ngn -novel page generator-

@subaru45

2013-12-27

1 なんぞや

ngn は簡単なテンプレートエンジンです。テキストファイル中のタグ (後述) 文字列で、テンプレートファイル中のタグ部分を置換したファイルを出力します。

開発目的はウェブサイト用の小説・文章ページを自動生成することですが、ほかの用途にも使えなくもないです。

出力形式はテンプレートファイルの拡張子で判別します。が、現状では html 以外はガン無視します。

ngn は UTF-8、EUC-JP、SJIS (CP932) の三種類の文字コードを扱うことができます。改行コード三種もばっちり扱えます。出力ファイルの文字コードと改行コードは、テンプレートファイルの文字コード・改行コードになります。つまり **Windows も Linux も OSX もばっちり** ということです。

1.1 将来的にやること

- LaTeX 出力サポート
- 出力形式を自前で追加可能に

1.2 うんぬん

導入の時点でそもそもハードルが高いですが、ngn は NYSL (煮るなり焼くなり好きにしろライセンス)[*1] を採用しています。NYSL の元で許される限りにおいて、如何様にも改変し配布し破棄することができます。

2 導入

プラットフォームによって導入方法は違います。

- Windows 使い
  bitbucket の下記ダウンロードページ
  http://bitbucket.org/subaru45/ngn/downloads
  から “win” の文字のついたファイルをダウンロードしてください。後は展開してできたフォルダの

[*1] http://nysl.com

ngn.exe をパスの通ったディレクトリに置くなりしてください。
GUI フロントエンドつくったほうがいいかなあ……。

- OSX 使い
  ぼくは Mac 持ってないので、自分でバイナリつくってください。Mac なら App Store に Clozure CL があったと思います[*2]。バイナリをつくったらパスの通ったところに置いて使ってください。あるいは wine で win 版動かすとか……。

- *nix 使い
  自分でバイナリつくってください。バイナリをつくったらパスの通ったところに置いて使ってください。あるいは wine で win 版動かすとか……。

2.1 バイナリをつくるには

ngn の導入には Common Lisp 処理系 (開発環境は Clozure CL) と quicklisp[*3] と shelly[*4] と guess[*5]が既に導入されていることを前提とします (導入方法は 4 節参照)。

実行可能バイナリの生成は shelly を利用します。Windows では shelly がうまく動かない感じ[*6]なので、shelly を使わない方法も書いておきます。

2.1.1 shelly ありのバイナリづくり

1. ソースの取得
   下記プロジェクトからソースを取得します:
   http://bitbucket.org/subaru45/ngn

2. 実行可能バイナリの作成
   取得したソースのディレクトリに入り

```
shly save-app
```

   を実行します。

2.1.2 shelly なしのバイナリづくり

1. ソースの取得
   下記プロジェクトからソースを取得します:
   http://bitbucket.org/subaru45/ngn/

[*2] https://itunes.apple.com/jp/app/clozure-cl/id489900618
[*3] Common Lisp 版 CPAN のようなもの。http://quicklisp.org/beta
[*4] シェルから Common Lisp のコードをさくっと実行できるユーティリティ。http://github.com/fukamachi/shelly
[*5] 文字コード判定ライブラリ。http://github.com/t-sin/guess
[*6] cygwin 下で CPAN のライブラリインスコに失敗したので諦めた。Perl わからんぬ。

2. 実行可能バイナリの作成

(a) 処理系の起動

取得したソースのディレクトリに入り処理系を起動します。

(b) バイナリの作成

次のコードを入力します:

```
(load "guessのパス/gues.asd")
(load "ngn.asd")
(ql:quickload :ngn)
(gc)
(ccl:save-application
  "ngn"  ;; windowsでは"ngn.exe"
  :toplevel-function #'ngn:app
  :prepend-kernel t
  :purify t)
```

成功なら処理系は自動的に終了します。

(c) ディレクトリに ngn (windows なら ngn.exe) ができていれば OK

3 つかいかた

ngn の使い方を説明します。

まず、出力したいテキストファイル中の文字列にタグを付けます。タグが付されたテキストファイルを**タグ付きファイル**と呼びます。タグ付きファイルにおいて、あるタグ spam が付けられた文字列を **spam のテキスト**と呼びます。次にテンプレートを用意し、ファイル中に出力したいタグ指定子を記述しておきます。最後に、タグ付きファイルとテンプレートを ngn に食わせると、テンプレートのタグ指定子の箇所に、そのタグのテキストが挿入されたファイルができあがります。

タグは書式さえ満たしていればいくつでも好きなように定義できます。もしテンプレート中にタグ付きファイルにないタグを指定した場合、そのタグ指定子の箇所には空文字列が挿入されます。

3.1 タグ付きファイル

まず、タグを付けたテキストファイルを作成します。

タグには一行タグとブロックタグの二種類があります。二種ともに、**行の先頭に**コロン ':' とタグ名を記述するという形式は同じです。タグ名に利用できる文字はアルファベット小文字 a-z、数字 0-9、ハイフン'-' のみです。

3.1.1 一行タグ

一行タグは、改行を含まない文字列を記述するのに用います。例えば、題名、著者名、日付などです。LaTeX でいうところの \title や \section 等のようなものだと考えてください (LaTeX の上記コマンドは改行も含められますが……)。

一行タグの書式は以下です。

ソースコード 1 一行タグの書式

```
:tag-name 文字列...
```

タグ名の後ろのスペースは一つです。それ以降は改行までのスペースを含むすべての文字がタグ名に対応するテキストデータとなります。

3.1.2 ブロックタグ

ブロックタグは、改行を含む文字列を記述するのに用います。例えば、本文、後書き、説明などです。文書構造を記述するのに便利でしょう。

ブロックタグの書式は以下です。

ソースコード 2　ブロックタグの書式

```
:tag-name[
文字列1...
文字列2...
...
文字列n...
:tag-name]
```

タグの開始・終了は、タグ名の後ろに括弧をつけて表します。開始は '['、終了は ']' です。カッコの直後で改行してください。ブロックタグ内にタグの記述がされていた場合、タグではなくただの文字列として処理されます。

3.1.3 注意事項

タグ付きテキストファイル中では、現状、行頭でコロンを使用できません (エスケープシーケンス未実装のため)。

同名のタグが複数存在した場合、ファイルの先頭に近いものが保持され、それ以降の同名タグは無視されます。例えば、下のタグ付きテキストファイル例の author タグのデータは「夏目漱石」となり「NATSUME Souseki」とはなりません。

ソースコード 3　タグ付きテキストファイル例

```
:title 吾輩は猫である
:author 夏目漱石

:body[
一

   吾輩（わがはい）は猫である。名前はまだ無い。
   どこで生れたかとんと見当（けんとう）がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪（どうあく）な種族であったそうだ。この書生というのは時々我々を捕（つかま）えて煮（に）て食うという話である。しかしその当時は何という考もなかったから別段恐しいとも思わなかった。ただ彼の掌（てのひら）に載せられてスーと持ち上げられた時何だかフワフワした感じがあったばかりである。掌の上で少し落ちつい て書生の顔を見たのがいわゆる人間というものの見始（みはじめ）であろう。この時妙なものだと思った感じが今でも残って いる。第一毛をもって装飾されべきはずの顔がつるつるしてまるで薬缶（やかん）だ。その後（ご）猫にもだいぶ逢（あ）ったがこんな片輪（かたわ）には一度も出会（でく）わした事がない。のみならず顔の真中があまりに突起している。そうしてその穴の中から時々ぷうぷうと煙（けむり）を吹く。どうも咽（む）せぽくて実に弱った。これが人間の飲む煙草（たばこ）というものである事はようやくこの頃知った。
:body]


:author NATSUME Soseki

:description[
慶応3年1月5日（新暦2月9日）江戸牛込馬場下横町に生まれる。本名は夏目金之助。帝国大学文科大学（東京大学文学部）を卒業後、東京高等師範学校、松山中学、第五高等学校などの教師生活を経て、1900年イギリスに留学する。帰国後、第一高等学校で教鞭をとりながら、1905年処女作「吾輩は猫である」を発表。1906年「坊っちゃん」「草枕」を発表。1907年教職を辞し、朝日新聞社に入社。そして「虞美人草」「三四郎」などを発表するが、胃病に苦しむようになる。1916年12月9日、「明暗」の連載途中に胃潰瘍で永眠。享年50歳であった。
:description]
```

3.2 テンプレートファイル

出力ファイルの雛形を作ります。これは普通の HTML ファイルですが、タグのデータをどこに流し込むのか記述しておく必要があります。

テンプレートファイル中のタグの指定は

ソースコード 4　タグ指定の書式

```
#|tag-name|#
```

の書式で行います。これを書いた箇所がそのまま、そのタグのデータに置換されます。

もし指定したタグが存在しない場合、空文字列になります。

ソースコード 5　テンプレートファイルの例

```
<html lang="ja">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>- #|title|# -</title>

<style type="text/css">
</style>

</head>

<body>
<h1> #|title|# </h1>
<section id="author">
<h2>#|author|#</h2>
<br>
#|description|#
</section>

<section id="body">
#|body|#
</section>

</body>

</html>
```

3.3 ngn コマンド

タグ付きテキストファイルとテンプレートファイルの用意ができたら、ファイルを生成します。生成には ngn コマンドを用います。

ソースコード 6　ngn コマンドの使い方

```
  ngn [input-filepath] [template-filepath]
```

[input-filepath] はタグ付きテキストファイルのパス、[template-filepath] はテンプレートファイルのパスです。この二引数は必須です。引数が足りない、または存在しないファイルだった場合はメッセージを吐いて何もせず終了します (たぶん)[*7]。

タグ付きテキストファイルの内容が残念だった場合の挙動はわかりません。ちゃんと不正な入力は原因ごと表示するように、いつか改良します。

[*7] その場合の終了コードは 1 だったはず

処理結果のファイルはカレントディレクトリに出力されます。つまり、別のディレクトリに存在するファイルを引数に指定しても、処理結果はターミナルで ngn コマンドを実行したディレクトリに作成されます。もし同名のファイルが存在した場合は **上書きせず、標準出力に**処理結果を出力します。コマンドを叩いたらターミナルに HTML なりがずらずら表れた場合は、同名ファイルが存在したということになります。

処理結果のファイル名は [input-filepath] の拡張子を [template-filepath] の拡張子に置き換えたものになります。例えば、カレントディレクトリが /home/sora のとき、 [input-filepath] を /home/sora/text/wonder2.txt 、[template-filepath] を /home/sora/web/temp.html をそれぞれ引数として ngn を実行すると、出力ファイル名は wonder2.html となり、/home/sora に出力されます。

前 2 節でサンプルとしたファイルを引数に ngn を実行すると、次のような出力が得られます。

ソースコード 7　生成されたファイル

```
<html lang="ja">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>— 吾輩は猫である —</title>

<style type="text/css">
</style>

</head>

<body>
<h1> 吾輩は猫である </h1>
<section id="author">
<h2>夏目漱石</h2>
<br>
慶応3年1月5日（新暦2月9日）江戸牛込馬場下横町に生まれる。本名は夏目金之助。帝国大学文科大学（
    東京大学文学部）を卒業後、東京高等師範学校、松山中学、第五高等学校などの教師生活を経て
    、1900年イギリスに留学する。帰国後、第一高等学校で教鞭をとりながら、1905年処女作「吾輩は猫
    である」を発表。1906年「坊っちゃん」「草枕」を発表。1907年教職を辞し、朝日新聞社に入社。そ
    して「虞美人草」「三四郎」などを発表するが、胃病に苦しむようになる。1916年12月9日、「明暗
    」の連載途中に胃潰瘍で永眠。享年50歳であった。
</section>

<section id="body">
一<br>
<br>
   吾輩（わがはい）は猫である。名前はまだ無い。<br>
   どこで生れたかとんと見当（けんとう）がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いて
      いた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは
      書生という人間中で一番獰悪（どうあく）な種族であったそうだ。この書生というのは時々我々を捕
      （つかま）えて煮（に）て食うという話である。しかしその当時は何という考もなかったから別段恐
      しいとも思わなかった。たただ彼の掌（てのひら）に載せられてスーと持ち上げられた時何だかフワフ
      ワした感じがあったばかりである。掌の上で少し落ちついて書生の顔を見たのがいわゆる人間と
      ものの見始（みはじめ）であろう。この時妙なものだと思った感じが今でも残っている。第一毛をも
      って装飾されべきはずの顔がつるつるしてまるで薬缶（やかん）だ。その後（ご）猫にもだいぶ逢（
      あ）ったがこんな片輪（かたわ）には一度も出会（でく）わした事がない。のみならず顔の真中があ
      まりに突起している。そうしてその穴の中から時々ぷうぷうと煙（けむり）を吹く。どうも咽（む）
      せぽくて実に弱った。これが人間の飲む煙草（たばこ）というものである事はようやくこの頃知った
      。<br>
</section>

</body>

</html>
```

4　導入の導入

Windows 以外のプラットフォームでは導入の敷居が異常に高くて ggrks ではちょっとヒドいので、必要なものの導入方法を書いておきます。

4.1 Clozure CL の導入

下記 URL
http://ccl.clozure.com/download.html
から自分の OS・アーキテクチャのアーカイブファイルをダウンロードして、展開してください。展開したらそのフォルダにパスを通してください。

以上、導入完了。

末尾が “cl” になっている (windows では “cl.exe”) ファイルが処理系のインタプリタです。以降説明中に「処理系を実行」とか書かれていたらこいつを実行してください コマンドラインで)。

4.2 quicklisp の導入

下記のファイル
http://beta.quicklisp.org/quicklisp.lisp
をダウンロードして処理系のフォルダ (Clozure CL なら cclx.y, x と y は何か数字) に置きます。コマンドラインで処理系のフォルダに移動したあとインタプリタを起動して、次のようにコード打ちましょう。

```
(load "quicklisp.lisp")
(quicklisp-quickstart:install :path "./")
(ql:add-to-init-file)
```

以上、導入完了。

4.3 shelly の導入

処理系のインタプリタを立ち上げて次のコードを打ちましょう:

```
(ql:quickload :shelly)
```

以上、導入完了。

4.4 guess の導入

下記 URL
http://beta.quicklisp.org/quicklisp.lisp
からソースを一式取得して展開し、処理系のフォルダに展開してできたフォルダごと置いてください。

以上、導入完了。

5 あぺんでぃっくす

開発者向けな情報。あるいは仕様メモ。自分用。

5.1 タグ付きテキストファイルの仕様

タグ付きテキストファイルの仕様を示します。¡oneline-tag¿ と ¡block-tag¿ の部分が重要です。

ソースコード 8　タグ付きテキストファイル

```
<text> ::= <line>*
<line> ::= <oneline-tag> | <block-tag> | <plain-line>

<oneline-tag> ::= <tag-delimiter> <tag-identifer> <space> <data> <eol>

<block-tag> ::= <block-tag-open> <plain-line>* <block-tag-close>
<block-tag-open> ::= <tag-delimiter> <tag-identifer> <block-delimimter-open> <eol>
<block-tag-close> ::= <tag-delimiter> <tag-identifer> <block-delimimter-close> <eol>

<tag-identifer> ::= a-z0-9- (lower alphabets, numerics and hyphen)
<tag-delimiter> ::= :
<block-delimiter-open> ::= [
<block-delimiter-close> ::= ]

<plain-line> ::= .* <eol> (any strings s.t. ending with <eol>)
<eol> ::= \n
```

5.2 タグについてのフック

ngn をさらに拡張する人向けの情報です。

ngn では、抽出したタグをテンプレートに挿入する前に、抽出したタグに対してフック関数を実行するようになっています。これにより、抽出されたタグの中にさらに別の DSL が記述されている場合、それに処理を施してから、出力を行うことができます。

たとえば、タグのデータ内に Markdown などの 言語を埋め込んでおき、タグ抽出後にそれらを出力前に処理してしまうといった用途に使えます。

フック関数は ngn のメイン処理を行っている /src/ngn.lisp の ngn:ngn 関数のキーワード引数 tag-hook に渡してください。tag-hook はタグのリストを受け取り、タグのリストを返すように記述してください。

/* あ。ngn.generator を外部ファイルで拡張可能にするわけだし、このフックは実はいらない気がしてきた。いらないよなあ……。あとで消えるかも。*/

出力形式に依らない変換を行うのに用いてください。たとえば、

```
(times 10 コワイ)
```

を

```
コワイコワイコワイコワイコワイコワイコワイコワイコワイコワイ
```

に展開するとか、

```
(times 5 (random-pickup "死シし" "ぬヌ") (nl))
```

を

```
シ ヌ
死 ぬ
シ ぬ
し ぬ
し ヌ
```

に展開するとか (nl は new line)、そういう使い方。今のところ必要性は全くないけど。

5.2.1 タグのデータ構造

タグのデータ構造です。この構造を保つよう tag-hook を書いてください。

ソースコード 9　タグのデータ構造

```
<tags> ::= ( <tag>* )
<tag> ::= ( <oneline> | <block> )

<oneline> ::= <tag-identifer> <string>
<block> ::= <tag-identifer> <string-list>

<string-list> ::= ( <string>*)
<string> ::= Common Lisp の文字列
<tag-identifer> ::= Common Lisp のキーワード
```