ngx_small_lightで動的サムネイル生成

Tatsuhiko Kubo
cubicdaiya@gmail.com

at 第2回 闇鍋プログラミング勉強会

自己紹介

- 久保 達彦 (bokko)
- @cubicdaiya
- インフラ兼ソフトウェアエンジニア@pixiv

◎主な担当分野

○ミドルウェアの開発・メンテ

○パフォーマンスチューニング

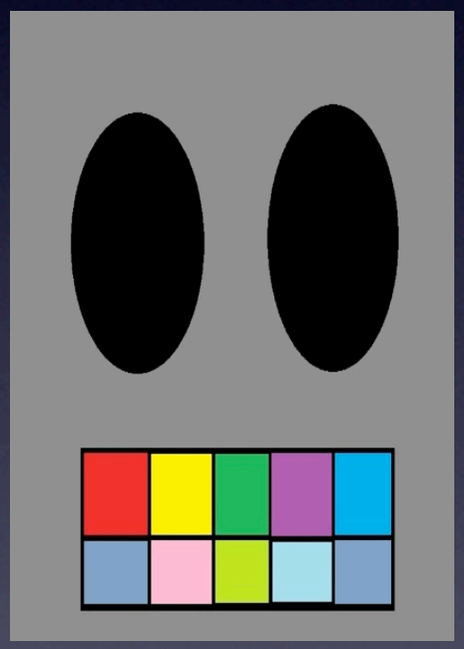
○アプリケーション開発・インフラもやるよ

周りがPHPやRuby書いてる中、黙々とC書いてます

ngx_small_light

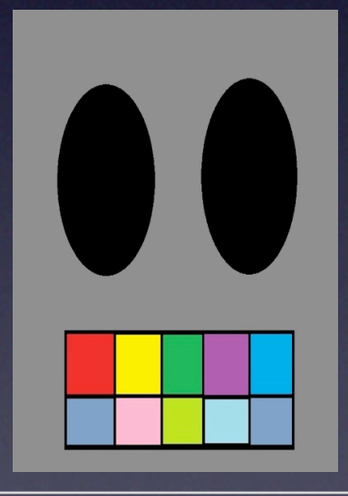
- 画像サムネイル生成サーバ with Nginx
- サムネイルの生成パターンをURLで指定
- 画像の変換はImageMagickで行う
- mod_small_lightのNginx移植版

URLのパスを変えるだけで
いろんなサムネイルを生成♪



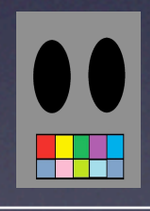
/tank.jpg

オリジナル



/small_light(p=medium)/tank.jpg

Mサイズ



/small_light(p=small)/tank.jpg

Sサイズ

ngx_small_lightでできること

- 画像のリサイズ
- 枠の付加
- アイコン埋め込み
- フォーマット変換(例:PNG -> JPEG)
- etc

Nginx

- 高速・省メモリなHTTPサーバ
- イベント駆動
- Apacheと同様、拡張モジュールが作れる
- ただし、コンパイル時に組み込む必要有り

設定ファイル

```
server {
    listen 80;
    server_name localhost;

    # small_lightを有効にする
    small_light on;
    # 変換パターンを定義
    small_light_pattern_define medium dw=500,dh=500;
    small_light_pattern_define small  dw=120,dh=120;

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

ngx_small_lightのディレクティブ

small_light	ngx_small_lightのon/off
small_light_pattern_define	生成パターンに名前を付ける
small_light_material_dir	合成素材用ディレクトリ

small_light

- ngx_small_lightのon/off(デフォルトはoff)

```
small_light on;
```

small_light_pattern_define

- サムネイル生成パターンに名前を付ける

```
small_light_pattern_define medium dw=500,dh=500;
```

以下の二つのURLは同じレスポンスを返す

```
http://localhost/small_light(dw=500,dh=500)/tank.jpg
http;//localhost/small_light(p=medium)/tank.jpg
```

dw, dh：生成するサムネイルの幅と高さ

small_light_material_dir

- 合成用素材ディレクトリ

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

- embedicon, ix, iyと組み合わせて使う

```
http://localhost/small_light(embedicon=icon.jpg,ix=0,iy=0)/tank.jpg
```

tank.jpgの座標(0, 0)に/var/materials/icon.jpgを埋め込む

主なパラメータ

dw	生成するサムネイルの幅
dh	生成するサムネイルの高さ
cw	キャンバスの幅
ch	キャンバスの高さ
cc	キャンバスの色
bw	ボーダー(枠)の幅
bh	ボーダー(枠)の高さ
q	画質(quality)
of	生成するサムネイルのフォーマット(jpg,gif,png)
jpeghint	JPEG用最適化オプション

主なパラメータ

dw	生成するサムネイルの幅
dh	生成するサムネイルの高さ
cw	キャンバスの幅
ch	キャンバスの高さ
cc	キャンバスの色
bw	ボーダー(枠)の幅
bh	ボーダー(枠)の高さ
q	画質(quality)
of	生成するサムネイルのフォーマット(jpg,gif,png)
jpeghint	JPEG用最適化オプション

その他のパラメータについては

http://p.tl/2sX9 (githubのWiki)を見てね♪

その他のプロダクト

- mod_small_light(by NHN Japan)
- mod_tofu(by クックパッド)
- Magickly
- Image Filter(Nginx標準モジュール)

その他のプロダクト

- mod_small_light(by NHN Japan)
- mod_tofu(by クックパッド)
- Magickly
- Image Filter(Nginx標準モジュール)

あれ？Nginxに同じことするモジュールある？

Image Filter

Nginx標準モジュール

さっきと同じことをする設定ファイル(ImageFilter)

```
location ~ /resize/(medium|small)/([^/]*\.jpg)$ {
    set $type $1;
    set $file $2;
    rewrite ^ /$file;
}
location ~ /[^/]*\.jpg$ {
    if ($type = medium) {
        set $w 500;
        set $h 500;
    }
    if ($type = small) {
        set $w 120;
        set $h 120;
    }
    image_filter_jpeg_quality $q;
    image_filter resize $w $h;
}
```

ngx_small_lightの場合

```
server {
    listen 80;
    server_name localhost;

    # small_lightを有効にする
    small_light on;
    # 変換パターンを定義
    small_light_pattern_define medium dw=500,dh=500;
    small_light_pattern_define small  dw=120,dh=120;

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

こっちの方がシンプルですよね♪

Image Filterのイケてないところ

- 設定が複雑化しやすい
- 変換の種類(resize,crop)や画質をディレクティブで指定しなければならない
- 名前付きパターンを実現しようとするとリクエストパラメータの解析ロジックを設定ファイル内に埋め込む必要がある
- 設定ファイルではなくプログラムを書いてる気分

ngx_small_lightの設定がシンプルな理由

- パラメータの解析をモジュール側でやってる
- 設定ファイル側は単にマッチしたパスをモジュール側に渡すだけ

その分の代償(パラメータのパーサ)

```
ngx_int_t ngx_http_small_light_parse_params(
    ngx_http_request_t *r,
    ngx_http_small_light_ctx_t *ctx,
    ngx_str_t *define_pattern,
    char *pv)
{
    char *tk, *tv, *sp1, *sp2;
    char *k, *kk, *v, *vv;
    ngx_str_t ks;
    char p[BUFSIZ];
    if (define_pattern->len > BUFSIZ - 1) {
        return NGX_ERROR;
    }
    ngx_cpystrn(p, define_pattern->data, define_pattern->len + 1);
    tk = strtok_r(p, ",", &sp1);
    while (tk != NULL) {
        tv = strtok_r(tk, "=", &sp2);
        k  = tv;
        v  = strtok_r(NULL, "=", &sp2);
        if (k == NULL || v == NULL) {
            return NGX_OK;
        }
        kk = ngx_palloc(r->pool, ngx_strlen(k) + 1);
        ngx_cpystrn(kk, k, ngx_strlen(k) + 1);
        ks.data = kk;
        ks.len  = ngx_strlen(kk);
        if (ngx_strcmp(k, "p") == 0) {
            ngx_cpystrn(pv, v, ngx_strlen(v) + 1);
        } else {
            vv = ngx_palloc(r->pool, ngx_strlen(v) + 1);
            ngx_cpystrn(vv, v, ngx_strlen(v) + 1);
            ngx_hash_add_key(&ctx->params, &ks, vv, NGX_HASH_READONLY_KEY);
        }
        tk = strtok_r(NULL, ",", &sp1);
    }

    return NGX_OK;
}
```

ところで、

サムネイル生成サーバ作って
何かうれしいことあるのか？という話

画像をアップロードできるような Webサービスの場合

- サムネイルは普通、アップロード時に生成される
- 各画像毎に何種類ものサムネイルが生成される

続・画像をアップロードできるような
Webサービスの場合

続・画像をアップロードできるような Webサービスの場合

■新しい種類のサムネイルが突然必要になることがある

続・画像をアップロードできるような Webサービスの場合

■新しい種類のサムネイルが突然必要になることがある

●全画像毎に新規にサムネイル生成するとか無理(時間がかかりすぎる)

続・画像をアップロードできるような Webサービスの場合

■新しい種類のサムネイルが突然必要になることがある

●全画像毎に新規にサムネイル生成するとか無理(時間がかかりすぎる)

■希にしか参照されないサムネイルがある

続・画像をアップロードできるような Webサービスの場合

■新しい種類のサムネイルが突然必要になることがある

●全画像毎に新規にサムネイル生成するとか無理(時間がかかりすぎる)

■希にしか参照されないサムネイルがある

●画像ストレージの容量がもったいない

続・画像をアップロードできるような Webサービスの場合

■新しい種類のサムネイルが突然必要になることがある

●全画像毎に新規にサムネイル生成するとか無理(時間がかかりすぎる)

■希にしか参照されないサムネイルがある

●画像ストレージの容量がもったいない

そこで動的生成ですよ！

ただ、サムネイルの生成処理自体はとても重いので
前段でキャッシュするのがマナーです。

最後に

- ngx_small_lightはgithubで公開中

https://github.com/cubicdaiya/ngx_small_light

- バグレポート&パッチ ウェルカム！

ご静聴ありがとございました