(self):
        pass
```

Wiki Macros in reStructuredText

For doing [Wiki Macros](#) 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.

Wiki Macro Example

```
{{{
#!rst

.. code-block:: RecentChanges

    Trac,3
}}}
```

Will result in the below:

05/22/09

- [TracRevisionLog](#)
- [TracUnicode](#)
- [TracLinks](#)

Or a more concise Wiki Macro like syntax is also available:

```
{{{
#!rst

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

Bigger ReST Example

The example below should be mostly self-explanatory:

```
{{{
#!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
}}}

```

Results in:

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](#).

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