

Rusty documentation

version 0.4.0.dev

Juha Mustonen

August 23, 2009

Contents

Rusty	4
Features	4
Quick introduction	4
Further information	5
Changelog	5
Release 0.4.0 (in development)	5
Release 0.3.1 (2009-08-08)	6
Release 0.3.0 (2009-07-07)	6
Release 0.2.2 (2009-06-18)	6
Release 0.2.1 (2009-06-16)	6
Release 0.2.0 (2009-06-01)	6
Release 0.1.0 (2009-03-31)	6
Development	6
Building	7
Testing	7
Licensing	7
Frequently asked questions	8
Glossary	8
rusty.xmltable -- XMLTable directive	9
Features	9
Usage	10
Example	12
Development	13
Module in detail	13
rusty.includesh -- Include shell scripts in documentation	13
Features	14
Usage	14
Module in detail	17
rusty.rolelist -- List document roles	17
Features	18
Usage	19
Module in detail	21
rusty.regxlist -- List regular expression matching entries	22
Example	23
Usage	25

Module in detail	27
rusty.exceltable -- ExcelTable directive	28
Usage	29
Example	30
Module in detail	32
Index	34
Global Module Index	35

Rusty

Rusty is a *collection of extensions (directives and roles)* for [Sphinx documentation framework](#). While the extensions are somewhat compatible with [docutils](#), the usage of *Sphinx is currently required*.

This document describes *how to install and develop* the rusty module. For the usage, select the extension of your choice from the *feature listing below*.

Features

Following extensions are currently available in the Rusty module. The detailed usage information for each extension are described in the subsections:

rusty.exceltable

Creates table from selected part of the Excel -document. Requires [xlrd](#) -module.

rusty.includesh

Extends the standard [include](#) directive by converting the given shell script file into RST format: comments are formatted into text and commands put in the code blocks.

rusty.regxlist

Creates bullet list based on *regular expression*. Capable to match both current document or any other file. Otherwise similar to `rolelist` directive.

rusty.rolelist

Creates the bullet list from all the entries of the selected role, with some additional ways to custom the output.

rusty.xmltable

Creates RST table from XML-document and based on query. Requires [BeautifulSoup](#) -module.

Quick introduction

Here you can find the minimum steps for installation and usage of the module:

1. Install `setuptools`:

```
wget peak.telecommunity.com/dist/ez_setup.py
sudo python ez_setup.py
```

2. Install [Rusty](#), [Sphinx](#) and additional modules:

```
sudo easy_install -U rusty Sphinx BeautifulSoup xlrd
```

Further information

3. Start new Sphinx powered documentation project or continue with existing documentation:

```
sphinx-quickstart
```

4. Configure rusty directive or role of your choice into Sphinx conf.py configuration file. See *features* for available extensions.

```
# Add ``rusty`` into extension list
extensions = ['sphinx.ext.autodoc', 'sphinx.ext.doctest', 'rusty.includesh']
```

5. Place directive/role in your document:

```
My document
=====
The contents of the setup script:

.. includesh:: setup.sh
```

6. Build the document:

```
sphinx-build -b html doc dist/html
```

7. That's it!

Further information

Interested in module? See following sections for further information about it.

Changelog

This sections lists the biggest changes done on each release.

Release 0.4.0 (in development)	5
Release 0.3.1 (2009-08-08)	6
Release 0.3.0 (2009-07-07)	6
Release 0.2.2 (2009-06-18)	6
Release 0.2.1 (2009-06-16)	6
Release 0.2.0 (2009-06-01)	6
Release 0.1.0 (2009-03-31)	6

Release 0.4.0 (in development)

Changes since release 0.3.0:

- € Added `file`-option to `reglist`-directive. It allows to create regexp matching list from any file and not just from the current document.
- € Fixed issue: the selection does not work with numbers containing a zero
- € Fixed issue: `paver-minilib.zip` is missing from the egg-package
- € Added support for matching reference URLs with `rolelist`-directive. This is done by using `refuri` flag. It comes handy when listing entries managed also by `sphinx.ext.extlinks`. See documentation for details.

Release 0.3.1 (2009-08-08)

- € Added quote support in `regxlist` regular expression definition: if the value starts and ends with `"`, they will be stripped. Then again, if the regexp contains some special characters like colon (`:`), quotes needs to be used.
- € Added quote support in `rolelist` value definition: if the value starts and ends with `"`, they will be stripped. Then again, if the regexp contains some special characters like colon (`:`), quotes needs to be used.
- € Improvements on building
- € Added icons for extensions
- € Added optional support for building PDF documents with `rst2pdf`

Release 0.3.1 (2009-08-08)

Changes since release 0.3.0:

- € Fixed issue: the selection does not work with numbers containing a zero

Release 0.3.0 (2009-07-07)

Changes since previous release:

- € Added new directive: `exceltable`
- € Improved `XMLTable` documentation
- € Fixed issue: `xmltable` does not stop processing even if file or query is missing
- € Improved python 2.4 compatibility: do not require `elementtree` module

Release 0.2.2 (2009-06-18)

Maintenance release, containing following changes:

- € Fixed 4 again: `paver-minilib.zip` is missing from the package

Release 0.2.1 (2009-06-16)

Changes since previous release:

- € Fixed 4: `setup.py` is missing from the package

Release 0.2.0 (2009-06-01)

Changes since previous release:

- € New directive: `xmltable`
- € New directive: `regxlist`
- € Improved unit testing
- € Improved building
- € Migrated version control from subversion to mercurial
- € Published the project in Bitbucket: <http://bitbucket.org/jmu/rusty>
- € Added FAQ

Release 0.1.0 (2009-03-31)

First release, containing following functionality

- € New directive: `includesh`
- € New directive: `rolelist`
- € Initial set of unit tests (see *testing*)
- € `Paver` powered build and release management

Development

Building

This section contains information for further development of the module.

Building

Python-based scription solution [Paver](#) scripting tool solution is used to build and distribute the software. Since the package already contains the dependencies to Paver, the basic building can be done as follows

```
python setup.py bdist_egg sdist_src
```

However, installation of the paver is strongly suggested. After installation, the documentation, building and packaging can be all in once:

```
paver package
```

To build just the documentation, run:

```
paver doc
```

Testing

The unit testing in Rusty is implemented by eating the own dog food: the latest version of extensions are always used to generate content the documentation. In addition to user documentation, there exists a separate test documentation that tries the different variations of the module usage.

To run the tests (to generate the test documentation), run shell command:

```
python setup.py test  
  
OR  
  
paver test
```

Licensing

The software is licensed with liberal MIT license, making it suitable for both commercial and open source usage:

The MIT License

Copyright (c) 2009 Juha Mustonen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

Frequently asked questions

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Frequently asked questions

Why rusty?

I think that documentation *does not need* to be a **necessary evil**. With a flexible tools even developers kind enjoy writing documentation.

Documentation framework [Sphinx](#) is an excellent tryout to make writing technical documents easier. With the [ReStructuredText](#) -documentation format it shows that it does not necessary have to be Word document - while providing advanced functionality to write or generate documentation. As fun as the document writing can be, we still want to get it does easily and fast, right?

So, **why rusty, then?** [Rusty](#) is my push to support non-binary documents and provide some examples how the people can take the advantage from the ReST - and this is done by writing extensions that I've found useful while writing documents.

What's with the name?

I like names that does not necessary mean anything but may sound like one. So, the name **Rusty** comes from:

- € Trust, trustworthy
- € tRuSTy => RST => ReStructuredText
- € Rust (?)
- € Python => py => rusTY

Development status of rusty?

Or how ready rusty is for real usage?

Glossary

directive

Directives are extension blocks that provides certain functionality into document. A distinctive difference with *roles* is that the latter are in-line definitions.

RST document with directive:

```
.. directive: optional argument
   :option1: opt value1
   :option2:
   content
```

regexp

Regular expression: a specific textual syntax used to match with the string. Powerful but somewhat complicated.

See [further information from Wikipedia](#).

role

In-line extensions. Usually used for defining part of the content:

```
Some text containing role:`target-name` as well as
other kind of role:`targets <target-name>`
```

```
Copy the :file: `app.conf` to :directory: `/etc/`
```

rusty.xmltable -- XMLTable directive

Platforms: Unix, Windows

Versionmodified

Note

The module depends on [BeautifulSoup](#) -module. Install it using `setuptools`:

```
sudo easy_install BeautifulSoup
```

Features	4
Development	6
Features	9
Usage	10
Example	12
Development	13
Module in detail	13
Features	14
Usage	14
Features	18
Usage	19
Example	23
Usage	25
Usage	29
Example	30

Features

XMLTable directive generates the table based on XML document and query, defined in the document. The *usage is simple*, but should cover most of the cases.

Iteration

The given query iterates all the items that matches with document.

Values

XML element values can be listed in a table cells. The selector is relative to iteration query, but can be also the parent and child elemnt of it. The text between elements is automatically taken, but it can be noted with `@value` -notation as well.

Attributes

XML element attributes can be shown as a value. The attribute is noted with `@attribute-name` -notation.

Header

Usage

Header columns can be defined for the table. Separate the values with comma.

Usage

Define `xml table` -directive into your document. The path to document is given with `file` option, and it is relative to document path. The directive argument is reserved for the optional table caption.

```
This is an example rst-document

.. xml table:: caption
   :file: path/to/document.xml
   :header: Name, Name, Name, Name
   :query: /xml /query

   element-name
   element-name@attrib
   self
   @attrib

* list
* item

And so on..
```

As it can be seen from the example, there are few options and content defined for the element. The description for each:

file (required)

Relative path (based on document) to XML file. Compulsory option.

header (optional)

Optional column fields for the header. If not defined, no table header is created. *Separate the column names with comma (,)*. Quotes are not needed and they are actually shown as is.

If the header is not defined, the generated table won't have either. If the number of header entries does not match with the rows (columns) of the directive content area, an error is raised.

query

XML query, or more like a path, **defines the name of elements that will be iterated from the XML document. Each element will create a new row to table.** For example, if XML document contains multiple `<animal>` elements (within `<zoo>` element), each entry is shown as a new row with the definition:

```
/zoo/animal
```

Then, the *content area* defines from which sub-elements the string value is taken from:

```
.. xml table:
   :file: animals.xml
   :query: /zoo/animal

   column1
```

Usage

```
column2
```

The query consists only from the strings and slashes. The first slash is ignored as the path always starts from the beginning of the XML document. Example:

```
/world/countries/country
```

See *advanced example*

widths

Optional parameter for defining the size of the column widths. The expected values are integers, separated with comma (the number of values must match) with number of columns.

content

Each row on the `xml table` -directive content area **defines the path to the sub-element** which string value is taken into table cell. To continue with the animal-example, let's consider following document:

```
<zoo>
  <animal id="L353">
    <name>Leo</name>
    <species>Lion</species>
  </animal>
  <animal id="C665">
    <name>Carl</name>
    <species>Camel</species>
  </animal>
</zoo>
```

Following directive iterates each animal and prints some info about it into table. Notice the matching header definition:

```
.. xml table:: Animal table caption
:file: example/animals.xml
:header: Name, Species, ID
:query: /zoo/animal

name
species
sel f@id
```

Producing following table:

Animal table caption

Name	Species	ID
Leo	Lion	L353
Carl	Camel	C665

If the selected element does not have string value, all the string values from the sub-elements are taken. Example:

Example

```
.. xml table: List of countries
: file: document.xml
: header: Description
: query: countries/country

sel f

.. xml table: List of countries
: file: example/document.xml
: header: Description
: query: countries/country

sel f
```

Example

Sometimes the example is the best way to learn. Consider this example:

XML document

First we have a XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<countries>
  <country id="finland">
    <name>Finland</name>
    <time>UTC+2</time>
    <currency>euro</currency>
  </country>
  <country id="england">
    <name>England</name>
    <currency>pound</currency>
    <time>UTC</time>
  </country>
</countries>
```

RST document

Now the text document could have something like:

```
.. xml table: List of countries
: file: document.xml
: header: ID, Name, Time, Currency, All
: widths: 60, 10, 10, 10
: query: /countries/country

name
sel f@id
time
currency
sel f
```

Note

Development

Note the special `self`-keyword: it references to the last element defined with the query. In the case of example, the `self` references to `country` element and `self@id` to `id` attribute element.
If the selected element has no string value, it retrieves the string from the subelements recursively.

Output

Once generated, the output shows the contents of the XML in table format.

List of countries

ID	Name	Time	Currency	All
Finland	finland	UTC+2	euro	Finland euro
England	england	UTC	pound	England pound UTC

Development

Known issues or development ideas for directive. May be fixed/implemented in the future.

- € File path cannot contain quotes (are spaces supported?)
- € Multiple entries per field
- € Support for remote resources

Module in detail

`XMLTableDirective` implements the `xml table`-directive. The implementation is based on two existing solutions:

- € `ListTable`, from the `docutils` module
- € `BeautifulStoneSoup`, from the `BeautifulSoup` module

The module introduces simple yet effective query language, that is used to select and generate the RST-table from the XML-document.

Desc

class

`rusty.xml table`. `XMLTableDirective`(*name,arguments,options,content,lineno,content_offset,block_text,*
`XMLTableDirective` implements the new restructured text directive. It's purpose is to provide an easy-to-use XML into RST table transition - in a form of `docutils` directive.

Desc

class `rusty.xml table`. `BeautifulTable`(*fobj*)

Class generates the list based table from the given document, suitable for the directive.

Class also implements the custom query format, is to use for the directive. Examples:

```
/root-el em/sub-el em/el em <- text value
/root-el em/sub-el em@attr <- attr value
```

Internally, the `BeautifulTable` uses the `BeautifulStoneSoup`, from [BeautifulSoup](#).

rusty. includesh -- Include shell scripts in documentation

Platforms: Unix, Windows

Features

Module extends the standard `include` directive by making it possible to include shell scripts into documents, while providing following features listed below. See *includesh-usage* for detailed example how to use the extension.

Features	4
Features	9
Usage	10
Features	14
Usage	14
Module in detail	17
Features	18
Usage	19
Usage	25
Usage	29

Features

Format conversion

Given shell file is converted into RST format using following rules:

- € Shell commands are turned into code-blocks, using bash highlight
- € Comments are turned into RST format, allowing standard formatting rules

Inclusion

Only partial inclusion is supported by providing options:

- € *start-after*: text to find in the external data file Only the content after the first occurrence of the specified text will be included.
- € *end-before*: text to find in the external data file Only the content before the first occurrence of the specified text (but after any after text) will be included.
- € *encoding*: name of text encoding The text encoding of the external data file. Defaults to the document's encoding (if specified).

Exclusion

Lines starting with double-comment character in shell file are skipped while converting to document

Note

- € Configuration parameter `file_inclusion_enabled` *must not be disabled* in order to use this directive.
- € Do not put the inclusion flags (*start-after*, *end-before*) after the double-comment characters, as they won't be recognized.

Usage

The usage of the extension can be described best by showing a real world example:

Software installation consists from several steps of shell commands - which are described in documentation, one by one. However, to prevent user to run same installation steps manually, one could create a complete shell script from the installations steps.

Features

While easy-to-run script is nice, the user ends up running the script blindfolded - without actually knowing what is happening as part of the installation process. Thus, to increase the knowledge about the software, it would be good to have the actual steps documented as well. Having same steps duplicated in script and documents leads eventually in out-dated material.

This directive helps to get best parts from both: having up-to-date setup script AND documentation. By generating the documentation from the script - and here's how:

Shell script: setup.sh

Consider having shell script, that is used for example as part of the installation process.

```
#!/bin/sh
##=====
## This is a application installation
## script. The comments are formatted using RST
##=====
# Installation consists from three phases:
#
# 1. Downloading files
# 2. Installation
# 3. Configuration
#
# Create temp directory
CDATE=$(date +%Y-%m-%d)
mkdir -P /tmp/$CDATE

# .. _clear_temp:
#
# In case of existing files, delete the temporary files first
rm -rf /tmp/$CDATE

# Download the package and locate it to ``temp`` directory.
# Finally, extract the package.
cd /tmp/$CDATE
wget http://server.com/downloads/package.tar.gz
tar -xzf package.tar.gz

# .. IMPORTANT: :
#
# In case of upgrade, :ref:`delete the temporary files <clear_temp>`.
#

# Open the config file in editor
vim /etc/application.conf

# Change the parameter ``listen_port`` if needed. The default value is: 8080: :
#
# listen_address = "0:0:0:0"
# listen_port = 8080
#
# Start the service to verify the configuration is valid
service application start

##
```

Document: document.rst

In the documentation, the shell script can be included as a part of the documentation, by

Features

using `includedesh` directive. The path to script is relative to the document.

Installation of the application can be completed by running shell script named ```setup.sh```:

```
chmod +x setup.sh
./setup.sh
```

Alternatively, you can run each command manually:

```
.. includedesh:: setup.sh
```

And you're done with installation. Next phase is to change the configuration:

1. Place the initial configuration in ```/etc/app.conf```
2. Open the config in editor and change as needed:

```
vim /etc/app.conf
```

```
.. includedesh:: app.conf
: start-after: <changethese>
: end-before: </changethese>
```

3. After finished, start the service:

```
/etc/init.d/app start
```

Output

When the document is processed, the output is shown as follows (compare the output with `setup.sh` script and document.rst):

Installation consists from three phases:

1. Downloading files
2. Installation
3. Configuration

Create temp directory

```
CDATE=$(date +%Y-%m-%d)
mkdir -P /tmp/$CDATE
```

In case of existing files, delete the temporary files first

```
rm -rf /tmp/$CDATA
```

Download the package and locate it to temp directory. Finally, extract the package.

```
cd /tmp/$CDATE
wget http://server.com/downloads/package.tar.gz
tar -xzf package.tar.gz
```

Module in detail

Important

In case of upgrade, *delete the temporary files.*

Open the config file in editor

```
vi m /etc/appl i cati on. conf
```

Change the parameter `l i sten_port` if needed. The default value is: 8080:

```
l i sten_address = "0: 0: 0: 0"  
l i sten_port = 8080
```

Start the service to verify the configuration is valid

```
servi ce appl i cati on start
```

Module in detail

This module provides the `i ncl udes h` directive. It is implemented by `I ncl udeShel l Di recti ve` class.

Desc

class

rusty. `i ncl udes h`. `I ncl udeShel l Di recti ve(name,arguments,options,content,lineno,content_offset,block)`
`IncludeShellDirective` implements the directive. The class is registered as a directive in `rusty. i ncl udes h. setup()`

Desc

class

rusty. `i ncl udes h`. `Shel l Converter(script_path,encoding='utf-8',comment_character='#')`
Converts shell script into another format - restructured documentation format in this case.

```
>>> sc = Shel l Converter(scri pt_path=' /tmp/scri pt. sh' , comment_character=' #' )  
>>> sc. to_rst(target_path=' /tmp/scri pt. output. rst' )
```

rusty. rolel i st -- List document roles

Platforms: Unix, Windows

Features	4
Features	9
Usage	10
Features	14
Usage	14
Features	18
Usage	19
Module in detail	21

Usage	25
Usage	29

Features

Rolelist *directive* provides a generic way to list *role* entries from current document. While it may not sound interesting, consider following use cases:

Changelog

This use case happens to be initial reason for writing the directive. Consider the changelog/release notes document, containing a list of new features, bug fixes and other changes. By using a role, each change can be separated for each other, by listing them in a separate list:

```

=====
Release Notes
=====

Release X
=====
Following issues have been fixed in this release:

Fixes

.. rolelist:: bug
   : template: #${value}: ${text}
   : level sup: 2
   : siblings:

New features

.. rolelist:: feat
   : level sup: 2
   : siblings:

All changes
-----
* Implemented the requested feature :feat: `Support for python 2.6 <235>`
* :feat: `Added support for MacOS`
* Fixed issue :bug: `285`
* Fixed issue :bug: `Crash on start-up <284>`

```

Note

The bug and ref used in the example are not part of the Sphinx nor docutils but custom roles. These and similar definitions can be added in Sphinx conf. py:

```

def setup(app):
    app.add_description_unit('bug', 'bug',
        'pair: %s; Defect')

```

Important

Using Sphinx's `extlinks -role?` Due the technical reasons, they cannot be referenced in traditional manner. Instead, use `refuri` flag and define the matching *regexp* for the `rolelist`. See `refuri` description for details.

Usage

The complete syntax for the directive:

```
.. rolelist:: [role-name]
   : template: [optional, with string value]
   : docwide:  [optional, set to list roles from non-nested sections - no value]
   : level sup: [optional, set with integer how high to climb in document tree]
   : siblings: [optional, controller option to leave out the sibling nodes
               when looking for role entries. The option *has no value* and
               siblings are *leave out by default*]
   : refuri:   [optional, flag whether to match link URLs instead of role
               names. See complete description below]
```

Which breaks down as follows:

role-name

required, string value

Name of the role entries that are wanted to be listed in the bullet list. See *example*

template

optional, string value

String template is used for formatting the bullet list items texts. The supported keywords are:

€ **value**: The contents between `'` or `<>`

€ **text**: The contents of `:role:'this if key is defined <key>'`

The keywords are marked with notation: `$keyword` or `${keyword}`. The following example is the default template, as the *option is optional*:

```
${text} (${value})
```

For example with `:file:` roles you might want to use following template instead of the default one:

```
.. rolelist:: file
   : template: $text
```

docwide

optional, no value

A option flag to search only in nested sections or thorough the whole document. The flag is **optional**, and the default value is `False`.

Important

If the directive is in section and `docwide` is disabled, it cannot see the roles from sibling section - only nested.

Usage

levelsup

optional, positive integer

Levelsup is an option that controls where to retrieve the roles - or how many steps to take up in the document tree nodes, to be exact. The allowed value is positive integer (≥ 1) and *default value is 0*. All the nested nodes are always visited when searching for roles.

Note

If the value for the Levelsup is very high, the endresult is the same as with docwiden option.

To get the idea from the option, let's see the example. The Levelsup options is required as the values are separate section:

Bug fixes

=====

```
.. rolelist:: rolename
   : levelsup: 1
   : template: #${value}: ${text}
   : siblings:
```

Changes

=====

- * Fixed :bug: `application crashes <235>` -issue
- * Fixed :bug: `application fails to load document <236>` -issue

siblings

optional, no value

controller option to take in the sibling nodes when looking for role entries. The option *has no value* and siblings are left out *by default*. Example:

Title

=====

Section A

The contents from next section are taken in now that ``siblings`` flag is set.

```
.. rolelist:: role
   : levelsup: 1
   : siblings:
```

Section B

These entries are found:

- * :role: `x`
- * :role: `y`
- * :role: `z`

Module in detail

refuri

optional, no value

Versionmodified

When `refuri` -flag is set, the value set as a *role* is considered as a *regex* that matches with the URL defined in reference (`refuri`). The usage of `refuri` is needed for example with Sphinx's `extlinks` where the roles are already transformed into references and no normal role matching can be used. Example:

Configuration:

```
extensions = [  
    'sphinx.ext.extlinks',  
    'rusty.rolelist',  
    # possibly few others as well...  
]  
  
extlinks = {'issue': ('http://bitbucket.org/jmu/rusty', 'issue')}
```

Document:

```
.. rolelist:: http://bitbucket.org/jmu/rusty/. *  
   :refuri:  
   :siblings:  
  
* Fixed :issue:`3`: Lorem ipsum  
* Fixed :issue:`4`: Lorem ipsum
```

Output:

```
€ No entries  
€ Fixed 3: lorem ipsum  
€ Fixed 4: lorem ipsum
```

Note

Rolelist can find the roles only for the current document - i.e. the documents included by Sphinx doctree *are not covered*. However, the documents imported by the `include` directive are read.

If the generated list is empty, the translatable string for having "No entries" item is generated in the list.

Tip

Finding a suitable selector (using options like `levelsup` and `siblings`) is sometimes a work of trial and error. This is due the node structure of the docutils.

In a case you don't get any results into list, try following:

1. Set `siblings` option to directive
2. Try increasing the number in `levelsup` option

Module in detail

Rolelist module implements the `rolelist` directive, which can be used for example as

rusty.regxlist -- List regular expression matching entries

follows:

```
Document with :file:`conf.xml` and :file:`conf.xml`.
```

```
The files listed in document:
```

```
.. rolelist:: file
```

Desc

```
class rusty.rolelist.RoleListNode(rawsource="", *children, **attributes)
```

Custom placeholder node that is replaced in process_rolelist

Desc

class

```
rusty.rolelist.RoleListDirective(name, arguments, options, content, lineno, content_offset, block_text, source)
```

IncludeShellDirective implements the directive. The class is registered as a directive in

rusty.rolelist.setup().

The structure of the directive:

```
.. rolelist:: name-of-role
   : template: {text} - {value}
   : docwideness:
   : levelsup: posit
   : siblings:
```

Desc

run()

Implements the directive. In this case, it only stores the given argument and options into custom RoleListNode object - which are taken from there in process_rolelist().

Desc

```
static rusty.rolelist.process_rolelist(app, doctree, docname=None)
```

Iterates the document nodes once the doctree has been constructed and replaces the custom rolelist node with actual content (a bullet list of selected role)

Desc

```
static rusty.rolelist.setup(app)
```

Extension setup, called by Sphinx

rusty.regxlist -- List regular expression matching entries

Platforms: Unix, Windows

Versionmodified

Regxlist (a.k.a *regular expression list*) is a *directive* that can create a bullet list from all the document entries matching with the defined *regular expression*.

The directive is a spin-off from rolelist -directive, but instead of matching the roles, the regxlist can list any entries matching with regular expression rule. This might be the wanted situation in some of the *use cases*.

Usage	10
Example	12
Usage	14
Usage	19

Example

Example	23
Usage	25
Module in detail	27
Usage	29
Example	30

Example

The initial reason for writing the *rolelist* -directive was the requirement to generate list of fixes and features listed in the changelog/release notes. After completing the rolelist it was noticed that role definition may not always be available/wanted. To provide alternative solution, regxlist directive was created. Also, regxlist -directive allows to match query against any file, not just current document.

Regxlist is handy both listing bug fixes and for example picking the action points from the meeting memos. Consider following example:

Source document

First example is an imaginary release notes where features and bug fixes are retrieved from the complete changelog.

```
This is an example document that uses the ``regxlist`` directive. One of the most obvious use cases is the release notes and generating the list of bug fixes.
```

****Release X****

Features:

```
.. regxlist:: Added: \s*(. *)
   : siblings:
   : level sup: 2
   : template: FEAT: ${0}
```

Bug fixes:

```
.. regxlist:: Fixed (#\d+)(:\s*)(?P<desc>. *)
   : siblings:
   : level sup: 2
   : template: BUG: ${desc} (${0})
```

All changes:

- Fixed #2345: Crash while loading invalid document
- Fixed #3454: Configuration parameter is missing
- Added: Support for Vista
- Changed default configuration parameter for ``show_version`` to ``False``

Second is an example meeting minutes where there often are actions points (AP) listed - and by highlighting them they hopefully will be done :)

****Meeting minutes****

Example

```
:author: Alan Author
:date: 2008-04-21

.. IMPORTANT:: Action points

.. regxlist:: (AP|-->)+\s+(to)*\s*(?P<name>(\w|\s)+): \s*(?P<desc>(\w|\n|\s)*)
: template: ${name}: ${desc}
: siblings:
: level sup: 2

- Software building process is still hard, AP Sarah: improve building process
- AP to Tom: document the building process
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean risus mauris,
  ultrices id, pretium sed, lobortis in, nisl. Nunc tincidunt neque vel libero
  hendrerit malesuada.
- Pellentesque dolor augue, dapibus dictum, elementum quis, tincidunt consectetur,
  nunc. Sed vulputate dolor ut sem. In varius rutrum odio.
- Release needs to be ready for monday --> Sarah: create official release
- Nunc sit amet neque sed lorem condimentum interdum. In hac habitasse platea dictumst.
  Vivamus vel libero eget lectus ultrices vestibulum.
- Blah blah...
```

Output

Output of the processed documents shown above.

This is an example document that uses the `regxlist` directive. One of the most obvious use cases is the release notes and generating the list of bug fixes.

Release X

Features:

- € FEAT: Support for Vista

Bug fixes:

- € BUG: Crash while loading invalid document (#2345)
- € BUG: Configuration parameter is missing (#3454)

All changes:

- € Fixed #2345: Crash while loading invalid document
- € Fixed #3454: Configuration parameter is missing
- € Added: Support for Vista
- € Changed default configuration parameter for `show_version` to `False`

Meeting minutes

author: Alan Author

date: 2008-04-21

Important

Action points

- € Sarah: improve building process
- € Tom: document the building process
- € Sarah: create official release

Usage

- € Software building process is still hard, AP Sarah: improve building process
- € AP to Tom: document the building process
- € Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean risus mauris, ultrices id, pretium sed, lobortis in, nisl. Nunc tincidunt neque vel libero hendrerit malesuada.
- € Pellentesque dolor augue, dapibus dictum, elementum quis, tincidunt consectetur, nunc. Sed vulputate dolor ut sem. In varius rutrum odio.
- € Release needs to be ready for monday --> Sarah: create official release
- € Nunc sit amet neque sed lorem condimentum interdum. In hac habitasse platea dictumst. Vivamus vel libero eget lectus ultrices vestibulum.
- € Blah blah...

Usage

The complete syntax for the directive:

```
.. regxlist:: [regexp-rule]
:template: [optional, with string value]
:docwide: [optional, set to list entries from the whole document. The option *has no value*]
:levelsup: [optional, set with integer how high to climb in document tree]
:siblings: [optional, controller option to leave out the sibling nodes
            when looking for role entries. The option *has no value* and
            siblings are *leave out by default*]
:file: [optional, if defined the regexp rules matches against the given
        file and the current document. File is relative path to document]
```

Which breaks down as follows:

regexp-rule

required, string value

Regular expression matching rule. The rule is likely to contain groups that can be then again listed in template:

```
.. regxlist:: Fixed: (\d\d\d\d)
```

Note

In a case you need to match text line written on multiple lines, remember to put `\n` in regexp rule:

```
.. regxlist:: (?P<desc>(\w|\n|\s)*)
```

file

optional, uri to file

By default, the regxlist matching is done into current (*processed*) RST -document. With the `file` -option, the regexp matching can be done in any document, say, txt-file or source code. Example:

```
.. regxlist:: New\s+(?P<what>\w*)
:file: ../CHANGES
:template: Implemented: ${what}
```

Output:

- € Implemented: directive

Usage

- € Implemented: directive
- € Implemented: directive
- € Implemented: directive
- € Implemented: directive

The file option expects to have path to matching document, relative to the current RST -document.

Note

- € Even if the given document is RST -document, it is not processed: the regexp matching is done into raw source
- € When the file is defined, the docwide, level sup and siblings -options *are not used*.

template

optional, string value

String template containing the placeholders. Template is used for formatting the bullet list items texts. The names of placeholders comes from the regular expression groups. Both **positional** and **named** entries are supported. Consider following *regular expression* and string:

```
regexp: Fixed #(\d+)\s*(?P<desc>. *)
string: Fixed #23453 compilation fails
```

Each (...) marks the group. The reference to group can be done with number (positional), starting from number 0. If expression contains the (?P<name> ...) -part, it can be referenced *also* with name. Thus, the previous example can be referenced as follows:

```
#{0}    --> 23453
#{1}    --> compilation fails
#{desc} --> compilation fails
```

docwide

optional, no value

A option flag to search only in nested sections or thorough the whole document. The flag is **optional**, and the default value is False.

Important

If the directive is in section and docwide is disabled, it cannot see the roles from sibling section - only nested.

levelsup

optional, positive integer

Levelsup is an option that controls where to retrieve the roles - or how many steps to take up in the document tree nodes, to be exact. The allowed value is positive integer (≥ 1) and *default value is 0*. All the nested nodes are always visited when searching for roles.

Note

Module in detail

If the value for the `level sup` is very high, the endresult is the same as with `docwi de` option.

To get a better idea from the option, see the *rolelist example*. The usage is same with this directive.

siblings

optional, no value

controller option to take in the sibling nodes when looking for role entries. The option *has no value* and siblings are left out *by default*. Example:

To get a better idea from the option, see the *rolelist example*. The usage is same with this directive.

Note

Regxlist can find the roles only for the current document - i.e. the documents included by Sphinx `doctree` *are not covered*. However, the documents imported by the `include` directive are read.

If the generated list is empty, the translatable string for having "No entries" item is generated in the list.

Tip

Finding a suitable selector (using options like `level sup` and `siblings`) is sometimes a work of trial and error. This is due the node structure of the docutils.

In a case you don't get any results into list, try following:

1. Set `siblings` option to directive
2. Try increasing the number in `level sup` option

If the text continues to multiple rows, remember to put `\n` in regexp:

```
.. regxlist:: BUG: (?P<desc>(.\n)*)
: template: $desc
: siblings:
```

- * BUG: defect description that continues to multiple rows
- * BUG: single line entry

Module in detail

Regxlist module implements the `regxlist` directive, which creates a list of entries that matches with the regular expression rule, found from the document:

Fixes in this release:

```
.. regxlist:: Fixed #\d+:.*
```

```
: template:
```

Desc

rusty.exceltable -- ExcelTable directive

class `rusty.regxlist.RegXListNode(rawsource="", *children, **attributes)`

Custom placeholder node that is replaced in `process_regxlist`

Desc

class

`rusty.regxlist.RegXListDirective(name, arguments, options, content, lineno, content_offset, block_text, ...)`

Implements the directive. The class is registered as a directive in `rusty.regxlist.setup()`.

The structure of the directive:

```
.. regxlist:: regexp-rule
: template: {text} - {value}
: docwilde:
: levelsup: posit
: siblings:
```

Alternatively, the a list can be generated from a separate document (and not the document itself). This can be done by providing the `file` option.

Desc

`run()`

Implements the directive. In this case, it only stores the given argument and options into custom `RoleListNode` object - which are taken from there in `process_rolelist()`.

Desc

static `regxlist.process_regxlist(app, doctree)`

Iterates the document nodes once the doctree has been constructed and replaces the custom `regxlist` node with actual content (a bullet list of selected entries)

Desc

static `regxlist.setup(app)`

Extension setup, called by Sphinx

rusty.exceltable -- ExcelTable directive

Platforms: Unix, Windows

Versionmodified

Note

The module depends on `xlrd` -module. Install it using `setuptools`:

```
sudo easy_install xlrd
```

Usage	10
Example	12
Usage	14
Usage	19
Example	23
Usage	25
Usage	29
Example	30

Usage

Define excel table -directive into your document. The path to document is given with file option, and it is relative to RST -document path. The directive argument is reserved for the optional table caption.

Show part of the excel -document as a table within document:

```
.. excel table:: caption
   :file: path/to/document.xls
   :header: 1
   :selection: A1:B2
   :sheet: 1
```

See further information about the possible parameters from documentation.

Following options and arguments are available. Directive has no content:

caption (optional argument)

Optional table can be provided next to directive definition. If caption is not provided, no caption is set for the table.

```
.. excel table:: Caption for the table
   :file: document.xls
```

file (required)

Relative path (based on document) to excel -document. Compulsory option. Use forward slash also in Windows environments.

```
.. excel table::
   :file: path/to/document.xls
```

selection (optional)

Selection defines from and to the selection reaches. If value is not defined, the whole data from sheet is taken into table. Following definitions are supported:

- € Complete name selection: A1: B2
- € Starting name selection: C4:
- € Ending name selection: : C4 (selecting all the cells til C4)
- € Numeric selection: 0, 0: 2, 2 (indexing start from 0 and first value denotes the column, next row)

Note

- € If the selection is bigger than the actual data, the biggest possible field (row and/or column) is taken
- € On the numeric selection, the order of values is: col index, row index, making the complete selection to be:

```
start-c-idx, start-r-idx: end-c-idx, end-r-idx
```

Example

sheet (optional)

Defines the *name* or *index number* of the sheet. The index value is numeric and it starts from zero (0). The first sheet is also the default value if option is not defined. Examples:

```
.. excel table::  
   :file: document.xls  
   :sheet: SheetName  
  
.. excel table::  
   :file: document.xls  
   :sheet: 0
```

header (optional)

Header option can be used either for providing the header fields:

```
.. excel table::  
   :header: Name1, Name2, Name3  
   :file: document.xls
```

or as a numeric value, it defines the *number of rows* considered header fields in the data:

```
.. excel table::  
   :header: 1  
   :file: document.xls
```

The default value is 0, meaning no header is generated/considered to be found from data

widths

By default, the column widths are taken from the content (excel sheet): Directive counts relative sizes for the columns. However, it is also possible to define custom widths for the table:

```
.. excel table:: Automatic column widths  
   :file: document.xls  
   :header: A, B, C
```

```
.. excel table:: Manual column widths  
   :file: document.xls  
   :header: A, B, C  
   :widths: 20, 20, 60
```

Note

When defining the widths manually, remember following:

- € Separate the widths with comma (,)
- € The number of width values must match with the columns
- € The sum of the widths should be: 100

Example

This section shows few examples how the directive can be used and what are the options

Example

with it. For a reference, **see source Excel -document used with the examples** .

Directive definition:

```
.. excel table:: Cartoon listing
   :file: example/cartoons.xls
   :header: 1
```

Output of the processed document:

Cartoon listing

Title	Author	Since	Added
Garfield	Jim Davis	1978	2009-06-21
Get Fuzzy	Darby Conley	1999	2009-06-21
The Incredible Hulk	Stan Lee and Larry Lieber	1979-1982	2009-06-21

Selection can be limited using selection option, we can take the sub-set of the data:

```
.. excel table:: Cartoon listing (subset)
   :file: example/cartoons.xls
   :header: 1
   :selection: A1:B3

.. excel table:: Only entry dates
   :file: example/cartoons.xls
   :header: 1
   :selection: D1:
```

Output of the processed document:

Cartoon listing

Title	Author
Garfield	Jim Davis
Get Fuzzy	Darby Conley

Only entry dates

Added
2009-06-21
2009-06-21
2009-06-21

The sheet can be selected by using sheet -option. The value can be either the *name of the sheet* or the *numeric index of the sheet*, starting from zero (0,1,2...):

```
.. excel table:: Sheet example
   :file: example/cartoons.xls
```

Module in detail

```
:sheet: 1
:selection: B2: C3
```

Output of the processed document:

Sheet example

Module in detail

This section provides some further information about internals of the module:

XMLTableDirective implements the excel table -directive.

Desc

class

rusty.excel table.ExcelTableDirective(*name,arguments,options,content,lineno,content_offset,block_* ExcelTableDirective implements the directive. Directive allows to create RST tables from the contents of the Excel sheet. The functionality is very similar to csv-table (docutils) and xmltable (rusty.xml table).

Example of the directive:

```
.. excel table :
   :file: path/to/document.xls
   :header: 1
```

Desc

class rusty.excel table.ExcelTable(*fobj,encoding='utf-8'*)

Class generates the list based table from the given excel-document, suitable for the directive.

Class also implements the custom query format, is to use for the directive.:

```
>>> import os
>>> from rusty import excel table
>>>
>>> fo = open(os.path.join(os.path.dirname(excel table.__file__),'../doc/example/cartoons.xls'),'r+b')
>>> et = excel table.ExcelTable(fo)
>>>
>>> table = et.create_table(fromcell='A1', tocell='C4')
>>> assert et.fromcell == (0, 0)
>>> assert et.tocell == (2, 3)
>>>
>>> table = et.create_table(fromcell='B10', tocell='B11', sheet='big')
>>> assert et.fromcell == (1, 9)
>>> assert et.tocell == (1, 10)
```

Desc

ExcelTable.create_table(*fromcell=None,tocell=None,nheader=0,sheet=0*)

Creates a table (as a list) based on given query and columns

fromcell:

The index of the cell where to begin. The default is from the beginning of the data set (0, 0).

tocell:

The index of the cell where to end the selection. Default is in the end of the data set.

nheader:

Number of lines which are considered as a header lines. Normally, the value is 0 (default) or 1.

sheet:

Module in detail

Name or index of the sheet as string/unicode. The index starts from the 0 and is the default value. If numeric value is given, provide it in format:

```
et.create_table(fromcell='A1', tocell='B2', sheet='2')
```

Index

B

BeautifulTable (class in [rusty.xmltable](#))

Building

C

[create_table\(\)](#) ([rusty.exceltable.ExcelTable](#) method)

D

directive

E

[ExcelTable](#) (class in [rusty.exceltable](#))

[exceltable](#) (module)

[ExcelTableDirective](#) (class in [rusty.exceltable](#))

I

[includesh](#) (module)

[IncludeShellDirective](#) (class in [rusty.includesh](#))

Installation

P

[process_regxlist\(\)](#) ([rusty.regxlist](#) static method)

[process_rolelist\(\)](#) ([rusty.rolelist](#) static method)

R

regexp

[regxlist](#) (module)

[RegXListDirective](#) (class in [rusty.regxlist](#))

[RegXListNode](#) (class in [rusty.regxlist](#))

role

[rolelist](#) (module)

[RoleListDirective](#) (class in [rusty.rolelist](#))

[RoleListNode](#) (class in [rusty.rolelist](#))

[run\(\)](#) ([rusty.regxlist.RegXListDirective](#) method)

([rusty.rolelist.RoleListDirective](#) method)

[rusty.exceltable](#) (module)

[rusty.includesh](#) (module)

[rusty.regxlist](#) (module)

[rusty.rolelist](#) (module)

[rusty.xmltable](#) (module)

S

[setup\(\)](#) ([rusty.regxlist](#) static method)
([rusty.rolelist](#) static method)

[ShellConverter](#) (class in [rusty.includesh](#))

T

Testing

X

[xmltable](#) (module)

[XMLTableDirective](#) (class in [rusty.xmltable](#))

Global Module Index

E

``exceltable <#module-exceltable>`_*(Unix Windows)*`, *Generate RST tables by from XML documents*

I

``includesh <#module-includesh>`_*(Unix Windows)*`, *Easy including of shell script into documentation*

R

``regxlist <#module-regxlist>`_*(Unix Windows)*`, *Creates a list from current RST document entries matching with the regular expression rule*

``rolelist <#module-rolelist>`_*(Unix Windows)*`, *List RST document roles in a bullet list*

[rusty.exceltable](#)

[rusty.includesh](#)

[rusty.regxlist](#)

[rusty.rolelist](#)

[rusty.xmltable](#)

X

``xmltable <#module-xmltable>`_*(Unix Windows)*`, *Generate RST tables by from XML documents*