

nginx\_small\_light

久保達彦

[bokko@pixiv.com](mailto:bokko@pixiv.com)

# 自己紹介

- ♦ 久保 達彦(bokko)
- ♦ @cubicdaiya(twitter, github)
- ♦ Senior Software Engineer@pixiv Inc.

◎主な担当分野

○ミドルウェアの開発・運用

○インフラ

○その他色々



# 最近の活動

◎自分で開発・公開中

○ ngx\_small\_light

○ ngx\_access\_token

○ neoagent(Memcached Proxy)

◎開発に参加・パッチ提供等

○ ngx\_mruby

○ mod\_small\_light

○ nginx



etc....

nginx\_small\_light

# ngx\_small\_light

- ◆ nginxモジュール
- ◆ nginxで動的なサムネイル生成ができる
- ◆ mod\_small\_lightをNginxに移植
- ◆ ImageMagick、Imlib2、GDをサポート
- ◆ 今ではmod\_small\_lightよりも高機能

# nginx\_small\_lightでできること

- ◆ 画像のリサイズ・クロップ・合成
- ◆ フォーマット変換(例：PNG -> JPEG)
- ◆ ブラー・(アン)シャープ加工
- ◆ etc

# Setup And Install

## ○ Setup

```
./setup --with-imlib2          # enable ImageMagick and Imlib2  
./setup --with-gd              # enable ImageMagick and GD  
./setup --with-imlib2 --with-gd # enable ImageMagick and Imlib2 and GD
```

## ○ Install

```
cd ${ngx_small_light_src_dir}  
./setup  
cd ${nginx_src_dir}  
./configure --with-pcre --add-module=${ngx_small_light_src_dir}  
make  
sudo make install
```

# Quick Start

```
small_light on;  
  
location ~ small_light[^/]*(.+)$ {  
    set $file $1;  
    rewrite ^ /$file;  
}  
                                # in server context
```

元画像のURI	/img/image.jpg
変換用のURI	/small_light(dw=300,dh=300)/img/image.jpg

# ディレクティブ一覧

<b>small_light</b>	nginx_small_lightのOn/Off
<b>small_light_pattern_define</b>	サムネイル生成パターン名の定義
<b>small_light_material_dir</b>	画像合成用素材の配置ディレクトリ
<b>small_light_imlib2_temp_dir</b>	テンポラリファイルディレクトリ(Imlib2用)

# ngx\_small\_lightを有効にする

```
small_light on;
```

# in main, server, location context

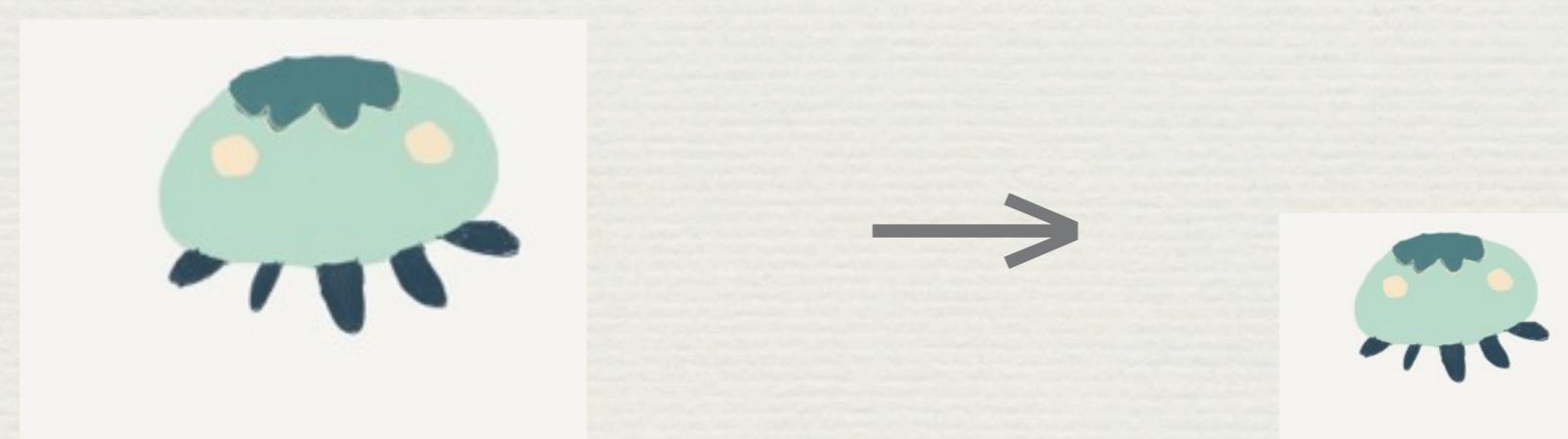
# サムネイル生成パターンの定義

```
small_light_pattern_define msize dw=500,dh=500;  
small_light_pattern_define ssize dw=120,dh=120;  
# in server context
```

- 以下のURIは同じレスポンス(サムネイル)を返す

/small\_light(p=ssize)/img/image.jpg

/small\_light(dw=120,dh=120)/img/image.jpg



# small\_light\_material\_dir

## ■設定

```
# not trailing stash (The icon.jpg is in /var/materials.)
```

```
small_light_material_dir /var/materials;
```

```
small_light_pattern_define embed dw=300,dh=300,embedicon=icon.jpg,ix=10,iy=150;
```

## ■URI

**/small\_light(p=embed)/img/image.jpg**

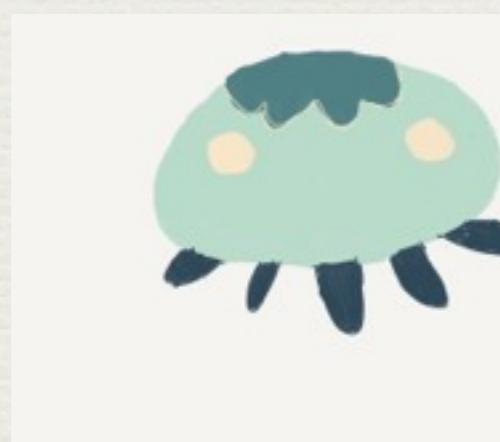


image.jpg

+



=



icon.jpg

# small\_light\_imlib2\_temp\_dir

```
# default setting
small_light_imlib2_temp_dir /tmp 1 2 0;
```

- ◆ Imlib2が生成するテンポラリファイルの保存場所
- ◆ Imlib2はファイル単位でしか画像を扱えないので  
このディレクティブが存在する
- ◆ 設定の書式はproxy\_temp\_pathと一緒に

# 基本的なパラメータ

<b>dw, dh</b>	生成するサムネイルの幅と高さ
<b>of</b>	生成するサムネイルのフォーマット(jpg,png,gif等)
<b>q</b>	画質(quality)
<b>da</b>	生成するサムネイルのアスペクト比(da=l ->長辺基準, etc)
<b>ds</b>	生成するサムネイルのスケーリング(da=s ->拡大, etc)
<b>p</b>	定義されたサムネイルパターンを代入
<b>e</b>	画像変換ライブラリを指定(imagemagick,imlib2,gd)
<b>jpeghint</b>	JPEG画像の読み込み時のダウンスケーリングによる最適化を有効にする

# Configuration Guide

[https://github.com/cubicdaiya/ngx\\_small\\_light/wiki/Configuration](https://github.com/cubicdaiya/ngx_small_light/wiki/Configuration)

オリジナルの開発者による公開資料

<http://www.slideshare.net/livedoor/smalllight2>

オリジナルの開発者による公開資料

<http://www.slideshare.net/livedoor/smalllight2>

細かい違いはあるけど大体一緒です

# mod\_small\_lightとの違い

- ofが指定されてない場合、入力・出力フォーマットが同じ
  - ◎mod\_small\_lightはofがないと常にJPEG
- pの値を上書きできない
  - ◎small\_light(p=embed,dw=300) ←embedパターン内のdwが優先される
- X-SmallLight-Descriptionヘッダを吐かない
- GDサポートの有無(mod\_small\_lightはImageMagickとImlib2のみ)
- etc...

# ngx\_small\_lightの類似モジュール

- ◆ Nginx Image Filter(*Official Module!*)
- ◆ ngx-gm-filter

# ngx\_small\_lightとの違い

- 変換ライブラリがGDまたはGraphicsMagickオンリー
- 変換に必要なパラメータをリクエストパラメータあるいはNginxの変数で受け取ってディレクティブに渡す
- モジュールの実装は簡単にしてnginx.confで頑張る感じ

# Nginx Image Filter と ngx\_small\_light

# ユースケースで比較

- ◆ /resize/img/\${type}/image.jpgでアクセス
- ◆ \${type}はmsizeとssizeの二種類
- ◆ 画像データはlocal(nginxのサーバ内)にある

# Nginx Image Filter

```
server {
    listen      9001;
    server_name localhost;
    root html;

    location ~ /resize/(msize|ssize)/(img/[^\/*\.\.(jpg)$ {
        set $type $1;
        set $file $2;
        rewrite ^ /$file;
    }

    location ~ /img/[^\/*\.\.jpg$ {
        internal;
        # default
        set $q 95;

        if ($type = msize) {
            set $width  500;
            set $height 500;
        }

        if ($type = ssize) {
            set $width  120;
            set $height 120;
        }

        image_filter_jpeg_quality $q;
        image_filter resize $width $height;
    }
}
```

# nginx\_small\_light

```
server {
    listen      9002;
    server_name localhost;

    location ~ /resize/(msize|ssize)/(img/[^\/*\.\.]+.(jpg))$ {
        set $type $1;
        set $file $2;
        proxy_pass http://127.0.0.1:9003/small_light(p=$type)/$file;
    }
}

server {
    listen      9003;
    server_name localhost;

    root html;
    small_light on;
    small_light_pattern_define msize dw=500,dh=500,q=95;
    small_light_pattern_define ssize dw=120,dh=120,q=95;

    location ~ small_light[^\/*\.\.]+(/.+)$ {
        set $file $1;
        rewrite ^ /$file last;
    }
}
```

# nginx\_small\_light(ホントはこう書きたい)

```
server {
    listen      9003;
    server_name localhost;

    root html;
    small_light on;
    small_light_pattern_define msize dw=500,dh=500,q=95;
    small_light_pattern_define ssize dw=120,dh=120,q=95;

    location ~ /resize/(msize|ssize)/(img/[^\"]*\.(jpg))$ {
        set $type $1;
        set $file $2;
        rewrite ^ /small_light(p=$type)/$file;
    }

    location ~ small_light[^\"]*/(.+)$ {
        internal;
        set $file $1;
        rewrite ^ /$file last;
    }
}
```

# Nginx Image Filter

- ◆ 変数の使い方次第でかなり柔軟な設定ができる
- ◆ 設定が複雑になりがち(変数だらけになりやすい)
- ◆ 画像変換エンジンが一種類
- ◆ 機能(パラメータ)少なめ

# nginx\_small\_light

- ♦ 比較的シンプルに書ける
- ♦ 機能(パラメータ)豊富
- ♦ 画像変換エンジンが三種類
- ♦ 画像データがローカルにあってもproxy\_pass  
しないといけないケースが('・ω・')

# 最後に

動的サムネイル生成は  
基本的に重いので前段で  
キャッシュするのがマナーです

pixivではNginxとApache Traffic Serverで  
頑張ってキャッシュしてます