インターフェイスの街角 (67) – Web ページの鮮度を視覚化する

増井 俊之

情報が紙に印刷されている場合には、インクや紙は時間の経過とともに劣化していきます。一方、Web ページ上の情報は時間が経っても外観が変化しないため、旧いページも新しいページも一見したところは同じです。このため、Web ページに書かれたまま、その後まったく手が加えられていない旧い情報を新しいものと勘違いすることがよくあります。

多くのニュースサイトや日記ページ、情報サイトなどでは日付が明記されているため、情報の新旧を取り違えることはめったにありません。しかし個人が趣味で作っているようなページでは、日付がきちんと表示されていない場合も珍しくないので、いつの時点の情報なのかが分からずに混乱してしまうこともあります。

Wiki Wiki Web のように頻繁に編集がおこなわれる Web ページの場合、たいていは編集した日付がページごとに記録されています。しかし、ページのどの部分が更新されたのかは、明示的に記述されていないかぎり分からないのが普通です。旧い情報が書かれたページの表記をすこし変えたり、ちょっとだけコメントを加えたようなときも、ページ全体の日付が更新されてしまいます。このような場合、ページ内の旧い部分と新しい部分が簡単に区別できるように視覚化されていれば便利でしょう。

また、ページの旧さだけでなく、アクセス状況も同時に視覚化できれば、

- 旧くても、よくアクセスされている情報（重要な情報）
- 旧いうえに、まったくアクセスされていない情報（重要でない情報）
- 新しい情報（アクセスが多いか、少ないかは不明）

といった区別をすることもできます。

情報の鮮度の視覚化

Web ページ上の情報の鮮度（新旧）を表現する方法は、いろいろと考えられます。

たとえば、慶應義塾大学の塚田浩二氏らが開発した〝廃れるリンク〟[1]の場合、Internet Explorer に対応した各種のフィルタをテキストや画像に適用し、情報が旧いほど文字や画像を読みにくくして、ページの内容やリンク先の情報の新旧を表現しています（図 1）。この手法では、レイアウトやバックグラウンド画像には手を加えず、文字の表示形式だけで新旧を表現しているため、どのような Web ページでも鮮度を視覚化できます。その反面、情報の作成日時のみを基準として視覚化しているため、人気が高く、しばしばアクセスされるページであっても、作成されてから時間が経てば経つほど廃れてみえてしまいます。

廃れるリンクのアイデアは継承しつつ、人気（アクセス状況）も反映させるために、以下のような方針で視覚化を考えてみました。

- ひさしぶりに旧いページにアクセスすると、古色蒼然とした見にくいページが表示される。
- リロードなどによってアクセスを増やすと多少見やすくなるが、旧いという情報は残る。

これを実現するために、旧さをテクスチャで、アクセス状況を明るさで表現したバックグラウンド画像を使うことにします。〝旧さをテクスチャで表現する〟と言われてもピンとこない人が多いと思いますが、今回はとりあえず次のような方法を適用してみることにしました。

- 作成の日付を表現する画像を使う。

図 1　廃れるリンク

図 2　2003 年 5 月 10 日を表現するバックグラウンド画像

- アクセス状況から算出した〝人気度〟を明度で表現する。

これは、次のようにして実現します。

- 日付を表現する画像を用意する
旧さが直感的に感じられるように、作成した日付を表現する文字列に染みのついたような画像を使います。たとえば、2003 年 5 月 10 日を表現するには、図 2 のような画像を使います。
- アクセス履歴にもとづくバックグラウンド画像の生成
上記のバックグラウンド画像を用いて、〝人気度〟にもとづいた視覚化をおこないます。たとえば、2003 年 5 月 10 日 12 時 34 分 56 秒に編集したテキストは、以下のようなタグで囲むことにより、バックグラウンド画像を表示します（誌面の都合上、⇒ で折り返しています）。

```
<div style="background-image: ⇒
url(weather.cgi?id=20030510123456)">
   ......
</div>
```

CGI プログラム weather.cgi は、アクセス履歴にもとづいてこの部分の人気度を計算し、さらに人気度から明度を計算してバックグラウンド画像とします。

これらの方針の実現手法を以下に示します。

バックグラウンド基本画像の生成

情報の作成年月日をうまく表現するバックグラウンド画像についていろいろと考えてみたのですが、適当なものを思いつかなかったので、とりあえずは図 2 のような画像で我慢することにしました。こういった画像をいちいち手作業で作るのは大変ですが、ビットマップ画像エディタとしてひろく使われている GIMP を利用すれば、簡単に自動生成できます。

GIMP は、GTK で作成された UNIX 上のビットマップ画像編集ツールで、最近は Windows や Macintosh などでも使えます。フリー・ソフトウェアでありながら、Photoshop などの高価なソフトウェアに匹敵する高度な画像処理がおこなえます。さらに、自前のプラグインの追加や、〝Script-Fu〟というスクリプト言語を用いた処理の自動化が可能といった大きな特徴もあります。SIC (Synergistic Image Creator)[1] のような高度なプラグインを使えば、写真を芸術作品であるかのように自動変換することも可能です。

Script-Fu は GIMP の処理を記述できる Scheme ふうのスクリプト言語で、GUI で操作可能な編集作業などをすべてプログラムで実行することができます。GIMP には、〝コーヒーの染み〟という効果を表現するスクリプト (coffee.scm) や、画像をセピア色に変換する〝旧い写真〟効果を表現するスクリプト (old_photo.scm) が標準で用意されています。そのための関数も定義されているので、年月日を表現する画像に対してこれらの関数を呼ぶだけで図 2 のような画像が生成できます。

日付文字列からバックグラウンド画像を生成するために、Ghostscript と Script-Fu スクリプト weather.scm（リスト 1）を利用します。この部分の処理は、Ruby で書いたライブラリ weather.rb（末尾のリスト 2）で実現しています。まず、日付文字列を PostScript ファイルに変換し、これを Ghostscript で画像ファイルにしたあと、GIMP を用いて図 2 のようなバックグラウンド画像に変換します。

人気度にもとづく画像処理

Web ページの詳細なアクセス状況を視覚化してもあま

1 http://www.dsn.t-kougei.ac.jp/cp/kyouin/kasao/kanse20000.htm

リスト 1　バックグラウンド画像生成スクリプト weather.scm

```
#
# 入力画像inに対し、"コーヒーの染み"と"旧い写真"のフィルタを適用してoutに格納する
#
(define (weather in out)
  (let* ((image (car (gimp-file-load 1 in in ))) drawable)
    (set! drawable (car (gimp-image-active-drawable image)))
    (and (= (car (gimp-drawable-is-rgb drawable)) 0)
         (gimp-convert-rgb image))
    ;;(gimp-display-new image) # バッチ処理をしない場合
    # "コーヒーの染み"効果
    (script-fu-coffee-stain image drawable 4 nil)
    # "旧い写真"効果
    (script-fu-old-photo image nil TRUE FALSE TRUE FALSE FALSE)
    (set! drawable (car (gimp-image-active-drawable image)))
    # "継ぎ目なしタイル"効果
    (plug-in-make-seamless 0 image drawable)
    (set! drawable (car (gimp-image-active-drawable image)))
    (gimp-file-save 1 image drawable out out)
    (gimp-quit 1)
    ))
```

図 3　作成年月日を指定した HTML ファイル

```
<div style="background-image: url(weather.cgi?id=20020209123456)">
この部分は<p>2002/2/9に<p>作成した<p>部分です
</div>
<p>
<div style="background-image: url(weather.cgi?id=20021031123456)">
こちらの部分は<p>2002/10/31に<p>作成した<p>部分です
</div>
```

り意味があるとは思えないので、以下のように簡単な方法で人気度を表現することにしました。

- 人気度は、時間の経過とともに指数的に減少する。
- アクセスがあると人気度が何割か回復する。

たとえば、1 年経つと人気度が半分になり、アクセスがあると失われた人気度が半分回復する設定にしたとしましょう。この場合、ページが作成されたときの人気度を 1.0 とすると、2 年後の人気度は 0.25 になります。ここで 1 回アクセスがあると、人気度は $0.25+(0.75/2)=0.625$ に回復し、さらに続けて何回もアクセスされると人気度が 1.0 に近づくことになります。最終アクセス日とそのときの人気度だけを記憶しておけば、このような計算をおこなうことができます。

この結果、誰もアクセスしていない旧いページを見ると人気度が低いため染みのある暗い画像がバックグラウンドに表示され、リロードすると人気度が回復して明度が上がり、バックグラウンド画像が白っぽくなります。このような計算は、リスト 3 の weather.cgi でおこないます。

人気度にもとづく明度の変換は、weather.rb のなかで、ImageMagick パッケージに含まれる convert コマンドを使用しています。

結果

Web ブラウザで図 3 のような HTML ファイルを参照すると、アクセス状況に応じて、⟨div⟩〜⟨/div⟩ で囲まれた部分ごとに異なるバックグラウンド画像が生成・表示されます。図 3 では、上半分が 2002 年 2 月 9 日に、下半分が 2002 年 10 月 31 日に作成されたと想定しています。その後、1 回もアクセスされなかった場合は、旧い部分ほど人気度が低くなっているためにページの表示は図 4 のようになります。

ここで画面を 1 回だけリロードすると、人気度が回復して表示は図 5 のように変わります。

もう 1 回リロードすると人気度がさらに回復して表示

図 4　ページ作成後、1 回もアクセスされていない場合の表示

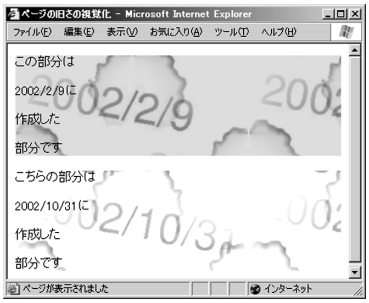


図 5　その後、1 回アクセスされたあとの表示

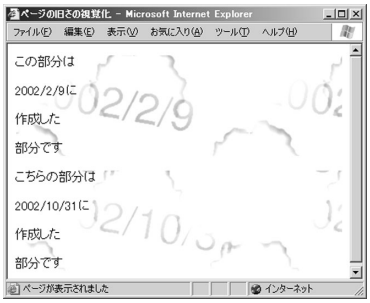
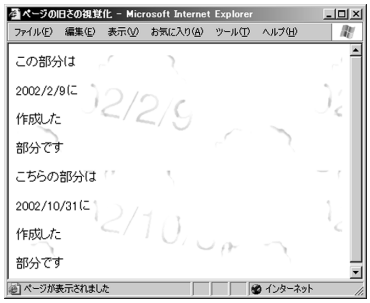


図 6　さらに、もう 1 回アクセスがあったあとの表示



が図 6 のように変わり、バックグラウンド画像はほとんど気にならない程度に明るくなります。

おわりに

Web ページを作成するたびに日付をタグで指定するのは面倒ですが、Wiki Wiki Web のページなどでは自動的にタグを付けることもできます。これを利用すれば、情報の鮮度を比較的簡単に指定できるでしょう。

鮮度だけではなく、その他の属性情報も使ってファイルのアイコンや Web ページを視覚化すれば、ファイルや Web ページをより直感的に把握できるようになるかもしれません。

（ますい・としゆき　産業技術総合研究所）

[参考文献]

[1] 塚田浩二、高林 哲、増井俊之「廃れるリンク」、情報処理学会論文誌、Vol.43, No.12, pp.3,718–3,721、2002 年 12 月

リスト 2　バックグラウンド画像生成ライブラリ weather.rb

```
require 'ftools'

class Weather
  GS = "/usr/bin/gs"
  CONVERT = "/usr/X11R6/bin/convert"
  GIMP = "gimp"

  def initialize(t=Time.new, cachedir="images")
    @cachedir = cachedir
    @t = t
    @year = @t.year
    @month = @t.month
    @day = @t.day
  end

  def dateid
    @t.strftime("%Y%m%d")
  end

  def datestr
    "#{@year}/#{@month}/#{@day}"
  end

  def cachefile
    "#{dateid}.png"
  end

  def cachepath
    "#{@cachedir}/#{cachefile}"
```

```
  end

  def create_postscript(psfile)
    File.open(psfile,"w"){ |f|
      f.puts "%!PS-Adobe-3.0"
      f.puts "10 100 translate"
      f.puts "-10 rotate"
      f.puts "0 0 moveto"
      f.puts "0.6 setgray"
      f.puts "/Helvetica findfont"
      f.puts "40 scalefont setfont"
      f.puts "(#{datestr}) show"
      f.puts "showpage"
    }
  end

  def postscript2image(psfile,imagefile)
    system "#{GS} -dBATCH -dNOPAUSE -sDEVICE=jpeg -sOutputFile=tmp.jpg -r100 -g300x200 #{psfile}"
    system "#{CONVERT} tmp.jpg #{imagefile}"
  end

  def weather_image(infile,outfile)
    system"#{GIMP} -i -b '(begin (load \"weather.scm\") (weather \"#{infile}\" \"#{outfile}\"))'"
  end

  def modify_contrast(infile,outfile,level)
    system "#{CONVERT} -modulate #{level} #{infile} #{outfile}"
  end

  def create_cache
    unless test(?d,@cachedir) then
      File.makedirs(@cachedir)
    end
    unless test(?f,cachepath) then
      create_postscript("tmp.ps")
      postscript2image("tmp.ps","tmp.png")
      weather_image("tmp.png",cachepath)
    end
  end

  def output(level=100,outfile=nil)
    create_cache
    if outfile then
      modify_contrast(cachepath,outfile,level)
    else
      modify_contrast(cachepath,"tmp.png",level)
      File.open("tmp.png","r"){ |f|
        $stdout.write(f.read)
      }
    end
  end
end
```

リスト 3　人気度をもとにバックグラウンド画像の明度を変換して表示する CGI プログラム weather.cgi

```
#!/usr/bin/env ruby

require 'cgi'
require 'weather.rb'
```

```
require 'sdbm'

N = 0.5 ** (1.0/(365 * 24 * 60 * 60))

def id2time(id)
  id =~ /(\d\d\d\d)(\d\d)(\d\d)(\d\d)(\d\d)(\d\d)/
  Time.local($1.to_i,$2.to_i,$3.to_i,$4.to_i,$5.to_i,$6.to_i)
end

def time2id(t)
  t.strftime("%Y%m%d%H%M%S")
end

def access(id)
  db = SDBM.open('list')
  # 前回計算時の時刻と人気度を取得
  if db[id] then
    (lastid,lastval) = db[id].split
  else
    lastid = id
    lastval = 1.0
  end

  # 現在の人気度の計算
  curval = lastval.to_f * (N ** (Time.now - id2time(lastid)))

  # アクセス後の人気度の計算
  newval = curval + (1.0 - curval) * 0.5

  # 時刻と人気度の格納
  db[id] = "#{time2id(Time.now)} #{newval}"
  db.close
  curval
end

# 人気度を明度に変換
def val2brightness(val)
  # 1.0 => 180, 0.0 => 50
  (50 + val * (170 - 50)).to_i
end

def main
  cgi = CGI.new('html3')
  id = cgi['id']
  t = id2time(id)

  print "Content-Type: image/png\r\n\r\n"

  w = Weather.new(t)
  b = access(id)
  w.output(val2brightness(b))
end

main
```