operation platform Documentation

Release lastest

golden

11月 29, 2017

Contents

CHAPTER 1

sphinx 使用说明

1.1 Sphinx 文档

1.1.1 前言

工作越来越久，就越来越觉得写文档是非常重要的，用过几种方式写文档:

- 做过blog，用过WordPress，也自己用python写过一个，维护成本太高了，工作忙的时候完全管不过来，离线的时候不能写。
- 各种云笔记，包括映像笔记，有道笔记，他们总有一定的局限性，在线才能同步文档，并且想导出到其他格式或是其他地方也很不方便，写出来的东西太离散，不能很好的组织目录。
- gitbook，个人觉得功能不行，markdown格式虽然简单易用易懂。但是对pdf排版特别不好。
- sphinx + reStructuredText 这个正在体验，等我写完这个说不定就喜欢上它了。

1.1.2 安装 Sphinx

1. 安装Python 环境,windows下需要安装，其他系统都会自带。
2. 安装sphinx

```
pip install sphinx
```

3. 执行 sphinx-quickstart 初始化目录

```
Welcome to the Sphinx 1.5.1 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:
```

```
You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name: operation platform
> Author name(s): golden

Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version []: 0.1
> Project release [0.1]:

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
http://sphinx-doc.org/config.html#confval-language.
> Project language [en]: zh

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]: y

Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/n) [n]: y
> doctest: automatically test code snippets in doctest blocks (y/n) [n]: y
> intersphinx: link between Sphinx documentation of different projects (y/n) [n]:␣
↪y
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: y
> coverage: checks for documentation coverage (y/n) [n]: y
> imgmath: include math, rendered as PNG or SVG images (y/n) [n]: y
> mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y
Note: imgmath and mathjax cannot be enabled at the same time.
imgmath has been deselected.
> ifconfig: conditional inclusion of content based on config values (y/n) [n]: y
> viewcode: include links to the source code of documented Python objects (y/n)␣
↪[n]: y
```

```
> githubpages: create .nojekyll file to publish the document on GitHub pages (y/
↪n) [n]: y

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]: y
> Create Windows command file? (y/n) [y]: y

Creating file .\source\conf.py.
Creating file .\source\index.rst.
Creating file .\Makefile.
Creating file .\make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file .\source\index.rst and create other␣
↪documentation
source files. Use the Makefile to build the docs, like so:
   make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.
```

4. 初始化之后的文档目录:

```
-|--source
    |--_static
    |--_templates
    conf.py
    index.rst
 |--build
 |--make.bat
 |--Makefile
```

5. 执行make.bat 可以把文档编译成各种格式。

```
Sphinx v1.6.3
Please use `make target' where target is one of
html        to make standalone HTML files
dirhtml     to make HTML files named index.html in directories
singlehtml  to make a single large HTML file
pickle      to make pickle files
json        to make JSON files
htmlhelp    to make HTML files and an HTML help project
qthelp      to make HTML files and a qthelp project
devhelp     to make HTML files and a Devhelp project
epub        to make an epub
latex       to make LaTeX files, you can set PAPER=a4 or PAPER=letter
text        to make text files
man         to make manual pages
texinfo     to make Texinfo files
gettext     to make PO message catalogs
changes     to make an overview of all changed/added/deprecated items
xml         to make Docutils-native XML files
pseudoxml   to make pseudoxml-XML files for display purposes
linkcheck   to check all external links for integrity
doctest     to run all doctests embedded in the documentation (if enabled)
coverage    to run coverage check of the documentation (if enabled)
```

1.1.3 配置

配置文件是放在source目录下面，名字叫 *conf.py*，是个python文件，主要配置是这样的。

```
version = 'lastest'
release = 'lastest'
import sphinx_rtd_theme
html_theme = 'sphinx_rtd_theme' # 主题设置
html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
```

1.2 reStructuredText 格式说明

reStructuredText是一种轻量级的文本标记语言，直译为：重构建的文本，为Python中Docutils项目的一部分。其一般保存的文件以.rst为后缀。在必要的时候，.rst文件可以被转化成PDF或者HTML格式，也可以有Sphinx转化为LaTex,man等格式，现在被广泛的用于程序的文档撰写。

1.2.1 标题等级

- # 及上划线表示部分，第一级
- * 及上划线表示章节，第二级
- = 及上划线表示小章节，第三级
- - 及上划线表示子章节，第四级
- ^ 及上划线表示子章节的子章节，第五级
- " 及上划线表示段落，第六级

1.2.2 段落

段落(ref)是reST文档中最基本的块。段落是由一个或多个空白行分离的简单的文本块。在Python中，缩进在reST中是具有重要意义，所以同一段落的所有行必须左对齐而且是同一级缩进。

1.2.3 内联标记

rest的内联标记非常丰富， reST 也允许自定义 “文本解释角色”', 这意味着可以以特定的方式解释文本. Sphinx以此方式提供语义标记及参考索引，操作符为 `:rolename:`content``.

Fig. 1.1: 自动标签

- `*text*` 是强调, 强调
- `**text**` 是重点强调, **重点强调**
- ``text`` 代码样式， `rm -f /`
- `:durole:`emphasis`` 也是强调 强调

- :durole:`strong` 也是重点强调 **重点强调**
- :durole:`literal` 也是代码样式， `代码样式`,
- :durole:`subscript` 这是下标 下标
- :durole:`superscript` 上标 上标
- :durole:`title-reference` 书、期刊等材料的标题 *书、期刊等材料的标题*
- :ref:`label-name` 引用,要先用 `.. _my-reference-label:` 定义唯一引用名,在标题前引用显示的标题名， 如 内联标记
- `.. _my-figure:..  figure::  whatever` 这是显示名字 引用 自动标签
- :doc:`../people` 链接到文档 *Sphinx* 文档
- :download:`this example script <../example.py>` 也是强调 这是一张图片
- :envvar:`A=B` 环境变量 , A=B
- :token:`ADFASDFASDFASDFASDF` 语法名子 , ADFASDFASDFASDFASDF
- :abbr:`LIFO (last-in, first-out)` 缩写 , LIFO (last-in, first-out)
- :command:`rm` 系统级命令 , **rm**
- :dfn:`rm` 在文本中标记术语定义. (不产生索引条目) , *rm*
- :file:`plus.png` 文件 , plus.png
- :kbd:`Control-x Control-f` 标记键值序列 , Control-x Control-f
- :mailheader:`Content-Type` RFC 822-样式邮件头的名字 , *Content-Type*
- :samp:`print 1+{variable}` 一块字面量文本 , print 1+*variable*
- :regexp:`rm` 正则表达式 , [1-9]
- :pep:`1#anchor` 对Python Enhancement Proposal 的参考. 会产生适当的索引条目及文本 “PEP number” ; , **PEP 1#anchor**
- :rfc:`1#anchor` Internet Request for Comments的参考. 也会产生索引条目及文本 “RFC number” ; 在HTML文档里是一个超链接 , **RFC 1#anchor**
- |today| 今天的日期 11月 29, 2017
- |version| 被项目文档的版本替换. lastest
- |release| 被项目文档的发布版本替换. lastest

星号及反引号在文本中容易与内联标记符号混淆，可使用反斜杠符号转义. 标记需注意的一些限制:

- 不能相互嵌套,
- 内容前后不能由空白: 这样写‘‘* text*‘‘ 是错误的,
- 如果内容需要特殊字符分隔. 使用反斜杠转义，如: thisis*one*word.

1.2.4 超链接

- 外部链接 使用 `链接文本 <http://example.com/>`_ 可以插入网页链接. 链接文本是网址，则不需要特别标记，分析器会自动发现文本里的链接或邮件地址.如 百度
- 内部链接 详见 自动标签

1.2.5 列表与引用

- * 开始的列表
- 是这样的

1. **1. 这样开始的列表** 这是说明
2. **是这样的** 这是说明

1. 这是嵌套
2. 列表
3. 第三项

1. #. 开始的是有序列表
2. 是这样的
3. 这样的

term (up to a line of text) Definition of the term, which must be indented

and can even consist of multiple paragraphs

next term

Description.

[1] is a reference to footnote 1, and[2] is a reference to footnote 2.

[3] is a reference to footnote 3.

1.2.6 表格

- 这是比较复杂的表格

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	...	...	

- 还有一种简单的表格

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

- 另一种简单的表格

Table 1.1: Truth table for “not”

A	not A
False	True
True	False

[1] This is footnote 1.
[2] This is footnote 2.
[3] This is footnote 3.

- 列表形式的表格

Table 1.2: Frozen Delights!

Treat	Quantity	Description
Albatross	2.99	On a stick!
Crunchy Frog	1.49	If we took the bones out, it wouldn't be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

- CSV 表格

Table 1.3: Frozen Delights!

Treat	Quantity	Description
Albatross	2.99	On a stick!
Crunchy Frog	1.49	If we took the bones out, it wouldn't be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

1.2.7 块

块在reStructuredText中的表现方式也有好几种，但是最常见的是文字块(Literal Blocks)。这种块的表达非常简单，就是在前面内容结束之后，用两个冒号" :: "(空格[Optional]，冒号，冒号）来分割，并在之后紧接着插入空行，而后放入块的内容，块内容要相对之前的内容有缩进。

这就是一个块:

```
for i in [1,2,3,4,5]:
  print i

就算空行也不能截断
```

这是一个普通快.

```
>>> print 'this is a Doctest block'
this is a Doctest block
```

这是一个文字块:

```
>>> This is not recognized as a doctest block by
reStructuredText.  It *will* be recognized by the doctest
module, though!
```

1.2.8 指令

指令或者标识符是一个通用的显式标记块。除了roles，指令或者标识符是reST的扩展机制，Sphinx大量地使用了它。使用都是 `..` 指令`::` 使用

支持如下指令:

- 警告： 支持
  - attention
  - caution
  - danger

  - error
  - hint
  - important
  - note
  - tip
  - warning
  - admonition， 如：

Danger: Beware killer rabbits!

Note: Beware killer rabbits!

- 图片：
  - images 普通图片
  - figure 带标题和可选图例的图片， 如：

- 特色表格 详见 表格
- 特色指令
  - raw 包括原生格式标记
  - include 在Sphinx中，当给定一个绝对的文件路径，该指令（标识符）将其作为相对于源目录来处理
  - class class属性赋给下一个元素

class `special`
  This is a “special” paragraph.

- HTML 特性
  - meta 生成HTML <meta> 标签
  - title 覆盖文件的标题
- 其他内容元素
  - contents 一个局部的，即只对当前文件的，内容表
  - container 具有特定类的容器，用于HTML 生成 div
  - rubir 一个与文件章节无关的标题
  - topic, sidebar 特别强调了内容元素
  - parsed-literal 支持行内标记的文字块
  - epigraph 带有属性行的块引用

- highlights, pull-quote 带自己的类属性的块引用
- compound 组合段落

如:

Topic Title

Subsequent indented lines comprise the body of the topic, and are interpreted as body elements.

Sidebar Title

Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

Lend us a couple of bob till Thursday.
I'm absolutely skint.
But I'm expecting a postal order and I can pay you back
as soon as it comes.
Love, Ewan.

```
def my_function():
    "just a test python code"
    print 8/2
```

$$_t(i) = P(O_1, O_2, \ldots O_t, q_t = S_i)$$

The 'rm' command is very dangerous. If you are logged in as root and enter

```
cd /
rm -rf *
```

you will erase the entire contents of your file system.

1.2.9 脚注

可以使用 `[#name]_` 标注在脚注的位置，在文档的最后的 `.. rubric:: Footnotes` 后添加脚注的内容，像这样:

Lorem ipsum[4] dolor sit amet ...[5]

[4] Text of the first footnote.
[5] Text of the second footnote.

1.3 Sphinx 标记结构

1.3.1 TOC树

由于reST不便于多个文件相互关联或者分割文件到多个输出文件中，Sphinx通过使用自定义的指令（标识符）来处理构成文档的单个文件的关系，这同样使用与内容表。toctree 指令（标识符）就是核心要素。

`.. toctree::` 该指令（标识符）使用在指令（标识符）主体中给出的文件作为单个TOCs(包含"sub-TOC树")，在当前位置插入一个"TOC树"

如下:

```
.. toctree::
 :maxdepth: 2

 intro
 strings
 datatypes
 numeric
 (many more documents listed here)
```

toctree 有几个选项：

- maxdepth 文件的内容表被加入的最大的深度
- numbered 开启章节编号
- titlesonly 仅显示在树中的文件的标题，而不是其他的同级别的标题
- glob 所有的条目都将进行匹配而不是直接按照列出的条目，匹配的结果将会按照字母表顺序插入

1.3.2 特殊名称

1. genindex 总索引
2. modindex python 模块索引
3. search 搜索页
4. 不要创建以 _ 开头的文件或目录

1.3.3 显示代码块

默认的你可以用 `::` 显示代码块，带上没有高亮. sphinx 代码高亮用的pygments模块。

- `.. highlight::  language`
- `.. code-block::  language`

都可以用来显示代码块，但是不知道为什么 highlight会报错：Error in "highlight" directive 支持高亮的语言有（pygments支持的）：

- none 没有高亮
- python
- guess 猜
- rest

- c
- ... 其他pygments支持的语言

如

Listing 1.1: this.py

```
def test_error(self, msg):
  print ""self""
  raise Exception(Exception('123'))
  return a
```

行号支持

- highlight 使用 *linenothreshold*，超过设置的行数将显示行号
- code-block 使用 *linenos*，显示行号
- code-block 使用 *emphasize-lines* 给的行号高亮
- `.. literalinclude::  filename` 显示文件代码
  - language 设置语言
  - emphasize-lines 高亮行号
  - linenos 显示行号
  - encoding 编码
  - pyobject 只包含特定的对象，如Timer.start
  - lines 包含行号
  - diff 对比
  - dedent 缩进

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#
# operation platform documentation build configuration file, created by
# sphinx-quickstart on Tue Aug  1 12:57:18 2017.
#
# This file is execfile()d with the current directory set to its
# containing dir.
#
# Note that not all possible configuration values are present in this
# autogenerated file.
#
# All configuration values have a default; values that are commented out
# serve to show the default.

# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
#
# import os
# import sys
# sys.path.insert(0, os.path.abspath('.'))


# -- General configuration ------------------------------------------------
```

```

# If your documentation needs a minimal Sphinx version, state it here.
#
# needs_sphinx = '1.0'

# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = ['sphinx.ext.autodoc',
              'sphinx.ext.doctest',
              'sphinx.ext.intersphinx',
              'sphinx.ext.todo',
              'sphinx.ext.coverage',
              'sphinx.ext.mathjax',
              'sphinx.ext.ifconfig',
              'sphinx.ext.viewcode',
              'sphinx.ext.githubpages',
              'sphinx.ext.graphviz']

# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']

# The suffix(es) of source filenames.
# You can specify multiple suffix as a list of string:
#
# source_suffix = ['.rst', '.md']
source_suffix = '.rst'

# The master toctree document.
master_doc = 'index'

# General information about the project.
project = u"golden's 文档笔记"
copyright = '2017, golden'
author = 'golden'

# The version info for the project you're documenting, acts as replacement for
# |version| and |release|, also used in various other places throughout the
# built documents.
#
# The short X.Y version.
version = 'lastest'
# The full version, including alpha/beta/rc tags.
release = 'lastest'

# The language for content autogenerated by Sphinx. Refer to documentation
# for a list of supported languages.
#
# This is also used if you do content translation via gettext catalogs.
# Usually you set "language" from the command line for these cases.
language = "zh_CN"
html_search_language = 'zh_CN'
source_encoding = 'UTF-8'
locale_dirs = ['locales', './locale']
# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# This patterns also effect to html_static_path and html_extra_path
exclude_patterns = []
```

```
84
85  # The name of the Pygments (syntax highlighting) style to use.
86  pygments_style = 'sphinx'
87
88  # If true, `todo` and `todoList` produce output, else they produce nothing.
89  todo_include_todos = True
90
91  # -- Options for HTML output ----------------------------------------------
92
93  # The theme to use for HTML and HTML Help pages.  See the documentation for
94  # a list of builtin themes.
95  #
96  html_theme = 'sphinx_rtd_theme'
97  html_theme_path = ['_themes']
98  html_theme_options = {
99      'collapse_navigation': True,
100     'display_version': True,
101     'navigation_depth': 3,
102 }
103 html_show_sourcelink = True
104 # Theme options are theme-specific and customize the look and feel of a theme
105 # further.  For a list of options available for each theme, see the
106 # documentation.
107 #
108 # html_theme_options = {}
109
110 # Add any paths that contain custom static files (such as style sheets) here,
111 # relative to this directory. They are copied after the builtin static files,
112 # so a file named "default.css" will overwrite the builtin "default.css".
113 html_static_path = ['_static']
114
115 # -- Options for HTMLHelp output ------------------------------------------
116
117 # Output file base name for HTML help builder.
118 htmlhelp_basename = 'operationplatformdoc'
119
120 # -- Options for LaTeX output ---------------------------------------------
121
122 latex_elements = {
123     # The paper size ('letterpaper' or 'a4paper').
124     #
125     # 'papersize': 'letterpaper',
126
127     # The font size ('10pt', '11pt' or '12pt').
128     #
129     # 'pointsize': '10pt',
130
131     # Additional stuff for the LaTeX preamble.
132     #
133     # 'preamble': '',
134
135     # Latex figure (float) alignment
136     #
137     # 'figure_align': 'htbp',
138     'preamble': '''
139 \\hypersetup{unicode=true}
140 \\usepackage{CJKutf8}
141 \\AtBeginDocument{\\begin{CJK}{UTF8}{gbsn}}
```

```
\\AtEndDocument{\\end{CJK}}
''',
}

# Grouping the document tree into LaTeX files. List of tuples
# (source start file, target name, title,
#  author, documentclass [howto, manual, or own class]).
latex_documents = [
    (master_doc, 'operationplatform.tex', 'operation platform Documentation',
     'golden', 'manual'),
]

# -- Options for manual page output ---------------------------------------

# One entry per manual page. List of tuples
# (source start file, name, description, authors, manual section).
man_pages = [
    (master_doc, 'operationplatform', 'operation platform Documentation',
     [author], 1)
]

# -- Options for Texinfo output -------------------------------------------

# Grouping the document tree into Texinfo files. List of tuples
# (source start file, target name, title, author,
#  dir menu entry, description, category)
texinfo_documents = [
    (master_doc, 'operationplatform', 'operation platform Documentation',
     author, 'operationplatform', 'One line description of project.',
     'Miscellaneous'),
]

# -- Options for Epub output ----------------------------------------------

# Bibliographic Dublin Core info.
epub_title = project
epub_author = author
epub_publisher = author
epub_copyright = copyright

# The unique identifier of the text. This can be a ISBN number
# or the project homepage.
#
# epub_identifier = ''

# A unique identification for the text.
#
# epub_uid = ''

# A list of files that should not be packed into the epub file.
epub_exclude_files = ['search.html']

# Example configuration for intersphinx: refer to the Python standard library.
intersphinx_mapping = {'https://docs.python.org/': None}
```

```
200 ##################################################################################
201 #          auto-created readthedocs.org specific configuration                   #
202 ##################################################################################
203
204
205 #
206 # The following code was added during an automated build on readthedocs.org
207 # It is auto created and injected for every build. The result is based on the
208 # conf.py.tmpl file found in the readthedocs.org codebase:
209 # https://github.com/rtfd/readthedocs.org/blob/master/readthedocs/doc_builder/
    ↪templates/doc_builder/conf.py.tmpl
210 #
211
212
213 import sys
214 import os.path
215 from six import string_types
216
217 from sphinx import version_info
218
219 # Get suffix for proper linking to GitHub
220 # This is deprecated in Sphinx 1.3+,
221 # as each page can have its own suffix
222 if globals().get('source_suffix', False):
223     if isinstance(source_suffix, string_types):
224         SUFFIX = source_suffix
225     else:
226         SUFFIX = source_suffix[0]
227 else:
228     SUFFIX = '.rst'
229
230 # Add RTD Static Path. Add to the end because it overwrites previous files.
231 if not 'html_static_path' in globals():
232     html_static_path = []
233 if os.path.exists('_static'):
234     html_static_path.append('_static')
235 html_static_path.append('/home/docs/checkouts/readthedocs.org/readthedocs/templates/
    ↪sphinx/_static')
236
237 # Add RTD Theme only if they aren't overriding it already
238 using_rtd_theme = False
239 if 'html_theme' in globals():
240     if html_theme in ['default']:
241         # Allow people to bail with a hack of having an html_style
242         if not 'html_style' in globals():
243             import sphinx_rtd_theme
244             html_theme = 'sphinx_rtd_theme'
245             html_style = None
246             html_theme_options = {}
247             if 'html_theme_path' in globals():
248                 html_theme_path.append(sphinx_rtd_theme.get_html_theme_path())
249             else:
250                 html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
251
252             using_rtd_theme = True
253 else:
254     import sphinx_rtd_theme
255     html_theme = 'sphinx_rtd_theme'
```

```
    html_style = None
    html_theme_options = {}
    if 'html_theme_path' in globals():
        html_theme_path.append(sphinx_rtd_theme.get_html_theme_path())
    else:
        html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
    using_rtd_theme = True

if globals().get('websupport2_base_url', False):
    websupport2_base_url = 'https://readthedocs.org/websupport'
    if 'http' not in settings.MEDIA_URL:
        websupport2_static_url = 'https://media.readthedocs.org/static/'
    else:
        websupport2_static_url = 'https://media.readthedocs.org//static'


#Add project information to the template context.
context = {
    'using_theme': using_rtd_theme,
    'html_theme': html_theme,
    'current_version': "latest",
    'MEDIA_URL': "https://media.readthedocs.org/",
    'PRODUCTION_DOMAIN': "readthedocs.org",
    'versions': [
    ("latest", "/zh/latest/"),
    ],
    'downloads': [
    ("pdf", "//readthedocs.org/projects/golden-note/downloads/pdf/latest/"),
    ("htmlzip", "//readthedocs.org/projects/golden-note/downloads/htmlzip/latest/"),
    ("epub", "//readthedocs.org/projects/golden-note/downloads/epub/latest/"),
    ],
    'subprojects': [
    ],
    'slug': 'golden-note',
    'name': u'golden-note',
    'rtd_language': u'zh',
    'canonical_url': 'http://golden-note.readthedocs.io/zh/latest/',
    'analytics_code': 'None',
    'single_version': False,
    'conf_py_path': '/source/',
    'api_host': 'https://readthedocs.org',
    'github_user': 'goodking-bq',
    'github_repo': 'golden-note',
    'github_version': 'master',
    'display_github': True,
    'bitbucket_user': 'None',
    'bitbucket_repo': 'None',
    'bitbucket_version': 'master',
    'display_bitbucket': False,
    'READTHEDOCS': True,
    'using_theme': (html_theme == "default"),
    'new_theme': (html_theme == "sphinx_rtd_theme"),
    'source_suffix': SUFFIX,
    'user_analytics_code': '',
    'global_analytics_code': 'UA-17997319-1',

    'commit': '6f843eb6',

```

```
}



if 'html_context' in globals():
    html_context.update(context)
else:
    html_context = context

# Add custom RTD extension
if 'extensions' in globals():
    extensions.append("readthedocs_ext.readthedocs")
else:
    extensions = ["readthedocs_ext.readthedocs"]
```

CHAPTER 2

Git 教程

CHAPTER 3

Python 语言，世界上最好的语言

3.1 Python 基础

3.1.1 Python 常用内建函数搜集

- **map**

  `map(function,iterable,...)` 对可迭代函数'iterable'中的每一个元素应用'function'方法，将结果作为list返回。

  如：

```
>>> map(lambda x:x*2,range(10))
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> map(lambda x,y: (y + x) * 2, range(10),range(10))
[0, 4, 8, 12, 16, 20, 24, 28, 32, 36]
```

- **filter**

  `filter(function or None, sequence)` 对指定序列执行过滤操作。用sequence里的每个元素去调用function，最后结果保护调用结果为True的元素,如果function参数为None，返回结果和sequence参数相同。

  例：

```
>>> filter(lambda x: x % 2 == 0, range(10))
[0, 2, 4, 6, 8]
```

- **reduce**

  `reduce(function, sequence[, initial])` 对参数序列中元素进行累积。function 必须要有两个参数。依次从sequence取一个元素，和上次调用的结果作为参数调用function。如果给了initial 参数，那么第一次调用就用initial 和sequence第一个元素作为参数。没有给就从sequence中去两个作为参数。

  例：

```
>>> reduce(lambda x,y: x * y,range(1,10))
362880
```

- **zip**

  `zip([iterable,...])` 接受一系列可迭代的对象作为参数，将对象中对应的元素打包成一个个tuple（元组），然后返回由这些tuples组成的list（列表）。若传入参数的长度不等，则返回list的长度和参数中长度最短的对象相同。利用*号操作符，可以将list unzip（解压）。

  例：

```
>>> a = ['a', 'b', 'c']
>>> b = [1, 2, 3, 4]
>>> c = ['~', '!', '@', '#', '$']
>>> zip(a, b, c)
[('a', 1, '~'), ('b', 2, '!'), ('c', 3, '@')]
>>> zip(*zip(a, b, c)) # 又还原了
[('a', 'b', 'c'), (1, 2, 3), ('~', '!', '@')]
```

- **hasattr**

  `hasattr(object,name)` 如果object对象具有与name字符串相匹配的属性，hasattr()函数返回真(true)；否则返回假(False)

  例：

```
>>> import os
>>> hasattr(os,'path')
True
>>> hasattr(os,'path1')
False
```

- **getattr**

  `getattr(object,name[,default])` 函数返回object的name属性值, 无属性会报错。

  例：

```
>>> import os
>>> getattr(os,'path')
<module 'ntpath' from 'C:\\Users\\Golden\\Anaconda3\\lib\\ntpath.py'>
>>> hasattr(os,'path1')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AttributeError: module 'os' has no attribute 'path1'
```

3.1.2 unittest 使用

Note: unittest 是python 最常用的单元测试框架.支持自动化测试,测试代码共享启动和关闭代码，集合测试以等。有几个概念:

1. test fixture: 开始测试前所作的工作，一般使用setUp()和tearDown()函数完成。
2. test case: 测试案例，最小的测试单元
3. test suite: 测试套件，测试案例的集合

4. test runner： 运行测试

命名方法:

- 测试类 以 `Test` 开头
- 测试方法 以 `test` 开头

测试要用到 `assert` 断言语法

python assert 断言是声明其布尔值必须为真的判定，如果发生异常就说明表达示为假。可以理解assert断言语句为raise-if-not，用来测试表示式，其返回值为假，就会触发异常。

---

test fixture

测试开始和结束的准备工作，一般用于创建和关闭数据库连接，开始或停止一个进程等, 如果是一系列测试需要相同的准备和结束工作，那建议写一个基类定义它们，具体测试在继承它

只需要定义下面两个函数:

- `setUp` 开始前的准备
- 和`tearDown` 结束后的操作

例:

`base_case.py`

Listing 3.1: base_case.py

```
class BaseTestCase(unittest.TestCase):
    def setUp(self):
        self.redis = redis.StrictRedis(host='192.168.137.3', db=6)

    def tearDown(self):
        self.redis.flushdb()  # 清空
```

test case

通过继承 `unittest.TestCase` 或是自己的 `unittest.TestCase` 基类，就可以创建测试单元，测试方法 都是以 “test“开头。

如:

Listing 3.2: test_case.py

```
# coding:utf-8
from __future__ import absolute_import, unicode_literals
import unittest
from .base_case import BaseTestCase

__author__ = "golden"
__date__ = '2017/8/11'


class TestString(BaseTestCase):
    def setUp(self):
        super(TestString, self).setUp()
        self.redis.set('test_string', 'python')
```

```

    def test_get_string_is_str(self):
        assert not isinstance(self.redis.get('test_string'), str)

    def test_get_string_is_bytes(self):
        assert isinstance(self.redis.get('test_string'), bytes)

    def test_get_string_value(self):
        assert self.redis.get('test_string') == 'python'


class TestHash(BaseTestCase):
    def setUp(self):
        super(TestHash, self).setUp()
        self.redis.hset('test_hash', 'key1', 'value1')
        self.redis.hset('test_hash', 'key2', 'value2')

    def test_get_all(self):
        res = self.redis.hgetall('test_hash')
        print(res)
        assert isinstance(res, dict)


if __name__ == '__main__':
    suite = unittest.TestLoader().loadTestsFromTestCase(TestString)
    unittest.main(defaultTest=suite)
```

最后目录结构为：

unit_test :

__init__.py
base_case.py
test_case.py
test_suite.py

要执行测试，可以在命令行执行命令：**python -m unittest unit_test.case** 得到下面结果:

```
> python -m unittest unit_test.test
{b'key1': b'value1', b'key2': b'value2'}
...F # 成功三个，失败一个
======================================================================
FAIL: test_get_string_value (unit_test.test.TestString)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\operation-platform\source\_static\unit_test\test.py", line 22, in test_get_
↪string_value
    assert self.redis.get('test_string') == 'python'
AssertionError

----------------------------------------------------------------------
Ran 4 tests in 0.016s

FAILED (failures=1) # 失败一个
```

test suite

测试套件是测试单元的集合，构建测试套件需要用到 `unittest.TestSuite()` 类,我们一般写成一个方法，如:

Listing 3.3: test_suite.py

```python
# coding:utf-8
from __future__ import absolute_import, unicode_literals
import unittest

__author__ = "golden"
__date__ = '2017/8/11'

from .test_case import *


def suite1():
    test_suite = unittest.TestSuite()
    test_suite.addTest(TestHash('test_get_all'))
    test_suite.addTest(TestString('test_get_string_is_str'))
    return test_suite


def suite2():
    tests = ['test_get_string_value', 'test_get_string_is_str']
    return unittest.TestSuite(map(TestString, tests))


def all():  # 多个套件还可以构成更大的套件
    _suite1 = suite1()
    _suite2 = suite2()
    return unittest.TestSuite([_suite1, _suite2])
```

运行命令 **`python -m unittest unit_test.test_suite.suite1`** 结果如下:

```
> python -m unittest unit_test.test_suite.suite1
{b'key2': b'value2', b'key1': b'value1'}
..
----------------------------------------------------------------------
Ran 2 tests in 0.009s # 只运行了两个测试

OK
```

跳过测试和预期的失败

由几个装饰器来控制:

- `@unittest.skip(reason)` 由什么原因跳过测试
- `@unittest.skipIf(condition, reason)` 满足 condition 条件时跳过
- `@unittest.skipUnless(condition, reason)` 不满足 条件时跳过
- `@unittest.expectedFailure` 如果失败，不包含在失败结果中
- `exception unittest.SkipTest(reason)` 报错错误
- 装饰器都可以用于方法或类

提供的其他测试方法

还有一些其他的测试方法，如:

- assertEqual(a, b) a == b
- assertNotEqual(a, b) a != b
- assertTrue(x) bool(x) is True
- assertFalse(x) bool(x) is False
- assertIs(a, b) a is b 3.1
- assertIsNot(a, b) a is not b 3.1
- assertIsNone(x) x is None 3.1
- assertIsNotNone(x) x is not None 3.1
- assertIn(a, b) a in b 3.1
- assertNotIn(a, b) a not in b 3.1
- assertIsInstance(a, b) isinstance(a, b) 3.2
- assertNotIsInstance(a, b) not isinstance(a, b) 3.2

详见 其他

生成 HTML 的测试报告

详见 HTMLTestRunner

3.1.3 struct 包

基本概念

Note: 对python基本类型值与用python字符串格式表示的C struct类型间的转化。

多用于存取文件，或是socket数据交换时使用。

它的类型对照表

格式	C 语言类型	python 类型	字节数
x	填充字节	no value	1
c	char	string of length 1	1
b	signed char	integer	1
B	unsigned char	integer	1
?	_Bool	bool	1
h	short	integer	2
H	unsigned short	integer	2
i	int	int	4
I	unsigned int	integer or long	4
l	long	integer	4
L	unsigned long	long	4
q	long long	long	8
Q	unsigned long long	long	8
f	float	float	4
d	double	float	8
s	char[]	string	1
p	char[]	string	1
P	void *	long	

Hint:

- 每个格式前面可以有个数字，表示个数
- s与p, s表示一定格式的字符串，但是p表示的是pascal字符串
- P用来转换一个指针，其长度和机器字长相关

对齐方式:

Character	Byte order	Size	Alignment
@(默认)	本机	本机	本机，凑够4个字节
=	本机	标准	none,按原字节数
<	小端	标准	none,按原字节数
>	大端	标准	none,按原字节数
!	network(大端)	标准	none,按原字节数

Hint:

- 小端：较高的有效字节存放在较高的存储器地址中，较低的有效字节存放在较低的存储器地址，符合计算机处理
- 大断：较低的有效字节存放在较高的存储器地址中，较高的有效字节存放在较低的存储器地址，符合人类正常思维逻辑

使用

- calcsize: 计算格式的字节长度

```
>>> struct.calcsize('>IH') # I 4个 H 2个 总共6个
6
```

- pack 和 unpack

pack 将python类型转换成C 二进制

unpack 则是反过来，将二进制转换成python类型

```
>>> struct.pack('<iHs',2,3,'e.w/'.encode())
b'\x02\x00\x00\x00\x03\x00e'
>>> struct.pack('<iH3s',22,3,'e.w/'.encode()) # 前面可以跟数字
b'\x16\x00\x00\x00\x03\x00e.w'
>>> struct.pack('<2i',22,3)
b'\x16\x00\x00\x00\x03\x00\x00\x00'
>>> struct.pack('>2i',22,3) # 大端 和小端 的区别
b'\x00\x00\x00\x16\x00\x00\x00\x03'
>>> struct.unpack('>2i',b'\x00\x00\x00\x16\x00\x00\x00\x03') #转换
成python数据
(22, 3)
```

- pack_into 和 unpack_from

3.2 Sqlalchemy 框架

3.2.1 Sqlalchemy 查询

sqlachemy的查询是非常强大，越是强大的东西越是复杂。 查询是通过 session 的 query() 实现。它可以接受任何数量的任何类和描述符的组合。

这里也只有一些常用的，需要更详细的，查看 官方文档 Query

query的 参数控制返回

- 如果跟的是模型类，那返回的就是这个这个类的实例或是列表。如:

```
session.query(User)
```

- 如果都给了表字段，那结果是元祖列表,如:

```
session.query(User.name, User.fullname)
```

- 如果同时给了模型类和表字段，那返回的是named tuples,如:

```
session.query(User, User.name)
```

- 你可以 label 方法控制列返回的名称。如:

```
session.query(User.name.label('name_label'))
```

- 你可以用aliased方法控制类返回的名称，如:

```
session.query(aliased(User, name='user_alias'))
```

- 可以用 limit,offset来控制返回的条数。如:

```
session.query(User).limit(10),
session.query(User).offset(1,20) # offset跟list的切片效果类似。
```

- 过滤数据使用 filter_by,filter

  - filter_by 参数为关键字参数，这种过滤功能有限 如:

```
filter_by(id=1)
```

  - filter 的参数是更像python的操作符，过滤功能很强大,如:

```
filter(User.id.in_([1,2])``
```

- 可以有多个过滤，也就是可以多个filter或是filter_by连着写。如:

```
session.query(User).filter(type=1).filter_by(User.id.in_([1,2,3]))
```

filter - 基本的操作符

数据过滤是通过filter来实现的，支持数据库里所有的操作符。

- 等于:

```
query.filter(User.name == 'ed')
```

- 不等于:

```
query.filter(User.name != 'ed')
```

- like:

```
query.filter(User.name.like('%ed'))
```

- in:

```
query.filter(User.id.in_([1,2,3]))
```

- not in:

```
query.filter(User.id.notin_([1,2,3]))
```

- is NULL:

```
query.filter(User.name == None)
query.filter(User.name.is_(None))
```

- is not NULL:

```
query.filter(User.name != None)
query.filter(User.name.isnot(None))
```

- and:

```
from sqlalchemy import and_

query.filter(and_(User.name == 'ed', User.fullname == 'Ed Jones'))
query.filter(User.name == 'ed', User.fullname == 'Ed Jones')
query.filter(User.name == 'ed').filter(User.fullname == 'Ed Jones')
```

- or:

```
from sqlalchemy import or_
query.filter(or_(User.name == 'ed', User.name == 'wendy'))
```

- match or contains:

```
query.filter(User.name.match('wendy'))
```

order_by - 排序

- 很简单的排序:

```
query.filter(User.name.match('wendy')).order_by(User.id)
query.filter(User.name.match('wendy')).order_by('id desc')
```

group_by - 分组

> **Hint:** SQL 的 group by 语句用于结合合计函数，根据一个或多个列对结果集进行分组。
> 多和统计函数一起使用，如 count（计数）,sum（求和）,avg（平均）

- 下面统计每个user_id 有多少个地址:

```
from sqlalchemy import func
query(Address.user_id, func.count('*')).group_by(Address.user_id)
```

- having 过滤统计数据，必须和 goup_by 一起使用，下面返回了user 地址大于1的user:

```
from sqlalchemy import func
query(Address.user_id, func.count('*')).group_by(Address.user_id).having(func.
↪count('*'))
```

text - 直接写sql

- 在text里写sql语句，并在 filter 和 order_by 中使用。看了下面几个例子就知道了:

```
from sqlalchemy import text
session.query(User).filter(text("id<224")).order_by(text("id")).all()
```

- text里可以用 :name 传动态参数，并params传值，如:

```
session.query(User).filter(text("id<:value and name=:name")). \
    params(value=224, name='fred').order_by(User.id).one()
```

- text里也可以给完整的sql语句,然后传给 from_statement 如下面这样匹配所有的列:

```
session.query(User).from_statement(text("SELECT * FROM user where name=:name")). \
    params(name='ed').all()
```

- 如果用from_statement中不是给的所有字段，那可用 columns 将值赋给字段，如:

```
stmt = text("SELECT name, id, fullname, password FROM users where name=:name")
stmt = stmt.columns(User.name, User.id, User.fullname, User.password)
session.query(User).from_statement(stmt).params(name='ed').all()
```

JOIN or OUTER JOIN - 更精简，效率更高

多张表联合查询的时候，可以这样写:

```
session.query(User, Address).filter(User.id==Address.user_id).\
                filter(Address.email_address=='jack@google.com').\
                all()
```

但是用 join 则更好

- 有外键关联:

```
session.query(User).join(Address).\
    filter(Address.email_address=='jack@google.com').all()
```

- 没有外键,则需要手动添加 join 关系:

```
session.query(User).join(Address,User.id==Address.user_id).\
    filter(Address.email_address=='jack@google.com').all()
```

Aliases - 别名

别名可以在这样的情况下使用:

```
from sqlalchemy.orm import aliased
adalias1 = aliased(Address) # 定义别名
adalias2 = aliased(Address) # 定义别名
for username, email1, email2 in \
    session.query(User.name, adalias1.email_address, adalias2.email_address).
↪\
    join(adalias1, User.addresses).\
    join(adalias2, User.addresses).\
    filter(adalias1.email_address=='jack@google.com').\
    filter(adalias2.email_address=='j25@yahoo.com'):
    print(username, email1, email2)
```

Subqueries - 子查询

要实现下面的sql:

```
SELECT users.*, adr_count.address_count FROM users LEFT OUTER JOIN
   (SELECT user_id, count(*) AS address_count
       FROM addresses GROUP BY user_id) AS adr_count
   ON users.id=adr_count.user_id
```

就需要用到子查询了:

```
from sqlalchemy.sql import func
stmt = session.query(Address.user_id, func.count('*').\
        label('address_count')).\
        group_by(Address.user_id).subquery()   #定义子查询
session.query(User, stmt.c.address_count).\
    outerjoin(stmt, User.id==stmt.c.user_id).order_by(User.id) # 这样使用
```

exists - 高效的子查询

Hint: EXISTS用于检查子查询是否至少会返回一行数据，该子查询实际上并不返回任何数据，而是返回值True或False

那怎么在 Sqlalchemy 写出 exists的 sql呢?

- 直接使用 `exists()` 方法:

```
from sqlalchemy.sql import exists
stmt = exists().where(Address.user_id==User.id)
session.query(User.name).filter(stmt)
```

- 使用 `any()` 方法，用于 **一对多/多对多** 关系，可在前面加 ~ 号表示 `not exists`:

```
session.query(User.name).filter(User.addresses.any(Address.email_address.like('
↪%google%')))
```

- 使用 `has()` 方法，用于 **多对一**，同样可在前面加 ~ 号表示 `not exists`:

```
session.query(Address).filter(~Address.user.has(User.name=='jack')).all()
```

- 使用 `contains()` 方法，用于 **一对多** 关系:

```
session.query.filter(User.addresses.contains(someaddress_object))
```

- 使用 `with_parent()` 方法，可用于 **任何关系**

  session.query(Address).with_parent(someuser, 'addresses')

subqueryload - 子查询加载

Hint: 当查询的表有关联的表时，它是关联的表的字段缓一步加载，也就是分两次查询一个query的数据，多和 first() limit() offset() order_by() 一起使用。

这对于数据量大的表来说很有用:

```
session.query(User).\
            options(subqueryload(User.addresses)).\
            filter_by(name='jack').one()
```

返回结果大小控制

- all() 返回所有

- first() 查询并返回第一条,没有数据为空
- one() 查询所有并严格返回一条数据，如果查询到多条数据或没有数据，都会报错
- one_or_none 同 one，没有数据会返回None，不会报错，其他一样。
- scalar 同 one，但是只返回那条数据的第一个字段。

3.2.2 表的继承

当我们有多个 *Model* 的结构都很相似的时候，我们就希望模型也能像python对象一样能够继承，这个Sqlalchemy 是完全支持的。

下面所有例子都是用 flask-sqlalchemy 来完成的。与纯粹的sqlalchemy差别不大

当然继承也有多种方式，完全能够满足需求。

Joined Table Inheritance

这种继承通过外键方式将基表和继承表相关联，这种继承的特点:

- 基类会在数据库建一张表，拥有基类的所有字段
- 继承的类也会在数据库建表，拥有继承的类的字段
- 继承类的数据会在两个表里面存放，它们的 ID 相同
- 继承类拥有基类的所有方法
- 查询基类会自动返回继承类的对象

具体的做法是:

- **基类的 __mapper_args__ 需要配置下面两个参数**
  - polymorphic_identity 表示是基类数据的类别，字符串就可以
  - polymorphic_on 类别字段的名称
- **继承的类需要配置__mapper_args__的参数**
  - polymorphic_identity 表示是数据的类别

一个例子:

```
class Animal(db.Model):
"""动物基类"""
__tablename__ = 'animal' # 会创建animal 表
id = db.Column(db.Integer, primary_key=True)
name = db.Column(db.String(255))
type = db.Column(db.String(20))
__mapper_args__ = {
    'polymorphic_identity': 'animal',
    'polymorphic_on': type
}


class Cat(Animal):
    """爬行动物"""
    __tablename__ = 'cat' # 会创建 cat 表
    id = db.Column(db.Integer, db.ForeignKey('animal.id'), primary_key=True)
    cat_name = db.Column(db.String(255))
```

```
    __mapper_args__ = {
        'polymorphic_identity': 'cat',
    }

class Dog(Animal):
    """爬行动物"""
    __tablename__ = 'dog' # 会创建dog 表
    id = db.Column(db.Integer, db.ForeignKey('animal.id'), primary_key=True)
    dog_name = db.Column(db.String(255))
    __mapper_args__ = {
        'polymorphic_identity': 'dog',
    }
```

测试数据:

```
>>> a=models.Animal()
>>> a.name='animal1'
>>> db.session.add(a)
>>> c=models.Cat()
>>> c.name='animal2'
>>> c.cat_name='cat1'
>>> db.session.add(c)
>>> d=models.Dog()
>>> d.cat_name='dog1'
>>> d.name='animal3'
>>> db.session.add(d)
>>> db.session.commit()
```

执行完之后，数据库的数据是这样的。

Table 3.1: Animal Table

id	name	type
1	animal1	animal
2	animal2	cat
3	animal3	dog

Table 3.2: Cat Table

id	cat_name
2	cat1

Table 3.3: Dog Table

id	dog_name
3	dog1

Single Table Inheritance

这种继承类是不会体现在具体表中，其特点:

- 只会创建基类表，一张表
- 拥有基类和继承类的所有字段

- 继承类拥有基类的所有字段

具体做法：

- **基类的 __mapper_args__ 需要配置下面两个参数**
  - polymorphic_identity 表示是基类数据的类别，字符串就可以
  - polymorphic_on 类别字段的名称
- **继承类需要配置__mapper_args__的参数：**
  - polymorphic_identity 表示是数据的类别
- 继承类不能加 __tablename__ 属性，否则会报错

例子：

```
class Animal(db.Model):
"""动物基类"""
__tablename__ = 'animal' # 会创建animal 表
id = db.Column(db.Integer, primary_key=True)
name = db.Column(db.String(255))
type = db.Column(db.String(20))
__mapper_args__ = {
    'polymorphic_identity': 'animal',
    'polymorphic_on': type
}


class Cat(Animal):
    """爬行动物"""
    cat_name = db.Column(db.String(255))
    __mapper_args__ = {
        'polymorphic_identity': 'cat',
    }

class Dog(Animal):
    """爬行动物"""
    dog_name = db.Column(db.String(255))
    __mapper_args__ = {
        'polymorphic_identity': 'dog',
    }
```

同样的测试语句：

```
>>> a=models.Animal()
>>> a.name='animal1'
>>> db.session.add(a)
>>> c=models.Cat()
>>> c.name='animal2'
>>> c.cat_name='cat1'
>>> db.session.add(c)
>>> d=models.Dog()
>>> d.cat_name='dog1'
>>> d.name='animal3'
>>> db.session.add(d)
>>> db.session.commit()
```

数据库的数据：

Table 3.4: Animal Table2

id	name	type	cat_name	dog_name
1	animal1	animal	NULL	NULL
2	animal2	cat	cat1	NULL
3	animal3	dog	NULL	dog1

Concrete Table Inheritance

这种继承只是语言上的继承，数据层不会有任何的关系，特点：

- 继承表会有基类的所有字段
- 基类的方法继承类不会继承
- 基类建表与否都没有关系
- 继承表之间也没有关系

具体做法：

1. *mapper.concrete* 基本继承

> **Warning:** 基表和继承表什么关系也没有
>
> - 查询基类的时候是不会查询到继承类的。
> - 基类的字段也不会继承，所有继承类是没有基类的字段，引用会报错。
> - 基表没有 __tablename__ 也会建表

- 基类不需要特殊设置
- 继承类需要在 __mapper_args__ 添加下面参数
  - concrete 设置为 `True` 说明是具体的，与基表没有具体的关系

一个例子：

```
class Animal(db.Model):
    """动物基类"""
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255))
    type = db.Column(db.String(20))

class Cat(Animal):
    """爬行动物"""
    __tablename__ = 'animal_cat'
    id = db.Column(db.Integer, primary_key=True)
    cat_name = db.Column(db.String(255))
    __mapper_args__ = {
        'concrete': True
    }

class Dog(Animal):
    __tablename__ = 'animal_dog'
    id = db.Column(db.Integer, primary_key=True)
    dog_name = db.Column(db.String(255))
```

```
    __mapper_args__ = {
        'concrete': True
    }
```

2. Polymorphic Loading - 多态加载继承

Hint: **多态** 意味着变量并不知道引用的对象是什么，根据引用对象的不同表现不同的行为方式。它在类的继承中得以实现，在类的方法调用中得以体现。

- *ConcreteBase* 具体类基类?

Warning: 之所以叫具体类，因为它会在数据库建表，对数据库来说是具体的。

同 *mapper.concrete* 的区别是:

- 查询基类会连带继承类一起查询，表之间的数据用union all连接起来
- 查询得到的对象是各自的对象

- 基类继承 ConcreteBase以及Base类
- 基类和继承类对的 __mapper_args__ 属性都需要添加下面内容
  - polymorphic_identity 类别区分
  - concrete 必须设置为 True

例子:

```
class Animal(ConcreteBase, db.Model):
    """动物基类"""
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255))

    @declared_attr
    def __mapper_args__(cls):
        return {'polymorphic_identity': cls.__name__.lower(),
                'concrete': True}

    def get_cat_name(self):
        return self.cat_name

    def get_dog_name(self):
        return self.dog_name


class Cat(Animal):
    """爬行动物"""
    __tablename__ = 'animal_cat'
    id = db.Column(db.Integer, primary_key=True)
    cat_name = db.Column(db.String(255))


class Dog(Animal):
    __tablename__ = 'animal_dog'
```

```
    id = db.Column(db.Integer, primary_key=True)
    dog_name = db.Column(db.String(255))
```

测试:

```
animal = Animal()
animal.name = 'animal1'
db.session.add(animal)
db.session.commit()
cat = Cat()
# cat.name = 'animal2' # 具体基类的字段不能被继承，不能被赋值
cat.cat_name = 'cat1'
db.session.add(cat)
db.session.commit()
dog = Dog()
# dog.name = 'animal2'
dog.dog_name = 'dog1'
db.session.add(dog)
db.session.commit()
animals = db.session.query(Animal).all()
cats = Cat.query.all()
dogs = Dog.query.all()
print(animals)
print(cats)
print(cats[0].get_cat_name())
print(dogs[0].get_dog_name())
print(dogs[0].get_cat_name()) # Dog 有这个方法，带上没有 cat_name属
性，所以报错。

[<monitor.models.test.Animal object at 0x000000000AFA1F28>,
↪<monitor.models.test.Cat object at 0x000000000E686080>,
↪<monitor.models.test.Dog object at 0x000000000E6865F8>]
[<monitor.models.test.Cat object at 0x000000000E686080>]
cat1
dog1

Error
Traceback (most recent call last):
  File "C:\Users\golden\Anaconda3\envs\flask\lib\unittest\case.
↪py", line 329, in run
    testMethod()
  File "D:\quleduo_manager\test\models.py", line 246, in test_
↪con
    print(dogs[0].get_cat_name())
  File "D:\quleduo_manager\monitor\models\test.py", line 23, in␣
↪get_cat_name
    return self.cat_name
AttributeError: 'Dog' object has no attribute 'cat_name'
```

3. Abstract Concrete Classes - 抽象具体类

   (a) 使用 AbstractConcreteBase 类

Warning:

- 基类默认建表，如果 __tablename__=None 则不建 ， 但是也不能查询
- 有字段会建表 但是官方说不会建表 好奇怪 。

- 继承类会继承所有方法和字段

- 基类继承 AbstractConcreteBase
- 继承类的 __mapper_args__ 需要下面参数
  - polymorphic_identity
  - concrete = True

官方给的例子:

```
from sqlalchemy.ext.declarative import AbstractConcreteBase

class Animal(AbstractConcreteBase, db.Model):
    """动物基类"""
    __tablename__ = None
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255))

    @declared_attr
    def __mapper_args__(cls):
        return {'polymorphic_identity': cls.__name__.lower(),
                'concrete': True} if cls.__name__ != "Animal" else {}

    def get_cat_name(self):
        return self.cat_name

    def get_dog_name(self):
        return self.dog_name


class Cat(Animal):
    """爬行动物"""
    __tablename__ = 'animal_cat'
    cat_name = db.Column(db.String(255))


class Dog(Animal):
    __tablename__ = 'animal_dog'
    dog_name = db.Column(db.String(255))


cat = Cat()
cat.name = 'animal2'
cat.cat_name = 'cat1'
db.session.add(cat)
db.session.commit()
dog = Dog()
dog.name = 'animal2'
dog.dog_name = 'dog1'
db.session.add(dog)
db.session.commit()
animals = db.session.query(Animal).all()
cats = Cat.query.all()
dogs = Dog.query.all()
print(animals)
print(cats)
print(cats[0].get_cat_name())
```

```
print(dogs[0].get_dog_name())
print(dogs[0].get_cat_name()) #报错

[<monitor.models.test.Dog object at 0x000000000E6A00F0>, <monitor.models.
↪test.Cat object at 0x000000000B0C4160>]
[<monitor.models.test.Cat object at 0x000000000B0C4160>]
cat1
dog1

Error
Traceback (most recent call last):
  File "C:\Users\golden\Anaconda3\envs\flask\lib\unittest\case.py", line␣
↪329, in run
    testMethod()
  File "D:\quleduo_manager\test\models.py", line 246, in test_con
    print(dogs[0].get_cat_name())
  File "D:\quleduo_manager\monitor\models\test.py", line 23, in get_cat_
↪name
    return self.cat_name
  File "C:\Users\golden\Anaconda3\envs\flask\lib\site-
↪packages\sqlalchemy\orm\attributes.py", line 293, in __get__
    return self.descriptor.__get__(instance, owner)
  File "C:\Users\golden\Anaconda3\envs\flask\lib\site-
↪packages\sqlalchemy\orm\descriptor_props.py", line 492, in __get__
    warn()
  File "C:\Users\golden\Anaconda3\envs\flask\lib\site-
↪packages\sqlalchemy\orm\descriptor_props.py", line 480, in warn
    (self.parent, self.key, self.parent))
AttributeError: Concrete Mapper|Dog|animal_dog does not implement␣
↪attribute 'cat_name' at the instance level.  Add this property␣
↪explicitly to Mapper|Dog|animal_dog.
```

3.2.3 Model 轻松转 dict 和 json

将sqlalchemy的模型数据转为dict 或是json

```
# coding:utf-8
from __future__ import absolute_import, unicode_literals
from flask_sqlalchemy import SQLAlchemy, Model, BaseQuery
from sqlalchemy.ext.declarative import DeclarativeMeta
import json


class JsonModel(Model):
    __exclude__ = ['id']  # to_dict 排除字段
    __include__ = []  # 包含字段
    __exclude_foreign__ = True  # 排除外键

    def dict(self):
        data = {}
        for field in self.__fields__():
            value = getattr(self, field)  # value
            if isinstance(value.__class__, DeclarativeMeta):
                data[field] = value.dict()
            elif not hasattr(value, '__func__') and not isinstance(value, BaseQuery):
                try:
```

```
                    data[field] = value
                except TypeError:
                    data[field] = None
        return data

    def json(self):
        data = {}
        for field in self.__fields__():
            value = getattr(self, field)  # value
            if isinstance(value.__class__, DeclarativeMeta):
                data[field] = value.dict()
            elif not hasattr(value, '__func__') and not isinstance(value, BaseQuery):
                try:
                    json.dumps(value)
                    data[field] = value
                except TypeError:
                    try:
                        data[field] = str(value)
                    except Exception as e:
                        data[field] = None
        return json.dumps(data)

    def __foreign_column__(self):
        data = []
        for column in self.__table__.columns:
            if getattr(column, 'foreign_keys'):
                data.append(column.key)
        return data

    def __fields__(self):
        fields = set(dir(self))
        if self.__exclude_foreign__:
            fields = fields - set(self.__foreign_column__())
        fields = fields - set(self.__exclude__)
        fields = set(list(fields) + self.__include__)
        return [f for f in fields if
                not f.startswith('_') and not f.endswith('_id') and f not in [
↪'metadata', 'query', 'query_class',
                                                                              'dict',
↪'json']]


db = SQLAlchemy(model_class=JsonModel)
```

3.3 Numpy - 科学计算包

3.3.1 什么是numpy

numpy(Numerical Python extensions)是一个第三方的Python包，用于科学计算。这个库的前身是1995年就开始开发的一个用于数组运算的库。经过了长时间的发展，基本上成了绝大部分Python科学计算的基础包，当然也包括所有提供Python接口的深度学习框架。

3.3.2 支持的数据类型

- `bool` 用一位存储的布尔类型（值为TRUE或FALSE）
- `inti` 由所在平台决定其精度的整数（一般为int32或int64）
- `int8` 整数，范围为 -128至127
- `int16` 整数，范围为 -2**16至32767
- `int32` 整数，范围为 -2**31至2**31-1
- `int64` 整数，范围为 -2**64至2**64-1
- `uint8` 无符号整数，范围为0至255
- `uint16` 无符号整数，范围为0至 65 535
- `uint32` 无符号整数，范围为0至 2**32-1
- `uint64` 无符号整数，范围为0至 2**64-1
- `float16` 半精度浮点数（16位）：其中用1位表示正负号，5位表示指数，10位表示尾数
- `float32` 单精度浮点数（32位）：其中用1位表示正负号，8位表示指数，23位表示尾数
- `float64` 或 `float` 双精度浮点数（64位）：其中用1位表示正负号，11位表示指数，52位表示尾数
- `complex64` 复数，分别用两个32位浮点数表示实部和虚部
- `complex128` 或 `complex` 复数，分别用两个64位浮点数表示实部和虚部

3.3.3 array 核心模块

Note: array - 由多个元素类型组成的数组。数组中所有元素的类型必须是相同的，要么是上面说的基本类型，要么是列表。 数组中有两个概念:

- axes(轴) 就是每个元素类型的长度
- rank(秩) 他是轴的个数，也叫组维度

如一个二维数组 array([[1,2,3],[3,4,5]]),他的秩 为2 ，轴为3,

创建数组

```
>>> import numpy as np
>>> a=np.array([1,2,3,4]) # 一维数组
>>> b=np.array([[1,2,3],['a','b','c']]) # 二维数组
>>> a.shape # shape属性只有一个元素，所以是一维数组
(4,)
>>> b.shape # shape属性有两个元素，所以是二维数组，0轴长度为2,1轴长度为3
(2, 3)
>>> np.arange(1,5,1) # 自动生成 开始，结束，步长
array([1, 2, 3, 4])
>>> np.linspace(1,5,5) # 等差数列 开始，结束，个数
array([ 1.,  2.,  3.,  4.,  5.])
>>> np.logspace(0,2,5) # 等比数列
array([   1.        ,    3.16227766,   10.        ,   31.6227766 ,  100.        ])
>>> np.empty((3,4),np.int) # 只分配空间，不初始化操作，速度最快。注意，没有初始化
```

```
array([[1739692720,         517, 1787770920,         517],
   [1787772000,         517, 1787543216,         517],
   [1787543088,         517, 1788022832,         517]])
>>> np.zeros((2,2)) # 初始化为0
array([[ 0.,  0.],
   [ 0.,  0.]])
>>> np.ones(2) #初始化 1
array([ 1.,  1.])
>>> np.fromstring(b'abcde',dtype=np.int8) # 从字符串生成 ， 取得ASCII编码值
array([ 97,  98,  99, 100, 101], dtype=int8)
>>> np.fromfunction(lambda x,y:x+y+1,(2,2)) # 从方法生成
array([[ 1.,  2.],
   [ 2.,  3.]])
```

存取元素

1. 一维数组

Note: 一维数组大致上和列表相同

```
>>> a=np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a[5] # 一维数组存取同  list
5
>>> a[1:-1:2] # 第三个参数表示步长
array([1, 3, 5, 7])
>>> a[::-1] # 步长1 负数表示顺序颠倒
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
>>> a[1,2]=100,200 #可以改变
array([  0, 100, 200,   3,   4,   5,   6,   7,   8,   9])
>>> b=a[3:6] # b与a使用相同的地址，改一个两个都改
>>> a[[2,4,-1]] #
array([200,   4,   9])
>>> x=np.arange(10,1,-1)
>>> x[np.array([3,3,1,8])] #array 取数据
array([7, 7, 9, 2])
>>> x[[3,3,1,8,3,3,3,3]].reshape(2,4)
array([[7, 7, 9, 2],
   [7, 7, 7, 7]])
```

2. 多维数据

Note: 多维数组需要两个点才能确定一个元素

```
>>> a=np.array([[1,2,3,4,5],[22,32,42,52,62],[33,43,53,63,73]])
>>> a[2,3] or a[(2,3)] # 第三列 第4个元素
63
>>> a[1:,[0,2,4]] # 1: 选取的是1行之后的所有行，[0,2,4] 选取的是行的第 0,2,4 个元
素
array([[22, 42, 62],
   [33, 53, 73]])
```

3. 结构数组

Note: numpy.dtype 很容易定义结构数组

```
>>> person=np.dtype({'names':['name','age'],'formats':['S30','i']},
↪align=True)
>>> persons=np.array([(b'zhang san',22),(b'li si',23),(b'wang er',44)],
↪dtype=person)
>>> persons.dtype # 类型
{'names':['name','age'], 'formats':['S30','<i4'], 'offsets':[0,32],
↪'itemsize':36, 'aligned':True}
>>> persons.shape # 这里为什么不是3,2
(3,)
>>> persons[2] # 原来是这样
(b'wang er', 44)
>>> persons[2]['name'] # 取姓名
b'wang er'
>>> persons.flags # 一些属性
  C_CONTIGUOUS : True # 数据存储区域是否是 C 语言格式的连续区域
  F_CONTIGUOUS : True # 数据存储区域是否是 Fortran 语言格式的连续区域
  OWNDATA : True # 数组是否拥有次数据存储区域，当一个数组是其他数组视图时为False
  WRITEABLE : True # 可写
  ALIGNED : True # 对齐
  UPDATEIFCOPY : False # 复制时更新
>>> persons.strides # 每个轴上相邻元素的地址差
(36,)
>>> persons.T # 转置
[(b'zhang san', 22) (b'li si', 23) (b'wang er', 44)] # 一维没变
>>> persons.T.flags
  C_CONTIGUOUS : True
  F_CONTIGUOUS : True
  OWNDATA : False #
  WRITEABLE : True
  ALIGNED : True
  UPDATEIFCOPY : False
```

ufunc 函数

ufunc 是 universal function 的缩写，他是一种对数组的每个元素进行运算的函数，都是用C所写，速度非常快。

1. 四则运算

Table 3.5: 四则运算对应的 ufunc 函数

四则运算表达式	对应的 ufunc 函数
y = x1 + x2	add(x1, x2)
y = x1 - x2	subtract(x1, x2)
y = x1 * x2	multiply(x1, x2)
y = x1 / x2	divide(x1, x2) , 如果是都是整数，用整数除法。
y = x1 / x2	true_divide(x1, x2) , 总是返回精确的商
y = x1 // x2	floor_divide(x1, x2) , 总是对返回值取整
y = -x1	negative(x1, x2)
y = x1 ** x2	power(x1, x2)
y = x1 % x2	remainder(x1, x2) , mode(x1, x2)

例:

```
>>> a=np.arange(1,20)
>>> b=np.arange(0,19)
>>> c=a+b
>>> a
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
↪ 18, 19])
>>> b
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
↪ 17, 18])
>>> c
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
↪ 35, 37])
```

2. 比较运算和布尔运算

Table 3.6: 比较运算与对应的 ufunc 函数

比较运算表达式	对应的 ufunc 函数
x1 == x2	equal(x1, x2)
x1 != x2	not_equal(x1, x2)
x1 < x2	less(x1, x2)
x1 <= x2	less_equal(x1, x2)
x1 > x2	greater(x1, x2)
x1 >= x2	greater_equal(x1, x2)
x1 and x2	logical_and(x1, x2) # x1 and x2 会报错
x1 or x2	logical_or(x1, x2)
not x2	logical_not(x2)
	any(x1) , x1 任何一个为True,返回True
	all(x1) , x1 全部为True,返回True
x1 & x2 , 按位与	between_and(x1, x2)
x1 | x2 , 按位或	between_or(x1, x2)
x1 ^ x2 , 按位亦或	between_xor(x1, x2)
~x2 , 按位非	between_not(x1)

例:

```
>>> ~np.arange(5)
array([-1, -2, -3, -4, -5], dtype=int32)
>>>~np.arange(5,dtype=np.uint8)
array([255, 254, 253, 252, 251], dtype=uint8)
```

3. 自定义 ufunc

   可以用 *frompyfunc()*, *vectorize()* 来把普通的对单个运算的方法转成 ufunc 方法。vectorize 可以通过otypes指定返回元素类型

   例：

```
import numpy as np

p_type = np.dtype({'names': ['name', 'age', 'sex'],
                   'formats': ['S30', 'i', 'S1']}, align=True)
a = np.array([('golden', 30, 'b'), ('gg', 20, 'g')], dtype=p_type)


def gender_cn(a):
    b = list(a)
    if a[2] == b'b':
        return 1
    elif a[2] == b'g':
        return 2
    else:
        return 0


f1 = np.frompyfunc(gender_cn, 1, 1)
f2 = np.vectorize(gender_cn, otypes=[np.bool])
f1(a) # np.array([1, 2])
f2(a) # np.array([True,True])
```

4. 广播

Tip: 当使用ufunc时，如果两个数组的形状不同，会做如下处理：

- 让所有输入数组都向其中维数最多的看齐，shape熟悉中不足的部分通过在前面加1补齐
- 输出输入的shape属性是输入数组的shape属性的各个轴上的最大值
- 如果输入数组的某个轴的长度为1或输出数组的对应轴长度相同，这个数组能够用来计算，否则报错。
- 当输入数组的某个轴的长度为1时，沿着此轴运算是用此轴上的第一组值

```
a = np.array([1,2,3,4]) # a.shape = (4,)
b = np.array([[1],[2],[3],[4],[5]]) # b.shape = (5, 1)
c = a + b
# c:  [[2 3 4 5]
#      [3 4 5 6]
#      [4 5 6 7]
#      [5 6 7 8]
#      [6 7 8 9]]
# c.shape = (5, 4)
# a + b 得到一个加法表，得到一个形状为 (5, 4) 数组
```

Note:

- 由于 a, b 的维数不同，根据规则，需要在让a的shape像b对齐，于是在a的shape前加1，变成 *(1, 4)*

- 这样 两个做加法运算的shape 属性为 *(1, 4), (5, 1)*, 根据规则，输出数组的shape是输入的各个轴上的最大值，所以结果形状是 (5, 4)
- 由于 a 的0轴长度为1，而b的0轴长度为5，所以需要将a的0轴长度扩展为5，相当于a.shape=1,4 a=a.repeat(5, axis=0),所以a最后变成了array([[1, 2, 3, 4], [1, 2, 3, 4],[1, 2, 3, 4],[1, 2, 3, 4],[1, 2, 3, 4]])
- 由于 b 的 1周长度为1，而a的1轴长度为4,为了相加，相当于 b=b.repeat(4,axis=1),变为array([[1, 1, 1, 1],[2, 2, 2, 2], [3, 3, 3, 3], [4, 4, 4, 4],[5, 5, 5, 5]])
- 最后相加得到结果。当然真正的过程不是这样，这样耗内存。

---

ogrid 专门用于创建广播运算的数组

x, y = np.ogrid[:5,:7] # x=array([[0],[1],[2],[3],[4]]) y=array([[0, 1, 2, 3, 4, 5, 6]])

mgrid 与 *ogrid* 类似，但是返回的是广播之后的数组

3.3.4 numpy 强大的函数库

随机函数

Table 3.7: 随机函数

函数名	功能	参数实例
rand	0~1之间的随机浮点数	rand(2,3)
randn	标准正太分布的随机数	rand(4,3)
randint	指定范围内的随机数	randint(2,5,(5,4)) # 范围和形状
normal	正态分布	normal(100,10,(5,4)) # 期望值 标准差 形状
uniform	均匀分布	uniform(10,20,(4,3)) #起始值 终止值 形状
poisson	泊松分布	poisson(2.0,(4,3)) # 系数
permutation	随机分布	permutation(10 or a) #返回的新数组
shuffle	随机打乱顺序	shuffle(a), # 将输入的数组顺序打乱
choice	随机抽取	choice(a,size=(3,3),p=a/np.sum(a)) #p 指定抽取元素的概率，表示越大，抽取概率也越大
seed	设置随机数种子	可以保证每次运行是得到相同的随机数

求和 平均值 方差

Table 3.8: 函数

函数名	功能	参数实例
sum	求和	sum(a,axis=1) # axis 对哪个轴求和，返回列表 keepdims参数指定是否保持原来的维数
mean	求期望	
average	加权平均数	
std	标准差	
var	方差	
product	连乘积	

大小和排序

Table 3.9: 大小和排序函数

函数名	功能	参数实例
min	最小值	
max	最大值	
minimum	二元最小值	
maximum	二元最大值	
ptp	最小值最大值之差	
argmin	最小值下标	
argmax	最大值下标	
unravel_index	一维下标转换成多维下标	
sort	数组排序	
argsort	计算数组排序的下标	
lexsort	多列排序	
partition	快速计算前K位	
argpartition	前k位下标	
media	中位数	
percentile	百分中位数	
searchsorted	二分查找	

统计函数

Table 3.10: 统计函数

函数名	功能	参数实例
unique	去除重复元素	
bincount	对整数数组的元素计数	
histogram	一维直方图统计	
digitze	离散化	

分段函数

Table 3.11: 分段函数

函数名	功能	参数实例
where	矢量化判断表达式	where(x>1,x*2,x*3) # 如果>1 ,x*2 否则 x*3
piece-wise	分段函数	piecewise(x,[x>1,x<1],[lambda x: x*x,lambda x: x*2,0]) # x>1=x*x x<1=x*2,else 0
select	多分支判断选择	

操作多维数组

Table 3.12: 操作多维数组的函数

函数名	功能	参数实例
concatenate	连接多个数组	
vstack	延0轴连接数组	
hstack	延1轴连接数组	
column_stack	按列连接多个一维数组	
split,array_split	将数组分为多段	
transpose	重新设置轴的顺序	
swapaxes	交换两个轴的顺序	

例：

```
>>> a=np.arange(3)
>>> a
array([0, 1, 2])
>>> b=np.arange(10,13)
>>> b
array([10, 11, 12])
>>> np.vstack((a,b))
array([[ 0,  1,  2],
       [10, 11, 12]])
>>> np.hstack((a,b))
array([ 0,  1,  2, 10, 11, 12])
>>> np.column_stack((a,b))
array([[ 0, 10],
       [ 1, 11],
       [ 2, 12]])
>>> c=np.random.randint(1,19,(1,2,3,4))
>>> c.shape
(1, 2, 3, 4)
>>> np.transpose(c,(2,1,0,3)).shape
(3, 2, 1, 4)
>>> np.swapaxes(c,2,3).shape
(1, 2, 4, 3)
```

3.4 Python 其他

3.4.1 Pip 管理 Python包

安装

直接使用 `easy_install pip` 就行了，想 pyenv 或是 anconda 环境都自动会装上

配置

配置文件放在:

- windows下： %USER HOME%\pip\pip.ini
- linux下: ~/.pip/pip.conf

如果想使用国内原，可以在配置文件里添加下面内容:

```
[global]
format = columns
index-url = http://pypi.douban.com/simple
trusted-host = pypi.douban.com
```

经常使用

- 命令

```
Usage:
  pip <command> [options]

Commands:
  install                     安装需要的包.
  download                    下载包.
  uninstall                   卸载.
  freeze                      生成当前环境的 requirements.
  list                        列出所有已安装的包.
  show                        某个包的详细信息.
  check                       验证已安装的包具有兼容的依赖性.
  search                      搜索某个包.
  wheel                       根据 requirements 创建wheel包.
  hash                        计算软件包档案的哈希.
  completion                  用于命令完成的一个助手命令.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --isolated                  Run pip in an isolated mode, ignoring environment␣
↪variables and user configuration.
  -v, --verbose               Give more output. Option is additive, and can be␣
↪used up to 3 times.
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output. Option is additive, and can be␣
↪used up to 3 times (corresponding to
                              WARNING, ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
```

```
  --proxy <proxy>              使用代理 [user:passwd@]proxy.server:port.
  --retries <retries>          Maximum number of retries each connection should␣
↪attempt (default 5 times).
  --timeout <sec>              Set the socket timeout (default 15 seconds).
  --exists-action <action>     Default action when a path already exists: (s)witch,
↪ (i)gnore, (w)ipe, (b)ackup,
                               (a)bort.
  --trusted-host <hostname>    Mark this host as trusted, even though it does not␣
↪have valid or any HTTPS.
  --cert <path>                Path to alternate CA bundle.
  --client-cert <path>         Path to SSL client certificate, a single file␣
↪containing the private key and the
                               certificate in PEM format.
  --cache-dir <dir>            Store the cache data in <dir>.
  --no-cache-dir               Disable the cache.
  --disable-pip-version-check
                               Don't periodically check PyPI to determine whether␣
↪a new version of pip is available for
                               download. Implied with --no-index.
```

- 关于 **pip install**
  - `pip install -U packages` 安装或升级包
  - `pip install git+<git url>` 从git上下载并安装包，git上都是最新的。
  - `pip install -U -r requirements.txt` 根据requirements安装包
  - `pip install package.whl` 安装whl包，由于windows编译环境难安装，所以可以从 python-libs 下载编译好的包安装,比如 mysql-python

3.4.2 anconda 一个好用 python 发行版

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

最好的是 他自带 `python` 多版本管理器，可以支持多 `python` 版本同时存在或切换，并且同时支持所有系统。 是python 版本管理利器。

安装

安装很简单，去 官网 下载就行没什么难度。

配置

它的配置文件:

- Windows 放在 C:\Users\username\.condarc
- Linux 放在 ~/.condarc
- 如果没有可自建

如果你想使用国内原 可以修改配置文件

```
channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
 - defaults
show_channel_urls: yes
```

或是执行命令:

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --set show_channel_urls yes
```

国内原下载非常快

使用

- 命令

```
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and␣
↪packages.

Options:

positional arguments:
  command
    info          显示当前conda环境的新信息.
    help          Displays a list of available conda commands and their help
                  strings.
    list          列出当前环境的所有包.
    search        搜索包.
    create        创建一个环境.
    install       当前环境安装包.
    update        更新包，也可以更新当前环境的python版本.
    upgrade       同update.
    remove        删除包.
    uninstall     同remove.
    config        配置 conda
    clean         Remove unused packages and caches.
    package       Low-level conda package utility. (EXPERIMENTAL)

optional arguments:
  -h, --help     Show this help message and exit.
  -V, --version  Show the conda version number and exit.

other commands, such as "conda build", are available when additional conda
packages (e.g. conda-build) are installed
```

- 常用命令
  - `conda info -e` 列出所有的虚拟环境
  - `conda create -n py2.7 python=2.7` 安装一个名为 py2.7的环境，python 版本为2.7的最新版
  - `conda update --all` 将所有包升级到最新版
  - `activate py2.7` 切换环境到 py2.7
  - `deactivate` 退出当前虚拟环境

CHAPTER 4

docker 使用

4.1 Docker 命令

Note: Docker 版本: `Docker version 17.06.0-ce, build 02c1d87`
各个版本命令稍有不同

详细命令:

```
Usage:  docker COMMAND

A self-sufficient runtime for containers

Options:
      --config string      本地配置文件路径
  -D, --debug              启用调试模式
      --help               打印本帮助文档
  -H, --host list          进程绑定的 socket(s)
  -l, --log-level string   设置日志级别 ("debug"|"info"|"warn"|"error"|"fatal") (默认
↪"info")
      --tls                连接方式使用 TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\Users\golden\.docker\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\Users\golden\.docker\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\Users\golden\.docker\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            版本

管理命令:
  config      管理设置
  container   管理 containers
```

```
  image       管理 images
  network     管理 networks
  node        管理 Swarm nodes
  plugin      管理 plugins
  secret      管理 Docker secrets
  service     管理 services
  stack       管理 Docker stacks
  swarm       管理 Swarm
  system      管理 Docker
  volume      管理 volumes

Commands:
  attach      重新登录一个正在执行的容器
  build       使用 Dockerfile 生成一个镜像
  commit      在老的镜像基础上创建一个新镜像
  cp          在镜像和本地之间拷贝文件
  create      创建一个新镜像
  diff        比较变化
  events      从服务器拉取个人动态，可选择时间区间。
  exec        在启动的镜像中执行一条命令
  export      将指定的容器保存成 tar 归档文件， docker import 的逆操作。导出后导入（exported-
↪imported)）的容器会丢失所有的提交历史，无法回滚。
  history     查看指定镜像的创建历史。
  images      查看镜像 列表
  import      导入一个镜像
  info        显示 Docker 系统信息，包括镜像和容器数。
  inspect     检查镜像或者容器的参数，默认返回 JSON 格式。
  kill        杀死一个或多个指定容器进程。
  load        从 tar 镜像归档中载入镜像， docker save 的逆操作。保存后再加载（saved-loaded）的
镜像不会丢失提交历史和层，可以回滚。
  login       登录一个镜像仓库。
  logout      退出
  logs        获取容器运行时的输出日志。
  pause       暂停某一容器的所有进程。
  port        List port mappings or a specific mapping for the container
  ps          列出所有运行中容器。
  pull        从 Docker Hub 中拉取或者更新指定镜像。
  push        将镜像推送至远程仓库，默认为 Docker Hub 。
  rename      重命名容器
  restart     重启容器
  rm          从本地移除一个或多个指定的镜像。
  rmi         从本地移除一个或多个指定的镜像。
  run         启动一个容器，在其中运行指定命令
  save        将指定镜像保存成 tar 归档文件， docker load 的逆操作。保存后再加载（saved-
↪loaded）的镜像不会丢失提交历史和层，可以回滚。
  search      从 Docker Hub 中搜索符合条件的镜像。
  start       启动停止的容器
  stats       Display a live stream of container(s) resource usage statistics
  stop        停止正在执行的容器
  tag         标记本地镜像，将其归入某一仓库。
  top         查看一个正在运行容器进程，支持 ps 命令参数。
  unpause     恢复某一容器的所有进程。
  update      更新容器配置
  version     版本
  wait        等待容器停止，返回退出码

Run 'docker COMMAND --help' for more information on a command.
```

4.1.1 docker build

- docker build 的命令:

```
Usage:      docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
      --add-host list              Add a custom host-to-IP mapping (host:ip)
      --build-arg list             创建镜像是的参数
      --cache-from stringSlice     Images to consider as cache sources
      --cgroup-parent string       Optional parent cgroup for the container
      --compress                   Compress the build context using gzip
      --cpu-period int             限制 CPU CFS周期
      --cpu-quota int              限制 CPU CFS配额
  -c, --cpu-shares int             设置 cpu 使用权重
      --cpuset-cpus string         指定使用的CPU id (0-3, 0,1)
      --cpuset-mems string         指定使用的内存 (0-3, 0,1)
      --disable-content-trust      忽略校验 (default true)
  -f, --file string                指定要使用的Dockerfile路径
      --force-rm                   设置镜像过程中删除中间容器
      --help                       Print usage
      --isolation string           使用容器隔离技术
      --label list                 设置镜像使用的元数据
  -m, --memory bytes               设置内存最大值
      --memory-swap bytes          置Swap的最大值为内存: '-1' 不限制
      --network string             设置网络 (default "default")
      --no-cache                   创建镜像的过程不使用缓存
      --pull                       尝试去更新镜像的新版本
  -q, --quiet                      安静模式，成功后只输出镜像ID
      --rm                         设置镜像成功后删除中间容器(default true)
      --security-opt stringSlice   安全选项
      --shm-size bytes             设置/dev/shm的大小
  -t, --tag list                   指定tag 'name:tag'
      --target string              设置目标镜像
      --ulimit ulimit              Ulimit配置 (default [])
```

- Dockerfile 创建规则

Note: Dockerfile里的指令是忽略大小写的，一般都是用大写，“#“作为注释，每一行只支持一条指令，每条指令可以携带多个参数。

指令根据作用可以分为两种:

- 构建指令 操作不会在运行image的容器上执行
- 设置指令 操作将在运行image的容器中执行

它的指令有以下这些:

- FROM 指定一个基础镜像，可以是任意的镜像
  1. 它一定是首个非注释指令
  2. 可以有多个，创建混合的镜像
  3. 没有指定tag，默认使用 latest
- MAINTAINER 指定镜像制作作者的信息

- RUN 在当前image中执行任意合法命令并提交执行结果
  1. 没一个 RUN 都是独立运行的
  2. 指令缓存不会在下个命令执行时自动失效
  3. RUN <command> (命令用shell执行 - `/bin/sh -c`)
  4. RUN ["executable", "param1", "param2" ...] (使用exec 执行)
- ENV 设置容器的环境变量，设置的变量可以用 docker inspect命令来查看
- USER 设置运行命令的用户，默认是 root
- WORKDIR 切换工作目录（默认是 /），相当于 cd, 对RUN,CMD,ENTRYPOINT生效
- COPY <src> <dest> 拷贝文件
  1. 源文件相对被构建的源目录的相对路径
  2. 源文件可以是一个远程的url，并且下下来的文件权限是 600
  3. 所有的新文件和文件夹都会创建UID 和 GID
- ADD <src> <dest> 从src复制文件到container的dest路径
  1. 源文件相对被构建的源目录的相对路径
  2. 源文件也可以是个url
  3. 如果源文件是可识别的压缩文件，会解压
- VOLUME 创建一个可以从本地主机或其他容器挂载的挂载点
- EXPOSE 指定在docker允许时指定的端口进行转发
  1. 端口可以有多个
  2. 运行容器的时候要用 -p 指定设置的端口
  3. 如 docker run -p expose_port:server_port image
- CMD 设置container启动时执行的操作
  1. 可以是自定义脚本
  2. 也可以是系统命令
  3. 有多个只执行最后一个
  4. 当你使用shell或exec格式时， CMD 会自动执行这个命令。
  5. RUN <command> (命令用shell执行 - `/bin/sh -c`)
  6. RUN ["executable", "param1", "param2" ...] (使用exec 执行)(指定了 ENTRYPOINT 必须用这种)
- ENTRYPOINT 设置container启动时执行的操作,和 CMD 差不多
  1. 如果同时设置了 CMD 和 ENTRYPOINT ,并且都是 命令形式， 那最后那个生效
  2. CMD 和 ENTRYPOINT 配合使用的话，ENTRYPOINT 和 CMD 只能是 ‘‘["param"]‘‘形式，‘‘ENTRYPOINT‘‘指定命令,‘‘CMD‘‘指定参数。
- ONBUILD 在子镜像中执行, 指定的命令在构建镜像时并不执行，而是在它的子镜像中执行。使用情景是在建立镜像时取得最新的源码
- ARG 定义的变量 只在建立 image 时有效，建立完成后变量就失效消失
- LABEL 定义一个 image 标签 Owner，并赋值，其值为变量 Name 的值。(LABEL Owner=$Name)

例子网上一大堆

4.1.2 docker run

启动一个容器，在其中运行指定命令。 -a stdin 指定标准输入输出内容类型，可选 STDIN/ STDOUT / STDERR 三项；

- -d 后台运行容器，并返回容器ID；
- -i 以交互模式运行容器，通常与 -t 同时使用；
- -t 为容器重新分配一个伪输入终端，通常与 -i 同时使用；
- –name”nginx-lb” 为容器指定一个名称；
- –dns 8.8.8.8 指定容器使用的DNS服务器，默认和宿主一致；
- –dns-search example.com 指定容器DNS搜索域名，默认和宿主一致；
- -h “mars” 指定容器的hostname；
- -e username”ritchie” 设置环境变量；
- –env-file[] 从指定文件读入环境变量；
- –cpuset”0-2” or –cpuset”0,1,2” 绑定容器到指定CPU运行；
- -c 待完成
- -m 待完成
- –net”bridge” 指定容器的网络连接类型，支持 bridge / host / none container:<namelid> 四种类型；
- –link[] 待完成
- –expose[] 待完成

4.2 windows10 上使用 docker

4.2.1 安装

1. 安装最新版的windows10(必须64位)
2. 开启hyper-v windows自己的虚拟机程序。
3. InstallDocker.msi 下载此文件,并安装
4. 打开命令行，运行 docker 命令，检查是否安装成功。

4.2.2 设置

1. Advanced里面可以设置docker使用的cpu个数和内存大小，如果启动的时候提示内存不足，可适当的调小内存。
2. 网络和代理也是一看就懂。
3. Docker Deamon 里可以编辑它的配置，如添加个仓库:

```
{
  "registry-mirrors": [
    "daocloud.io"
  ],
```

```
  "insecure-registries": [],
  "debug": false
}
```

CHAPTER 5

一些作品

5.1 一个简单的计算器

这是初学python时的一个练手脚本，使用了：

- wxpython
- math

功能：

- 初级
- 中级
- 高级
- ... 等等

给一个编译后的: `calculator.exe`

这是效果图

源码：

```
# coding:utf-8
from __future__ import absolute_import, unicode_literals
import wx
import math

'''
Created on 2014年7月8日

@author: golden
'''


class CalculatorFrame(wx.Frame):
    def __init__(self, parent, _id, size, pos, option):
        wx.Frame.__init__(self, parent, _id, u'我的计算器', size=size, pos=pos)
        self.StatusBar = self.CreateStatusBar()
        self.create_menu_bar()
        self.option = option
        panel = wx.Panel(self, -1)
```

```
        panel.SetBackgroundColour('white')
        self.set_button(panel)
        self.txt_box = wx.TextCtrl(panel, -1, "", pos=(20, 30), size=(340, 30),␣
↪style=wx.ALIGN_RIGHT)
        self.txt_box.SetForegroundColour('blue')
        wx.StaticText(panel, -1, u"输入:", pos=(10, 10))
        self.txt_box1 = wx.TextCtrl(panel, -1, "", pos=(20, 80), size=(340, 30),␣
↪style=wx.ALIGN_RIGHT)
        wx.StaticText(panel, -1, u"输出:", pos=(10, 60))

    def set_button(self, panel):
        i = 0
        if self.option == u'高级':
            for value, name, x_size, y_size, handle in self.high_button_vau:
                x, y = 1, 1
                x, y = x + i // 5, y + i % 5
                pos = (20 + (y - 1) * 70, 120 + (x - 1) * 50)
                size = (x_size, y_size)
                i += 1
                self.create_button(panel, label=value, pos=pos, size=size,␣
↪handle=handle, name=name)
        elif self.option == u'初级':
            for value, name, x_size, y_size, handle in self.simple_button_value:
                x, y = 1, 1  # x: 列  y: 行
                x, y = x + i // 5, y + i % 5
                pos = (20 + (y - 1) * 70, 120 + (x - 1) * 50)
                size = (x_size, y_size)
                i += 1
                self.create_button(panel, label=value, pos=pos, size=size,␣
↪handle=handle, name=name)
        elif self.option == u'中级':
            for value, name, x_size, y_size, handle in self.middle_button_vau:
                x, y = 1, 1
                x, y = x + i // 5, y + i % 5
                pos = (20 + (y - 1) * 70, 120 + (x - 1) * 50)
                size = (x_size, y_size)
                i += 1
                self.create_button(panel, label=value, pos=pos, size=size,␣
↪handle=handle, name=name)
        else:
            print('版本选择错误')

    def create_button(self, panel, label, pos, size, handle, name):
        button = wx.Button(panel, label=label, pos=pos, size=size, name=name)
        button.SetForegroundColour('blue')
        button.Bind(wx.EVT_BUTTON, handle, button)
        button.Bind(wx.EVT_ENTER_WINDOW, self.on_enter_window, button)
        button.Bind(wx.EVT_LEAVE_WINDOW, self.on_enter_window, button)

    def create_menu_bar(self):
        menu_bar = wx.MenuBar()
        for menu in self.menu_data:
            menu_label = menu[0]
            menu_item = menu[1:]
            menu_bar.Append(self.create_menu(menu_item), menu_label)
        self.SetMenuBar(menu_bar)
        return menu_bar

```

```
    def create_menu(self, menu_data):
        menu = wx.Menu()
        for eachLabel, eachStatus, eachHandler in menu_data:
            if not eachLabel:
                menu.AppendSeparator()
                continue
            menu_item = menu.Append(-1, eachLabel, eachStatus)
            self.Bind(wx.EVT_MENU, eachHandler, menu_item)
        return menu

    def on_click(self, event):
        self.txt_box.SetValue(self.txt_box.GetValue() + event.GetEventObject().
↪GetLabel())

    def del_click(self, event):
        self.txt_box.SetForegroundColour('Default')
        val = self.txt_box.GetValue()
        self.txt_box.SetValue(val[:-1])
        self.txt_box1.SetValue('')

    def clear_click(self, event):
        self.txt_box1.SetForegroundColour('Default')
        self.txt_box.SetValue('')
        self.txt_box1.SetValue('')

    def equ_click(self, event):
        try:
            self.txt_box1.SetForegroundColour('Default')
            self.txt_box1.SetValue(str(eval(self.txt_box.GetValue())))
        except ZeroDivisionError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'除数为0，请重新输入！')
        except SyntaxError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
        except ValueError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')

    def suq_click(self, event):
        try:
            self.txt_box1.SetForegroundColour('Default')
            self.txt_box1.SetValue(str(math.pow(eval(self.txt_box.GetValue()), 2)))
        except SyntaxError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
        except ValueError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')

    def suqt_click(self, event):
        try:
            self.txt_box1.SetForegroundColour('Default')
            self.txt_box1.SetValue(str(math.pow(eval(self.txt_box.GetValue()), 1.0 /␣
↪2)))
        except SyntaxError:
            self.txt_box1.SetForegroundColour('red')
            self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
```

```
129             except ValueError:
130                 self.txt_box1.SetForegroundColour('red')
131                 self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
132 
133     def suq3_click(self, event):
134         try:
135             self.txt_box1.SetForegroundColour('Default')
136             self.txt_box1.SetValue(str(math.pow(eval(self.txt_box.GetValue()), 3)))
137         except SyntaxError:
138             self.txt_box1.SetForegroundColour('red')
139             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
140         except ValueError:
141             self.txt_box1.SetForegroundColour('red')
142             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
143 
144     def suqt3_click(self, event):
145         try:
146             self.txt_box1.SetForegroundColour('Default')
147             self.txt_box1.SetValue(str(math.pow(eval(self.txt_box.GetValue()), 1.0 /␣
    ↪3)))
148         except SyntaxError:
149             self.txt_box1.SetForegroundColour('red')
150             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
151         except ValueError:
152             self.txt_box1.SetForegroundColour('red')
153             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
154 
155     def acos_click(self, event):
156         try:
157             self.txt_box1.SetForegroundColour('Default')
158             self.txt_box1.SetValue(str(math.acos(eval(self.txt_box.GetValue()))))
159         except ZeroDivisionError:
160             self.txt_box1.SetForegroundColour('red')
161             self.txt_box1.SetValue(u'除数为0，请重新输入！')
162         except SyntaxError:
163             self.txt_box1.SetForegroundColour('red')
164             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
165         except ValueError:
166             self.txt_box1.SetForegroundColour('red')
167             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
168 
169     def acosh_click(self, event):
170         try:
171             self.txt_box1.SetForegroundColour('Default')
172             self.txt_box1.SetValue(str(math.acosh(eval(self.txt_box.GetValue()))))
173         except ZeroDivisionError:
174             self.txt_box1.SetForegroundColour('red')
175             self.txt_box1.SetValue(u'除数为0，请重新输入！')
176         except SyntaxError:
177             self.txt_box1.SetForegroundColour('red')
178             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
179         except ValueError:
180             self.txt_box1.SetForegroundColour('red')
181             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
182 
183     def asin_click(self, event):
184         try:
185             self.txt_box1.SetForegroundColour('Default')
```

```
186             self.txt_box1.SetValue(str(math.asin(eval(self.txt_box.GetValue()))))
187         except ZeroDivisionError:
188             self.txt_box1.SetForegroundColour('red')
189             self.txt_box1.SetValue(u'除数为0，请重新输入！')
190         except SyntaxError:
191             self.txt_box1.SetForegroundColour('red')
192             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
193         except ValueError:
194             self.txt_box1.SetForegroundColour('red')
195             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
196 
197     def atan_click(self, event):
198         try:
199             self.txt_box1.SetForegroundColour('Default')
200             self.txt_box1.SetValue(str(math.atan(eval(self.txt_box.GetValue()))))
201         except ZeroDivisionError:
202             self.txt_box1.SetForegroundColour('red')
203             self.txt_box1.SetValue(u'除数为0，请重新输入！')
204         except SyntaxError:
205             self.txt_box1.SetForegroundColour('red')
206             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
207         except ValueError:
208             self.txt_box1.SetForegroundColour('red')
209             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
210 
211     def atan2_click(self, event):
212         try:
213             self.txt_box1.SetForegroundColour('Default')
214             self.txt_box1.SetValue(str(math.atan2(eval(self.txt_box.GetValue()))))
215         except ZeroDivisionError:
216             self.txt_box1.SetForegroundColour('red')
217             self.txt_box1.SetValue(u'除数为0，请重新输入！')
218         except SyntaxError:
219             self.txt_box1.SetForegroundColour('red')
220             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
221         except ValueError:
222             self.txt_box1.SetForegroundColour('red')
223             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
224 
225     def atanh_click(self, event):
226         try:
227             self.txt_box1.SetForegroundColour('Default')
228             self.txt_box1.SetValue(str(math.atanh(eval(self.txt_box.GetValue()))))
229         except ZeroDivisionError:
230             self.txt_box1.SetForegroundColour('red')
231             self.txt_box1.SetValue(u'除数为0，请重新输入！')
232         except SyntaxError:
233             self.txt_box1.SetForegroundColour('red')
234             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
235         except ValueError:
236             self.txt_box1.SetForegroundColour('red')
237             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
238 
239     def cos_click(self, event):
240         try:
241             self.txt_box1.SetForegroundColour('Default')
242             self.txt_box1.SetValue(str(math.cos(eval(self.txt_box.GetValue()))))
243         except ZeroDivisionError:
```

```
244             self.txt_box1.SetForegroundColour('red')
245             self.txt_box1.SetValue(u'除数为0，请重新输入！')
246         except SyntaxError:
247             self.txt_box1.SetForegroundColour('red')
248             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
249         except ValueError:
250             self.txt_box1.SetForegroundColour('red')
251             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
252
253     def cosh_click(self, event):
254         try:
255             self.txt_box1.SetForegroundColour('Default')
256             self.txt_box1.SetValue(str(math.cosh(eval(self.txt_box.GetValue()))))
257         except ZeroDivisionError:
258             self.txt_box1.SetForegroundColour('red')
259             self.txt_box1.SetValue(u'除数为0，请重新输入！')
260         except SyntaxError:
261             self.txt_box1.SetForegroundColour('red')
262             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
263         except ValueError:
264             self.txt_box1.SetForegroundColour('red')
265             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
266
267     def sin_click(self, event):
268         try:
269             self.txt_box1.SetForegroundColour('Default')
270             self.txt_box1.SetValue(str(math.sin(eval(self.txt_box.GetValue()))))
271         except ZeroDivisionError:
272             self.txt_box1.SetForegroundColour('red')
273             self.txt_box1.SetValue(u'除数为0，请重新输入！')
274         except SyntaxError:
275             self.txt_box1.SetForegroundColour('red')
276             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
277         except ValueError:
278             self.txt_box1.SetForegroundColour('red')
279             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
280
281     def tan_click(self, event):
282         try:
283             self.txt_box1.SetForegroundColour('Default')
284             self.txt_box1.SetValue(str(math.tan(eval(self.txt_box.GetValue()))))
285         except ZeroDivisionError:
286             self.txt_box1.SetForegroundColour('red')
287             self.txt_box1.SetValue(u'除数为0，请重新输入！')
288         except SyntaxError:
289             self.txt_box1.SetForegroundColour('red')
290             self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
291         except ValueError:
292             self.txt_box1.SetForegroundColour('red')
293             self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
294
295     def tanh_click(self, event):
296         try:
297             self.txt_box1.SetForegroundColour('Default')
298             self.txt_box1.SetValue(str(math.tanh(eval(self.txt_box.GetValue()))))
299         except ZeroDivisionError:
300             self.txt_box1.SetForegroundColour('red')
301             self.txt_box1.SetValue(u'除数为0，请重新输入！')
```

```
302             except SyntaxError:
303                 self.txt_box1.SetForegroundColour('red')
304                 self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
305             except ValueError:
306                 self.txt_box1.SetForegroundColour('red')
307                 self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
308 
309         def degrees_click(self, event):
310             try:
311                 self.txt_box1.SetForegroundColour('Default')
312                 self.txt_box1.SetValue(str(math.degrees(eval(self.txt_box.GetValue()))))
313             except ZeroDivisionError:
314                 self.txt_box1.SetForegroundColour('red')
315                 self.txt_box1.SetValue(u'除数为0，请重新输入！')
316             except SyntaxError:
317                 self.txt_box1.SetForegroundColour('red')
318                 self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
319             except ValueError:
320                 self.txt_box1.SetForegroundColour('red')
321                 self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
322 
323         def radians_click(self, event):
324             try:
325                 self.txt_box1.SetForegroundColour('Default')
326                 self.txt_box1.SetValue(str(math.radians(eval(self.txt_box.GetValue()))))
327             except ZeroDivisionError:
328                 self.txt_box1.SetForegroundColour('red')
329                 self.txt_box1.SetValue(u'除数为0，请重新输入！')
330             except SyntaxError:
331                 self.txt_box1.SetForegroundColour('red')
332                 self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
333             except ValueError:
334                 self.txt_box1.SetForegroundColour('red')
335                 self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
336 
337         def log10_click(self, event):
338             try:
339                 self.txt_box1.SetForegroundColour('Default')
340                 self.txt_box1.SetValue(str(math.log10(eval(self.txt_box.GetValue()))))
341             except ZeroDivisionError:
342                 self.txt_box1.SetForegroundColour('red')
343                 self.txt_box1.SetValue(u'除数为0，请重新输入！')
344             except SyntaxError:
345                 self.txt_box1.SetForegroundColour('red')
346                 self.txt_box1.SetValue(u'输入算式错误，请重新输入！')
347             except ValueError:
348                 self.txt_box1.SetForegroundColour('red')
349                 self.txt_box1.SetValue(u'输入值错误，不能运算，请重新输入！')
350 
351         def pi_click(self, event):
352             self.txt_box.SetValue(self.txt_box.GetValue() + event.GetEventObject().
    ↪GetLabel())
353             self.txt_box1.SetValue(str(math.pi))
354 
355         def e_click(self, event):
356             self.txt_box.SetValue(self.txt_box.GetValue() + event.GetEventObject().
    ↪GetLabel())
357             self.txt_box1.SetValue(str(math.e))
```

```
358
359     def on_close_me(self, event):
360         self.dlg = wx.MessageDialog(None, u'确定关闭？', u'消息框(MessageDialog)', wx.
    ↪YES_NO | wx.ICON_WARNING)
361         if (self.dlg.ShowModal() == wx.ID_NO):
362             self.dlg.Destroy()
363         else:
364             self.Destroy()
365
366     def copy_click(self, event):
367         if wx.TheClipboard.Open():
368             data_obj = wx.TextDataObject()
369             data_obj.SetText(self.txt_box1.GetValue())
370             wx.TheClipboard.SetData(data_obj)
371         else:
372             wx.MessageBox(u"不能打开剪切板", u"Error")
373
374     def paste_click(self, event):
375         self.txt_box.Paste()
376
377     @staticmethod
378     def clear_clipboard(event):
379         if wx.TheClipboard.Open():
380             wx.TheClipboard.Flush()
381             wx.TheClipboard.Close()
382         else:
383             wx.MessageBox(u"不能打开剪切板", "Error")
384
385     @staticmethod
386     def about_me_click(event):
387         wx.MessageBox(u'计算器版本1.0', u"关于", wx.OK | wx.ICON_INFORMATION)
388
389     def set_click(self, event):
390         ob = event.GetEventObject()
391         opt = ob.GetLabel(event.GetId())
392         self.chang_win(opt)
393
394     def option_click(self, event):
395         dlg = wx.SingleChoiceDialog(None, u'选择计算器', u'选项', [u'初级', u'中级', u'高级
    ↪'])
396         if dlg.ShowModal() == wx.ID_OK:
397             opt = dlg.GetStringSelection()
398             self.chang_win(opt)
399         else:
400             dlg.Destroy()
401
402     @property
403     def menu_data(self):
404         return (
405             (u"文件",
406              (u"新建", u"新建窗口", self.new_win),
407              (u'关闭', u'关闭当前窗口', self.on_close_me)),
408             (u'编辑',
409              (u'复制', u'复制结果到剪切板', self.copy_click),
410              (u'粘贴', u'粘贴剪切板内容到输入框', self.paste_click),
411              (u'清空剪切板', u'清空剪切板', self.clear_clipboard),
412              ('', '', ''),
413              (u'选项', u'选项', self.option_click)),
```

```
            (u'设置',
             (u'初级', u'初级', self.set_click),
             (u'中级', u'中级', self.set_click),
             (u'高级', u'高级', self.set_click)),
            (u'帮助',
             (u'关于', u'关于', self.about_me_click))
        )

    @property
    def high_button_vau(self):
        x_size = 60
        y_size = 40
        return (
            (u'1', u'数字1', x_size, y_size, self.on_click),
            (u'2', u'数字2', x_size, y_size, self.on_click),
            (u'3', u'数字3', x_size, y_size, self.on_click),
            (u'+', u'加法(正)', x_size, y_size, self.on_click),
            (u'-', u'减肥(负)', x_size, y_size, self.on_click),

            (u'4', u'数字4', x_size, y_size, self.on_click),
            (u'5', u'数字5', x_size, y_size, self.on_click),
            (u'6', u'数字6', x_size, y_size, self.on_click),
            (u'*', u'乘法', x_size, y_size, self.on_click),
            (u'/', u'除法', x_size, y_size, self.on_click),

            (u'7', u'数字7', x_size, y_size, self.on_click),
            (u'8', u'数字8', x_size, y_size, self.on_click),
            (u'9', u'数字9', x_size, y_size, self.on_click),
            (u'(', u'左括号', x_size, y_size, self.on_click),
            (u')', u'右括号', x_size, y_size, self.on_click),

            (u'0', u'数字0', x_size, y_size, self.on_click),
            (u'.', u'点号', x_size, y_size, self.on_click),
            (u'=', u'等号', x_size, y_size, self.equ_click),
            (u'clear', u'清除', x_size, y_size, self.clear_click),
            (u'del', u'删除', x_size, y_size, self.del_click),

            (u'^2', u'平方', x_size, y_size, self.suq_click),
            (u'2✓', u'平方根', x_size, y_size, self.suqt_click),
            (u'3✓', u'根三', x_size, y_size, self.suqt3_click),
            (u'e', u'自然常数', x_size, y_size, self.e_click),
            (u'π', u'圆周率', x_size, y_size, self.pi_click),

            (u'|', u'按位或', x_size, y_size, self.on_click),
            (u'~', u'按位取反', x_size, y_size, self.on_click),
            (u'&amp;', u'按位与', x_size, y_size, self.on_click),
            (u'&lt;&lt;', u'向左移', x_size, y_size, self.on_click),
            (u'&gt;&gt;', u'向右移', x_size, y_size, self.on_click),

            (u'^', u'按位异或', x_size, y_size, self.on_click),
            (u'acos', u'反余弦', x_size, y_size, self.acos_click),
            (u'acosh', u'反双曲余弦', x_size, y_size, self.acosh_click),
            (u'asin', u'反正弦', x_size, y_size, self.asin_click),
            (u'atan', u'反正切', x_size, y_size, self.atan_click),

            (u'atan2', u'反正切', x_size, y_size, self.atan2_click),
            (u'atanh', u'反双曲正弦', x_size, y_size, self.atanh_click),
            (u'cos', u'余弦', x_size, y_size, self.cos_click),
```

```
            (u'cosh', u'双曲余弦', x_size, y_size, self.cosh_click),
            (u'sin', u'正弦', x_size, y_size, self.sin_click),

            (u'tan', u'正切', x_size, y_size, self.tan_click),
            (u'tanh', u'双曲正切', x_size, y_size, self.tanh_click),
            (u'degrees', u' x （弧长） 转成角度', x_size, y_size, self.degrees_click),
            (u'radians', u' x(角度） 转成弧长', x_size, y_size, self.radians_click),
            (u'log10', u'log10', x_size, y_size, self.log10_click)
        )

    @property
    def simple_button_value(self):
        x_size = 60
        y_size = 40
        return ((u'1', u'数字1', x_size, y_size, self.on_click),
                (u'2', u'数字2', x_size, y_size, self.on_click),
                (u'3', u'数字3', x_size, y_size, self.on_click),
                (u'+', u'加法(正)', x_size, y_size, self.on_click),
                (u'-', u'减肥(负)', x_size, y_size, self.on_click),

                (u'4', u'数字4', x_size, y_size, self.on_click),
                (u'5', u'数字5', x_size, y_size, self.on_click),
                (u'6', u'数字6', x_size, y_size, self.on_click),
                (u'*', u'乘法', x_size, y_size, self.on_click),
                (u'/', u'除法', x_size, y_size, self.on_click),

                (u'7', u'数字7', x_size, y_size, self.on_click),
                (u'8', u'数字8', x_size, y_size, self.on_click),
                (u'9', u'数字9', x_size, y_size, self.on_click),
                (u'(', u'左括号', x_size, y_size, self.on_click),
                (u')', u'右括号', x_size, y_size, self.on_click),

                (u'0', u'数字0', x_size, y_size, self.on_click),
                (u'.', u'点号', x_size, y_size, self.on_click),
                (u'=', u'等号', x_size, y_size, self.equ_click),
                (u'clear', u'清除', x_size, y_size, self.clear_click),
                (u'del', u'删除', x_size, y_size, self.del_click))

    @property
    def middle_button_vau(self):
        x_size = 60
        y_size = 40
        return ((u'1', u'数字1', x_size, y_size, self.on_click),
                (u'2', u'数字2', x_size, y_size, self.on_click),
                (u'3', u'数字3', x_size, y_size, self.on_click),
                (u'+', u'加法(正)', x_size, y_size, self.on_click),
                (u'-', u'减肥(负)', x_size, y_size, self.on_click),

                (u'4', u'数字4', x_size, y_size, self.on_click),
                (u'5', u'数字5', x_size, y_size, self.on_click),
                (u'6', u'数字6', x_size, y_size, self.on_click),
                (u'*', u'乘法', x_size, y_size, self.on_click),
                (u'/', u'除法', x_size, y_size, self.on_click),

                (u'7', u'数字7', x_size, y_size, self.on_click),
                (u'8', u'数字8', x_size, y_size, self.on_click),
                (u'9', u'数字9', x_size, y_size, self.on_click),
                (u'(', u'左括号', x_size, y_size, self.on_click),
```

```
                (u')', u'右括号', x_size, y_size, self.on_click),

                (u'0', u'数字0', x_size, y_size, self.on_click),
                (u'.', u'点号', x_size, y_size, self.on_click),
                (u'=', u'等号', x_size, y_size, self.equ_click),
                (u'clear', u'清除', x_size, y_size, self.clear_click),
                (u'del', u'删除', x_size, y_size, self.del_click),

                (u'sin', u'正弦', x_size, y_size, self.sin_click),
                (u'cos', u'余弦', x_size, y_size, self.cos_click),
                (u'tan', u'正切', x_size, y_size, self.tan_click),
                (u'e', u'自然常数', x_size, y_size, self.e_click),
                (u'π', u'圆周率', x_size, y_size, self.pi_click),
                )

    @property
    def win_size(self):
        return ((u'初级', 400, 410),
                (u'中级', 400, 510),
                (u'高级', 400, 670),
                )

    def on_enter_window(self, event):
        ob = event.GetEventObject()
        ob.SetForegroundColour('red')
        ob.SetBackgroundColour('white')
        self.StatusBar.SetBackgroundColour('grey')
        self.StatusBar.SetStatusText(ob.GetName())
        event.Skip()

    def on_leave_window(self, event):
        ob = event.GetEventObject()
        ob.SetForegroundColour('blue')
        ob.SetBackgroundColour('Default')
        self.StatusBar.SetBackgroundColour('Default')
        self.StatusBar.SetStatusText('')
        event.Skip()

    def new_win(self, event):
        self.Destroy()
        app = wx.App(redirect=False)
        frame = CalculatorFrame(parent=None, _id=-1, size=(400, 620), pos=(100, 100),␣
↪option=option)
        frame.Show()
        app.MainLoop()

    def chang_win(self, opt):
        self.Destroy()
        for _opt, x_size, y_size in self.win_size:
            size = (x_size, y_size)
            if _opt == opt:
                app = wx.App(redirect=False)
                frame = CalculatorFrame(parent=None, _id=-1, size=size, pos=(100, 50),
↪ option=opt)
                frame.Show()
                app.MainLoop()


```

```
586 if __name__ == "__main__":
587     app = wx.App(redirect=False)
588     option = u'初级'
589     frame = CalculatorFrame(parent=None, _id=-1, size=(400, 410), pos=(100, 50),␣
    ↪option=option)
590     frame.Show()
591     app.MainLoop()
592     app.Destroy()
```

5.2 一个简单的糗百客户端

这是初学python时的一个练手脚本，使用了：

- wxpython
- requests
- BeautifulSoup4
- ...

功能:

- 后台爬取数据
- 预先加载
- 图片也能显示
- 类别切换
- ... 等等

给一个编译后的: `qb.exe`

这是效果图

源码：

```
# coding:utf-8

from __future__ import absolute_import, unicode_literals

__author__ = "golden"
__date__ = '2017/8/3'

```

```
# !/usr/bin/python
# -*- coding:utf-8 -*-


import wx
import threading
import time
import sys, os
import requests
from bs4 import BeautifulSoup


#######################################################################
class SpiderThread(threading.Thread):
    """
        爬虫 线程
    """

    # ----------------------------------------------------------------------
    def __init__(self, category, page_data, current_page, current_item, show_info, set_status_text, lock, name):
        threading.Thread.__init__(self)
        self.category = category
        self.page_data = page_data
        self.current_page = current_page
        self.current_item = current_item
        self.set_status_text = set_status_text
        self.show_info = show_info
        self.lock = lock
        self.name = name

    def run(self):
        self.load_page()

    def get_page(self, page_num):
        wx.CallAfter(self.set_status_text, message='正在加载第 %s 页...' % str(page_num))
        URL = "http://www.qiushibaike.com/" + self.category + "/page/" + str(page_num)
        agent_header = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
        headers = {'User-Agent': agent_header}
        page_content = requests.get(URL, headers=headers).content  # .decode('utf-8').encode('GBK','ignore')
        soup = BeautifulSoup(page_content, "lxml")
        items1 = soup.find_all('div', class_='article')
        item_number = 1
        if not os.path.exists('tmp_jpg'):
            os.makedirs('tmp_jpg')
        if not os.path.exists(r'tmp_jpg/def.jpg'):
            ir = requests.get('http://pic8.nipic.com/20100703/4887831_015505282659_2.jpg')
            open('tmp_jpg/def.jpg', 'wb').write(ir.content)
        for item in items1:
            try:
                myjpg = item.find('div', class_='thumb').find('img')
            except Exception as e:
                myjpg = None
            stats_vote = item.find('span', class_='stats-vote').find('i').get_text()  ####提取好笑个数
```

```
61              stats_comments = item.find('span', class_='stats-comments').find('i').get_
↪text()  ####提取回复个数
62              voting = [span.get_text() for span in item.find_all('span', class_=
↪'number hidden')]  ####提取顶、拍个数
63              auth = item.find('h2').get_text().strip()
64              content = item.find('div', class_='content').find('span').get_text().
↪strip()
65              if page_num not in self.page_data.keys():
66                  self.page_data[page_num] = {}
67              self.page_data[page_num].update(
68                  {item_number: {
69                      'auth': auth,
70                      'content': content,
71                      'stats_vote': stats_vote,
72                      'stats_comments': stats_comments,
73                      'voting_up': voting[0],
74                      'voting_down': voting[1],
75                      'jpg': '',
76                      'jpg_name': '',
77                  }})
78              if myjpg:
79                  jpg_name = myjpg['src'].split('/')[-1]
80                  ir = requests.get(myjpg['src'].replace('//', 'http://'))
81                  open('tmp_jpg/' + jpg_name, 'wb').write(ir.content)
82                  self.page_data[page_num][item_number].update({
83                      'jpg': myjpg,
84                      'jpg_name': jpg_name,
85                  })
86              item_number += 1
87              wx.CallAfter(self.set_status_text, message='第 %s 页已加载 %s 条' %␣
↪(str(page_num), str(item_number)))
88          wx.CallAfter(self.show_info, message='第%s页(共%s条)加载完成。' % (str(page_num),
↪ str(item_number)))
89
90      def load_page(self):
91          while self.lock.isSet():
92              if self.page_count < self.current_page + 2:
93                  try:
94                      self.get_page(str(self.page_count + 1))
95                      time.sleep(0.1)
96                  except Exception as ex:
97                      msg = u'无法连接糗百: %s' % ex
98                      wx.CallAfter(self.set_status_text, message=msg)
99                      time.sleep(1)
100             else:
101                 msg = u'加载爬虫休眠中...'
102                 wx.CallAfter(self.set_status_text, message=msg)
103                 time.sleep(3)
104         wx.CallAfter(self.set_status_text, message='%s 成功退出。' % self.name)
105
106     @property
107     def page_count(self):
108         return len(self.page_data.keys())
109
110
111 ########################################################################
112 class MyFrame(wx.Frame):
113     """
```

```
114         重构Frame
115     """
116 
117     # --------------------------------------------------------------------------
118     def __init__(self, page_data, current_page, current_item):
119         self.page_data = page_data
120         self.current_page = current_page
121         self.current_item = current_item
122         self.current_page_data = {}
123         self.current_item_data = {}
124         self.category = 'hot'
125         self.lock = threading.Event()
126         wx.Frame.__init__(self, None, -1, u'我的糗百客户端', size=(600, 720))
127         self.create_menu_bar()
128         panel = wx.Panel(self, -1)
129         panel.SetBackgroundColour('white')
130         self.qbtext = wx.TextCtrl(panel, -1, pos=(100, 10), size=(400, 150),
131                                   style=wx.TE_CENTER | wx.TE_READONLY | wx.TE_
    ↪MULTILINE | wx.TE_NOHIDESEL | wx.TE_RICH2)
132         self.stc = wx.StaticText(panel, -1, pos=(150, 0))
133         self.stccom = wx.StaticText(panel, -1, pos=(150, 155))
134         self.jpgbutton = wx.BitmapButton(panel, -1)
135         self.status_bar = self.CreateStatusBar()
136         next_button = wx.Button(panel, label=u'下一条', pos=(520, 300), size=(40, 100),
    ↪ style=wx.BU_ALIGN_MASK)
137         next_button.Bind(wx.EVT_BUTTON, self.next_item, next_button)
138         previous_button = wx.Button(panel, label=u'上一条', pos=(20, 300), size=(40,␣
    ↪100), style=wx.BU_ALIGN_MASK)
139         previous_button.Bind(wx.EVT_BUTTON, self.previous_item, previous_button)
140         self.jpgbutton.Bind(wx.EVT_BUTTON, self.next_item, self.jpgbutton)
141         self.show_info()
142         self.Show()
143 
144     def show_info(self, message=''):
145         self.load_info()
146         if self.current_item_data:
147             text = self.current_item_data.get('content')
148             voting_up = self.current_item_data.get('voting_up')
149             voting_down = self.current_item_data.get('voting_down')
150             stats_comments = self.current_item_data.get('stats_comments')
151             stats_vote = self.current_item_data.get('stats_vote')
152             auth = self.current_item_data.get('auth')
153         else:
154             text = '正在加载中...'
155             voting_up = 0
156             voting_down = 0
157             stats_comments = 0
158             stats_vote = 0
159             auth = ''
160         self.qbtext.SetLabel(text)
161         self.stc.SetLabel(
162             u'第 ' + str(self.current_page) + u' 页 第 ' + str(self.current_item) + u'␣
    ↪条 作者: ' + auth)
163         self.stccom.SetLabel(
164             u'%s个顶  %s个拍  %s个评论  %s个好笑' % (voting_up, voting_down, stats_
    ↪comments, stats_vote))
165         self.jpgbutton.SetBitmap(self.jpg)
166         self.jpgbutton.SetPosition(self.jpg_pose)
```

```
        self.jpgbutton.SetSize(self.jpg_size)

    def set_status_text(self, message):
        if not self.status_bar.GetStatusText() == message:
            self.status_bar.SetStatusText(message)

    def next_item(self, event):
        if self.current_page_item_count > self.current_item:
            self.current_item += 1
        else:
            self.current_page += 1
            self.current_item = 1
            if str(self.current_page) in self.page_data:
                self.current_page_data = self.page_data[str(self.current_page)]
        self.show_info()

    @property
    def current_page_item_count(self):
        return len(self.current_page_data.keys())

    def previous_item(self, event):
        if self.current_item > 1:  # 当前页面大于1,到当前页前一条
            self.current_item -= 1
        else:  # 到前一页最后一条
            if self.current_page > 1:  # 有前一页
                self.current_page -= 1
                self.current_page_data = self.page_data[str(self.current_page)]
                self.current_item = max(self.current_page_data.keys())
            else:
                self.set_status_text(u'前面没有页了')
        self.show_info()

    def load_info(self):
        if not self.current_page_data:
            self.current_page_data = self.page_data.get(str(self.current_page), {})
        if self.current_item in self.current_page_data.keys():
            self.current_item_data = self.current_page_data[self.current_item]
            if self.current_item_data.get('jpg'):
                jpg_path = 'tmp_jpg/' + self.current_item_data.get('jpg_name')
                jpg = wx.Image(jpg_path, type=wx.BITMAP_TYPE_JPEG)
                W, H = jpg.GetWidth(), jpg.GetHeight()
                if (W > 400 and H <= 500) or (W > H and W > 400 and H > 500):
                    H = 400 * H / W
                    W = 400
                elif (W <= 400 and H > 500) or (400 < W < H and H > 500):
                    W = 500 * W / H
                    H = 500
                self.jpg_pose = (300 - W / 2, 420 - H / 2)
                self.jpg_size = (W, H)
                self.jpg = jpg.Rescale(W, H).ConvertToBitmap()
            else:
                jpg_path = 'tmp_jpg/def.jpg'
                self.jpg_pose = (150, 270)
                self.jpg_size = (301, 300)
                self.jpg = wx.Image(jpg_path, type=wx.BITMAP_TYPE_JPEG).
↪ConvertToBitmap()
        else:
            jpg_path = 'tmp_jpg/def.jpg'
```

```
            self.jpg_pose = (150, 270)
            self.jpg_size = (301, 300)
            self.jpg = wx.Image(jpg_path, type=wx.BITMAP_TYPE_JPEG).ConvertToBitmap()

    def create_menu_bar(self):
        menu_bar = wx.MenuBar()
        for each in self.menu_data:
            menu_label = each[0]
            menu_item = each[1:]
            menu_bar.Append(self.create_menu(menu_item), menu_label)
        self.SetMenuBar(menu_bar)
        return menu_bar

    def create_menu(self, menu_data):
        menu = wx.Menu()
        kind = wx.ITEM_NORMAL
        for _data in menu_data:
            if len(_data) == 3:
                label, status, handler = _data
            else:
                label, status, handler, kind = _data
            if not label:
                menu.AppendSeparator()
                continue
            menu_item = menu.Append(-1, label, status, kind)
            self.Bind(wx.EVT_MENU, handler, menu_item)
        return menu

    def set_category(self, event):
        categorys = {
            '8hr': '热门',
            'hot': '24小时',
            'imgrank': '热图',
            'text': '文字',
            'history': '穿越',
            'pic': '糗图',
            'textnew': '新鲜'
        }
        categorys = {categorys[key]: key for key in categorys}
        menu_bar = self.GetMenuBar()
        item_id = event.GetId()
        item = menu_bar.FindItemById(item_id)
        category = categorys.get(item.GetLabel())
        self.category = category
        self.page_data = {}
        self.current_page_data = {}
        self.current_item_data = {}
        self.current_page = 1
        self.current_page = 1
        self.show_info()
        self.stop_spider()
        self.start_spider()

    def defa(self):
        pass

    @property
    def menu_data(self):
```

```
282         return (
283             (u"文件",
284              (u"新建", u"新建窗口", self.defa),
285              (u'关闭', u'关闭当前窗口', self.defa)),
286             (u'编辑',
287              (u'复制', u'复制结果到剪切板', self.defa),
288              (u'粘贴', u'粘贴剪切板内容到输入框', self.defa),
289              (u'清空剪切板', u'清空剪切板', self.defa),
290              ('', '', ''),
291              (u'选项', u'选项', self.defa)),
292             (u'分类',
293              (u'热门', u'热门', self.set_category, wx.ITEM_RADIO),
294              (u'24小时', u'24小时', self.set_category, wx.ITEM_RADIO),
295              (u'热图', u'热图', self.set_category, wx.ITEM_RADIO),
296              (u'文字', u'文字', self.set_category, wx.ITEM_RADIO),
297              (u'穿越', u'穿越', self.set_category, wx.ITEM_RADIO),
298              (u'糗图', u'糗图', self.set_category, wx.ITEM_RADIO),
299              (u'新鲜', u'新鲜', self.set_category, wx.ITEM_RADIO)),
300             (u'帮助',
301              (u'关于', u'关于', self.defa))
302 
303         )
304 
305 
306         ########################################################################
307 
308     def start_spider(self):
309         self.lock.set()
310         sp = SpiderThread(self.category, self.page_data, self.current_page, self.
    ↪current_item, self.show_info,
311                           self.set_status_text, lock=self.lock, name=self.category)
312         sp.setDaemon(1)
313         sp.start()
314         self.spider_thread = sp
315         self.set_status_text('%s 爬虫启动。' % self.category)
316 
317     def stop_spider(self):
318         self.lock.clear()
319         while True:
320             if self.spider_thread.is_alive():
321                 time.sleep(1)
322             else:
323                 self.set_status_text('%s 爬虫成功退出' % self.category)
324                 break
325 
326 
327 class MyApp(wx.App):
328     """
329     重构APP
330     """
331 
332     # ----------------------------------------------------------------------
333     def __init__(self, page_data, current_page, current_item):
334         """Constructor"""
335         wx.App.__init__(self)
336         frame = MyFrame(page_data, current_page, current_item)
337         frame.start_spider()
338         frame.Center()
```

```
339         frame.Show()
340         self.page_data = page_data
341         self.current_page = current_page
342         self.current_item = current_item
343         self.frame = frame
344 
345     def MainLoop(self):
346         return super(MyApp, self).MainLoop()
347 
348 
349 ######################################################################
350 
351 
352 if __name__ == '__main__':
353     page_data = {}  # 数据
354     current_page = 1  # 当前页数,从1开始
355     current_item = 1  # 当前条数,从1开始
356     app = MyApp(page_data, current_page, current_item)
357     app.MainLoop()
```

CHAPTER 6

其他页面

- genindex
- search

Index