重新設計了 Pelican 的主題與插件

目錄

2015年2月14日更新

前言: 新天新地，將一切都更新了[1]

不知不覺間放任這邊長草很久了，從上次 折騰主題 到現在都快三年了， 而從上次 寫了篇告白信 到現在也有快兩年了。 這期間曾經把主題配色從 Bootstrap 2 默認的 白底黑字改成了讓眼睛更舒適的黑底白字，也不過是用 drop-in 的配色方案而已，沒有本質上的改進。

洞中一日世上千載，兩年裏 Bootstrap 已經升上 v3.3 , 而 Pelican 則已經升到 3.5 了。 早就眼饞 Bootstrap 和 Pelican 中的諸多新功能新設計，不過無奈於時間有限只能飽飽眼福。

近日想寫的東西越積越多，終於下定決心花了前前後後 **兩個月** 的時間重新設計了一遍 Pelican 的主題，配合一些我覺得有用的插件。於是本博客就變成你們現在看到的樣子了。 （以及本篇博文也用了兩個月的時間寫完，其間還發了幾篇別的短文，算是恢復寫博客的嘗試吧。）

在邁阿密參加 ICSR 2015 的時候 拍到的街邊一家叫 Pelican 的旅館

Bootstrap 3 的新設計

- 全新的 優先移動設備 (mobile-first) 響應式 (responsive) 設計。 原本Bootstrap 2雖然有響應式設計， 不過諸多細節不能符合我的需求，最終還是得手工 hack @media 查詢去微調。 現在的 優先移動設備 (mobile-first) 響應式 (responsive) 柵格系統 (grid system) 則相對顯得科學很多了，也終於能在手持 設備上看起來舒服一些。諸位可以嘗試改變窗口寬度，或者在不同的手持設備上打開這個 blog ，體驗一下這個頁面在不同顯示器大小中的效果。如果仍有問題歡迎 發 Issue 給我 。
- 科學的 導航欄 (Navbar) 。 比 Bootstrap 2 那個科學很多了。無論是 保持 (sticky) 在上端還是跟着浮動， 或者像這邊這樣 自動隱藏 都很簡單。

更多細節參考 Bootstrap 3 主頁 。

Pelican 3.5 的新功能

- Python 2 和 Python 3 統一代碼： 再沒有惱人的 unicode 相關的問題了。這對 blog 系統來說相當重要啊。 而且還能方便切換

pypy 等不同的解釋器。
- 全新的插件系統：非常多功能強大的 插件 等着你。
- 增強了導入系統：嗯總算可以導入我的中文的 wordpress 博客了。（雖然那邊長草更久了……）
- 站內鏈接 ：不用 硬編碼(hard code) 目標頁面的鏈接了，可以直接寫源文件的位置然後讓 pelican 處理，這樣能簡化各種 插件(plugin) 和 主題(theme) 的實現。

更多細節參考 Pelican 文檔 。

新的文件夾佈局

Pelican 的新文件夾佈局

```
.
├──── cache          生成頁面的 pickle 緩存
├──── content        讀取的全部內容
│    ├──── <categories>    按分類存放的文章
│    ├──── pages           像 About 這樣的固定頁面
│    └──── static          文章內用到的靜態內容
├──── drafts         文章的草稿箱
├──── Makefile       生成用的 makefile
├──── pelicanconf.py 測試時用的快速 Pelican 配置
├──── publishconf.py 部署時用的耗時 Pelican 配置
├──── output       -> ../farseerfc.github.io
├──── plugins      -> ../pelican-plugins
└──── theme        -> ../pelican-bootstrap3
```

之前的博客 仍然留在 github 上，其中的內容完全搬過來了。開始寫老博客的時候 Pelican 版本較早，沒有形成好的 文件夾佈局，導致生成的文章、使用的模板和撰寫的內容全都混在一起，非常難以管理， 於是趁改版之際用了新的文件夾佈局方式，並分爲 4 個 git repo 分別管理歷

史。

首先是存放 總的博客內容的 repo ， 其佈局是如圖那樣的。這樣將生成的靜態網站和生成網站用的配置啦內容啦分開之後，頓時清晰了很多。

然後這個內容 repo 中的三個符號鏈接分別指向三個子 repo（沒用 git submodule 管理純粹是因爲偷懶）。 theme 指向 pelican-bootstrap3 ，是我修改過的 pelican 主題。 plugins 指向 pelican-plugins ，由於 plugins 的質量有些參差不齊，其中不少 plugin 都按我的需要做了些許修改，一些是功能改進，另一些則是修bug（比如不少 plugin只支持 python 2）。 最後 output 指向 farseerfc.github.io 也就是發佈的靜態網站啦。

接下來從 **主題** 和 **插件** 兩個方面介紹一下改版的細節。

主題： Material Design 風格的 Bootstrap 3

上篇 博文 就總結了我爲了這個博客尋找了一堆 CSS 框架，並且最終決定用 bootstrap-material-design , DandyDev/pelican-bootstrap3 和 Bootstrap 3 這三個項目結合的方式實現這個模板的主題。 這三個項目都或多或少經過了我的修改，修改後的項目以 pelican-bootstrap3 爲基礎放在 這裏 ，包括 Bootstrap3 樣式 和 Material 樣式。

對 Bootstrap 3 的定製

由於架構完善，修改 Bootstrap 3 感覺非常簡單。另一方面我在 Web 前端技術上的技能點也不多， 所以修改的地方非常有限，只能按我自己的需求定製而已。

響應式設備的大小

修改了 Bootstrap 3 響應式設備的大小

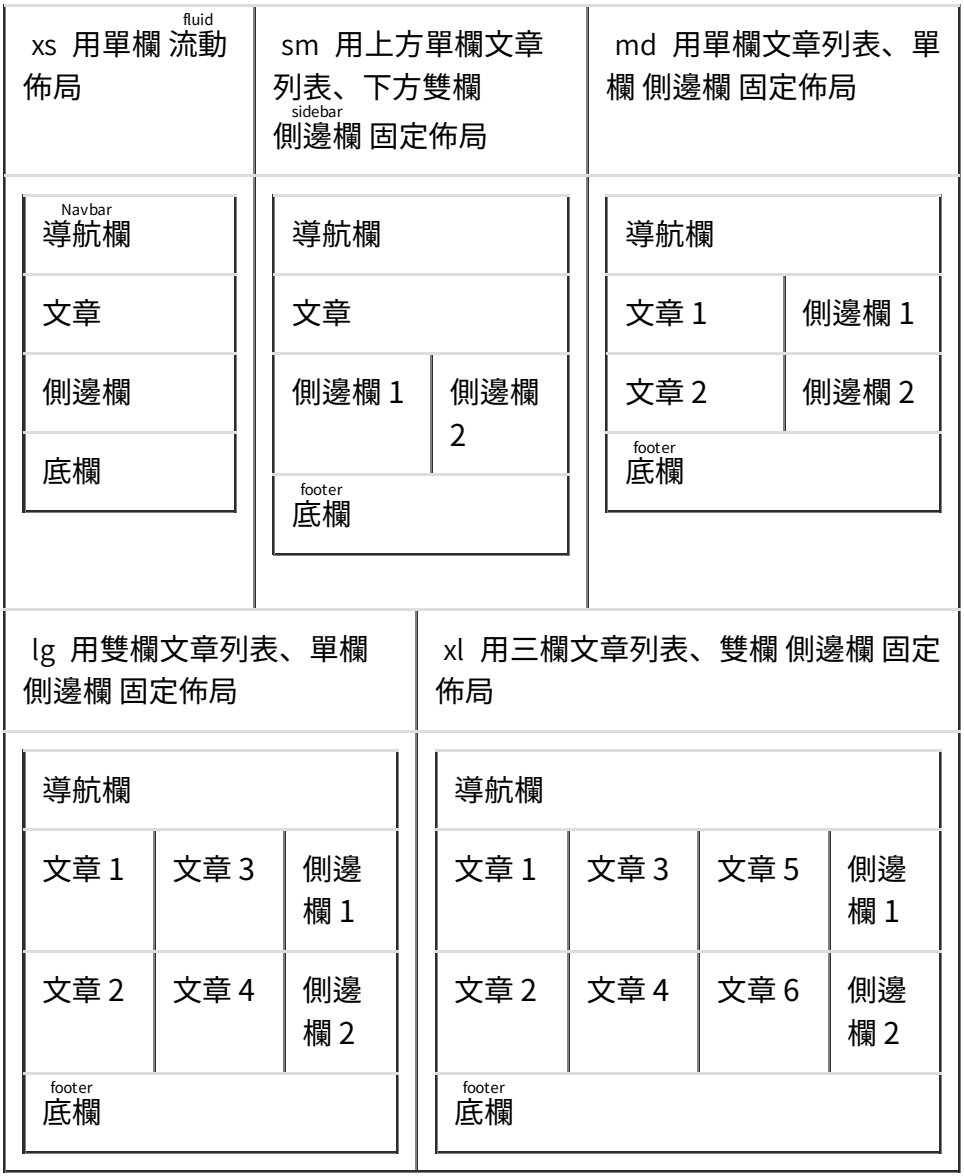
```
@screen-xs:    320px;
@screen-sm:    598px; /*  768px; */
@screen-md:    952px; /*  992px; */
@screen-lg:   1350px; /* 1200px; */
@screen-xl:   2030px;
@container-sm:  582px; /*  750px; */
@container-md:  930px; /*  970px; */
@container-lg: 1320px; /* 1170px; */
@container-xl: 1990px;
```

首先把 Bootstrap 3 默認適配的幾個 響應式設備的大小 改成了我需要的大小。 xs 和 sm 的大小分別按照我的手機屏幕 **豎屏** 和 **橫屏** 時候的瀏覽器頁面寬度來算， md 是想兼容 Nexus 7 橫屏 960 的寬度以及 一個常見上網本 1024 的寬度。 lg 的大小則按照常見的筆記本 1366 寬的屏幕來適配。

這裏 Bootstrap 3 支持的設備大小的一個問題是，它最多考慮到 1200 像素寬的顯示器，而更寬的 比如 1600、 2048 甚至 2560 像素寬的顯示器現在也並不少見，其結果就是頁面中左右兩側 有很大的空間被浪費掉了。作爲深受這一問題困擾的用戶之一，我用 這裏介紹的方法 給 bootstrap 增加了一類「比大更大 (bigger than bigger)」的 xl 響應式設備尺寸，寬度設爲支持 2048 像素寬的顯示器，具體的修改反映在 variables.less 文件裏。

根據寬度自動分欄和瀑布式佈局

接下來目標是讓主頁的文章列表像 Google+ 主頁那樣根據顯示器寬度自動調整分欄，使得寬度不同的 顯示器上每個分欄的寬度接近。想要達到的效果是，根據上面定義的屏幕寬度尺寸：



一開始純粹用 Bootstrap3 的響應式柵格實現這個分欄佈局，結果發現效果不太理想， 因爲文章列表和側邊欄的高度是變化的，會導致柵格間留下大片空白。後來改用 這裏示範的純CSS瀑布式佈局 實現文章和側邊欄的佈局，具體的實現代碼在 waterfall.less ，總算達到了想要的佈局了。

正文的樣式

最最重要的是文章正文的樣式。這裏我想要達到的效果是，在大屏幕上用更大的字號，讓讀者 看起來更舒適，同時在小屏幕上用比較小的字號，最終保證基本上「一行」的文字數接近。這個修改 主要針對 .jumbotron ， 用了 不太科學的方式 代碼太長就不貼全了。

一些細微的定製

把主題配色改成了現在這樣的淡紫色 @brand-primary: darken(#6B5594, 6.5%); ，配合我的頭像風格， 這個修改只需要一行。 接着刪掉了 .btn 的 white-space: nowrap; 讓按鈕的文字可以換行， 這也只是一行修改。

2015年1月29日更新

另外我也不太喜歡 Bootstrap 3 默認在手機上的 摺疊導航欄 (collapsed navbar)， 摺疊之後的操作不夠直觀方便而且依賴 javascript 所以有 bug …… 於是我把它關掉了， 具體方式是在 variables.less 把 @grid-float-breakpoint 和 @grid-float-breakpoint-max 都設爲0就可以了。

對 bootstrap-material-design 的定製

這裏定製的地方不多。原樣式中一個不太科學的做法是所有 .btn 都強制加上了陰影 效果，這在已經有陰影的環境裏用的話非常礙眼，像是 Win9x 風格的厚重睫毛膏。既然可以單獨 給每個樣式加陰影，於是就把 .btn 強制的陰影去掉了，只保留鼠標懸停之後強調的陰影。

其它定製的細節麼就是統一配色風格，修補漏洞錯誤，微調響應式效果而已，這裏不細說。

將以上兩者整合在 pelican-bootstrap3 裏

Pelican 實現顯示源代碼按鈕

顯示源代碼按鈕借用了 Pelican 配置中自帶的 OUTPUT_SOURCES 選項將源文件複製到輸出文件夾：

```
OUTPUT_SOURCES = True
OUTPUT_SOURCES_EXTENSION = '.rst'
```

然後在 Makefile 裏用 pygmentize 把所有源代碼文件着色：

```
find -iname "*.rst" | parallel -I@ pygmentize -f html -o @.html @
```

最後在按鈕按下的時候用 jQuery 載入源代碼：

```
<a onclick="$.get('{{SITEURL}}/{{article.slug}}.rst.html', function(data){$('#source-code').html(data)});$('#article-content').toggle();$('#source-content').toggle();">
```

雖然難看的 hack 比較多，但是能用！

雖說 pelican-bootstrap3 是我 fork 出來的，不過由於我修改的地方實在太多，代碼看來基本上 接近重寫了一份。好在之前有給 pelican 寫 bootstrap 2 主題的經驗，這次修改算得上駕輕就熟。 可以對比一下 上游作者的博客 和這裏的樣子體會一下感覺。 具體修改過的地方包括：

1. 套用 bootstrap-material-design 的各個元素樣式。
2. 在文章列表模板應用上面提到的 Bootstrap 3 的柵格佈局和瀑布式佈局。
3. 翻譯到多個語言，這裏在後面的 i18n-subsite 插件裏詳述。
4. 套用後面會介紹到的各種插件。
5. 統一側邊欄的樣式到一個模板裏。
6. 添加 Atom 訂閱按鈕和 breadcrumb 條。
7. 對正文中出現的插圖，添加點擊放大的功能，通過 Bootstrap 的 modal 實現。
8. 上面提到的用 這個bootstrap插件 讓導航欄自動隱藏。

9. **顯示源代碼按鈕**，也就是每篇文章信息欄中的 </> 按鈕。

插件: 發揮 Pelican 和 reStructuredText 的優勢

先列舉一下我目前用到的所有插件：

```
PLUGINS = ["i18n_subsites",
           "plantuml",
           "youku",
           "youtube",
           'tipue_search',
           'neighbors',
           'series',
           'bootstrapify',
           'twitter_bootstrap_rst_directives',
           "render_math",
           'extract_toc',
           'summary']
```

嗯其實不算多。接下來逐一介紹一下這些各具特色的插件。

i18n-subsites

這個插件的目的是創建 國際化 (internationalization) 子站 (subsite)。

之前介紹 Pelican 配置的時候就提到過， 原本的 Pelican 就支持一篇文章用多種語言書寫，有 lang 屬性註明這篇文章使用的 語言，以及 slug 屬性註明多語言的翻譯之間的關聯，換句話說同一篇文章的多個語

言 版本應該有相同的 slug 和不同的 lang 。然後原本 Pelican 裏對多語言的 實現方式是，首先有一個 **主語言** 是模板和大部分文章採用的語言，文章列表中會優先列出 用 **主語言** 撰寫的文章，然後從 **主語言** 的文章鏈接到別的翻譯版本。 很多博客系統和CMS對多語言的支持都是這樣的，這種處理方式的缺點也顯而易見：作爲 **主語言** 的語言必須足夠通用，纔能讓進來的人找到合適的翻譯版本，所以通常 **主語言** 都是英語。

而這個插件做的事情描述起來很簡單：將文章按語言屬性分到多個子站，每個子站獨立放在各自的文件夾。 比如主站是 https://farseerfc.github.io/ 的話，那麼英語的子站就可以是 https://farseerfc.github.io/en/ 。 然後分別對多個子站生成靜態頁面。具體的實現方式是對 pelican 的頁面生成步驟做了拆分：

1. pelican 按正常情況讀入文章，生成元信息。
2. i18n-subsites 針對每個語言，覆蓋掉 pelican 的一些選項設置比如路徑和 URL ， 分別調用 pelican 的頁面生成器按模板生成文章。
3. 對共用的靜態內容比如模板的 js 和 css 文件，只在主站中生成，子站中的相應鏈接全部鏈回主站。

雖然描述起來簡單，但是這個插件可以說最大化利用了 Pelican 的插件系統，實現細節相對比較 複雜，大概是我用的這些插件裏面最複雜的了。不誇張的說 Pelican 3.4 支持的新插件 API 和 站內鏈接功能基本上就是爲了配合這個插件的。至於具體它會覆蓋哪些 Pelican 的配置，請參閱它的 README.md文件 。

按內容拆分多語言子站的做法只解決了問題的一半，還留下另一半的問題，也即對模板的翻譯。 對這個問題， i18n-subsites 提供了兩套方案供選擇：

1. 用覆蓋配置路徑的方式讓每個子站套用不同的模板。這配置起來簡單，但是對模板維護起來有點困難。
2. 用 jinja2 的 i18n 插件，配合 Python 的 gettext 庫實現內容翻譯。這個方案 配置起來比較複雜 ，但是配置好之後用起來就很方便了。 只是要記得每次修改了模板都要更新翻譯，處理 *.po 和 *.mo 文件等等瑣碎事宜。

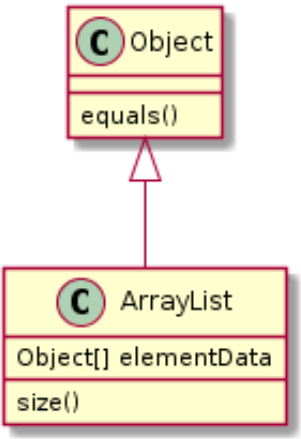
這裏我用 jinja2 的 i18n 插件的方式實現了模板的翻譯， 各個語言的

翻譯在這裏， 然後用 這裏的 SCons 腳本 根據內容是否變化自動更新 po 和 mo 文件。

配置好這一套方案之後，還要注意在模板和文章中處理好鏈接。用 Pelican 3.4 之後推薦的 新的文章間鏈接的寫法以及將 SITEURL 設置爲實際 URL 並且關閉 RELATIVE_URLS 之後，應該就不會出沒什麼問題了（可能還要考慮使用的模板和插件的兼容性，大部分都是寫死了 URL 的問題）。

plantuml

嵌入 PlantUML 的示例



PlantUML 是一個Java實現的， 用接近文字描述的語言繪製 UML 圖或者 GUI 界面圖的工具，非常適合嵌入在 Markdown、reStructuredText、 AsciiDoc 等這種輕量級標記語言裏。 然後麼這個 plantuml 插件就是定義了一個新的 reStructuredText 指示符(directive) .. uml:: ，把嵌入的內容提取出來調用 plantuml 命令處理 成圖像然後再插入到文章中。

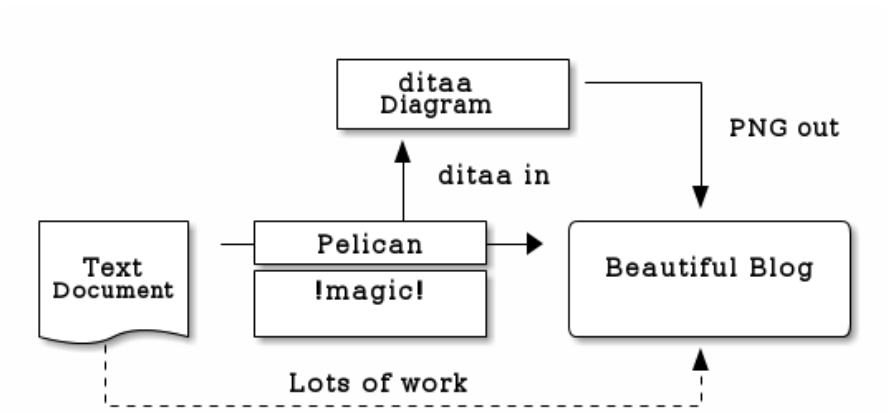
比如示例裏的這個 UML 圖就是用這樣一段簡單的文字描述生成的：

```
.. uml::

    Object <|-- ArrayList

    Object : equals()
    ArrayList : Object[] elementData
    ArrayList : size()
```

實際用起來這個插件實現上稍微有點小問題：首先它只支持 python2，所以我把它改寫成了 python 2 和 3 都通用的語法；其次它原本輸出的文件夾似乎會被 pelican 刪掉，所以把它改了個位置； 然後它輸出的 URL 也和 i18n-subsites 插件間有不兼容的問題，也順帶修掉了。 修改之後的代碼在這裏 。

2015年1月30日更新

嵌入 Ditaa 的示例



plantuml 是繪製UML的，除此之外還有一個類似的工具是繪製一般的 流程圖 (diagram) 的，叫 ditaa ，和 plantuml 非常像，也比較像 reStructuredText 的表格。 於是我也照貓畫虎實現了一個 ditaa 的指示符 (directive)，用起來類似這樣：

```
.. ditaa::

                +-------------+
                |  ditaa    |-------+
                | Diagram   |       |
                +-------------+       | PNG out
                   ^              |
                   | ditaa in       |
                   |              v
     +--------+   +--------+----+    /---------------\
     |        | --+  Pelican  +--> |               |
     | Text   |   +-------------+    | Beautiful Blog|
     |Document|   |  !magic!    |    |               |
     |    {d}|   |           |    |               |
     +---+----+   +-------------+    \---------------/
       :                           ^
       |        Lots of work           |
       +-------------------------------+
```

render-math

嵌入公式的示例

示範行內公式 \(A_\text{c} = (\pi/4) d^2\).

整行公式

\begin{equation*} \alpha{}_t(i) = P(O_1, O_2, \ldots O_t, q_t = S_i \lambda{}) \end{equation*}

這個插件提供在 reStructuredText 中用 LaTeX 語法插入數學公式的能力，定義了 :math: 行內角色 (role) 和 .. math:: 指示符 (directive)。 實際工作的渲染庫當然是大名鼎鼎的 MathJax，這個插件 會用 MathJax 的 CDN 載入，所以也沒有額外的依賴文件。（只是不知道是否會被國內牆掉， 如果公

式顯示不正常請 **務必** 告訴我。）

youtube 和 youku

顧名思義，這兩個插件分別實現嵌入 youtube 和 youku 視頻。其中 youtube 是原本就有的插件， youku 是我照貓畫虎抄的。 之前寫了一篇 KDE5 Plasma 之跳動賣萌的活動按鈕 用到了這兩個插件。

tipue_search

Tipue search 是一個非常有意思也很強大的搜索工具， 通過 jQuery 實現靜態博客的站內搜索功能。實現方式是，它需要你寫一個 json 文件，包含 整個網站的 **全部** 文章的標題和文字內容，然後在搜索的時候讀入這個 json 做搜索（是不是有點耍賴）。 雖然聽起來會有性能問題，但是應用在小型的靜態博客上效果意外很不錯，比如本站的所有文章內容放在一起的 json 也只有 300KiB 左右。

這個插件就是自動在 pelican 輸出完全部靜態網頁之後，調用 beautifulsoup4 從所有網頁中抽取出 純文本，產生這個 json 給 Tipue 用。

neighbors 和 series

這兩個插件比較類似也都比較簡單， neighbors 提供一篇文章的前後文章信息， 在主題模板裏可以用來製作 **上一篇** 和 **下一篇** 按鈕。 series 提供將多篇文章歸類爲一個 **系列** 的支持，當然也需要在 主題模板中定義顯示「文章系列」的列表。這兩個插件的效果都能在本文末尾，評論區上方的部分看到。

bootstrapify 和

bootstrapify 和 twitter_bootstrap_rst_directives

這兩個插件讓文章的 **正文** 套用上 Bootstrap 的樣式。

bootstrapify 這個插件實現得比較簡單，用 beautifulsoup4 在靜態網頁的結果裏面過濾元素， 對 table ，img ，embed ，iframe ，video ，object 這幾個標籤套用上 響應式嵌入對象的類 讓他們更美觀。

twitter_bootstrap_rst_directives 這個插件則是增加了幾個 reStructuredText 的 行內(role)角色 和 指示符(directive) 。 它實現的 行內(role)角色 包括： 用 :kbd: 實現如 Ctrl+C 這樣的鍵盤快捷鍵， 用 :code: 嵌入代碼片段，用 :glyph: 嵌入字符圖標。 它實現的 指示符(directive) 包括： labels 行內標籤 ， alerts 提示段落 ， panels 嵌入面板 ， 以及還有一個 media 混排圖標 。

對其中的 panel 我改寫了它在文章正文中的樣式，在 lg 或者 xl 的屏幕寬度下，分別用 \(\frac{1}{2}\) 和 \(\frac{1}{3}\) 大小的嵌入面板，簡單實現和正文文字的圖文混排。

除此以外我還在 twitter_bootstrap_rst_directives 這個插件裏套用它的框架實現了兩個額外 的 行內(role)角色 ， 分別是 :ruby: ：通過 html5 的 <ruby> 標籤實現文字上方的注音（firefox下 不支持 ，會使用文字後的括號顯示）， 以及 :html: ：在 行內插入 裸(raw) html 標籤（這屬於 Markdown 的基本功能，在 reStructuredText 這邊由於要考慮多種輸出格式於是就比較麻煩了）。這兩個 行內(role)角色 的 實現代碼在這裏 。

2015年2月3日更新

今天又在 twitter_bootstrap_rst_directives 裏增加了兩個 行內(role)角色 。 一個是 :twi: 用來寫 twitter 用戶的鏈接，比如 @farseerfc ，另一個是 :irc: 用來指向 freenode 的 channel ，比如 #yssyd3 。

2015年2月14日更新

今天增加了 .. friend:: 用來寫好友鏈接，以及 fref 用來引用好友，比如 LQYMGT 這樣。

extract_toc 和 summary

最後是這兩個有點「名不副實」的插件。

reStructuredText 原本就有自動生成目錄(toc)的功能，用起來也非常簡單，只需要在想要插入目錄的地方寫一行 .. contents:: ，剩下的都由 docutils 自動生成了。 只是當然這樣生成的目錄肯定會插入在文章的正文裏，而 extract_toc 這個插件的作用就是簡單地 把這個目錄抽取出來，讓模板能在別的地方放置這個目錄。比如我這裏就把目錄放在了一個 panel 裏。

然後 Pelican 也原本就有從文章中抽取總結(summary)顯示在文章列表的功能。 Pelican 原始的實現似乎是按照文字數抽取前半段，不總是適合作爲總結。 於是這個 summary 插件的作用其實是允許在正文中以特殊的註釋的方式標註哪些部分應該被抽出來作爲總結。 summary 這個插件原本的實現只允許抽取一段文字，我又對它的實現做了少許擴充，允許標註多段 文字合併起來作爲總結。

2015年1月29日更新

今天在 extract_toc 插件的幫助下，在側邊欄裏放了一個 Bootstrap affix 的目錄， 它保持在頁面的右側位置不變，方便導航到文章的各個地方。具體實現方法除了 Bootstrap 3 的 Affix 文檔 ，還參考了 這篇更詳細的說明 。

結語

這個博客的配置都可以在 github 上找到 ，包括用來 自動生成整個博客的 Makefile ，由於比較長，這裏就不再貼了。

折騰這個主題前後歷時兩個月，期間學會了不少東西，也算是不錯的收穫吧。 現在既然基礎打好了，接下來就要開始多寫博客了。（希望拖延症不會再犯……）

最近發現除了我的博客之外還有一個網站 Kansas Linux Fest fork 了我的主題，不過他們用了我修改的早期版本，還是原本的 Bootstrap 3 和

bootstrap-material-design 樣式。自己草草修改的東西被別人用到果然還是有點小激動呢， 以及接下來不能馬馬虎虎地寫 commit 消息了。

[1] 賽65:17「看哪！我造新天新地」啟21:5「我將一切都更新了。」