

Get Wallpaper

XnnYygn 2014-12-03

1. overview

1.1 keyword

wallpaper, rails, spring mvc, java, item CF, imagemagick

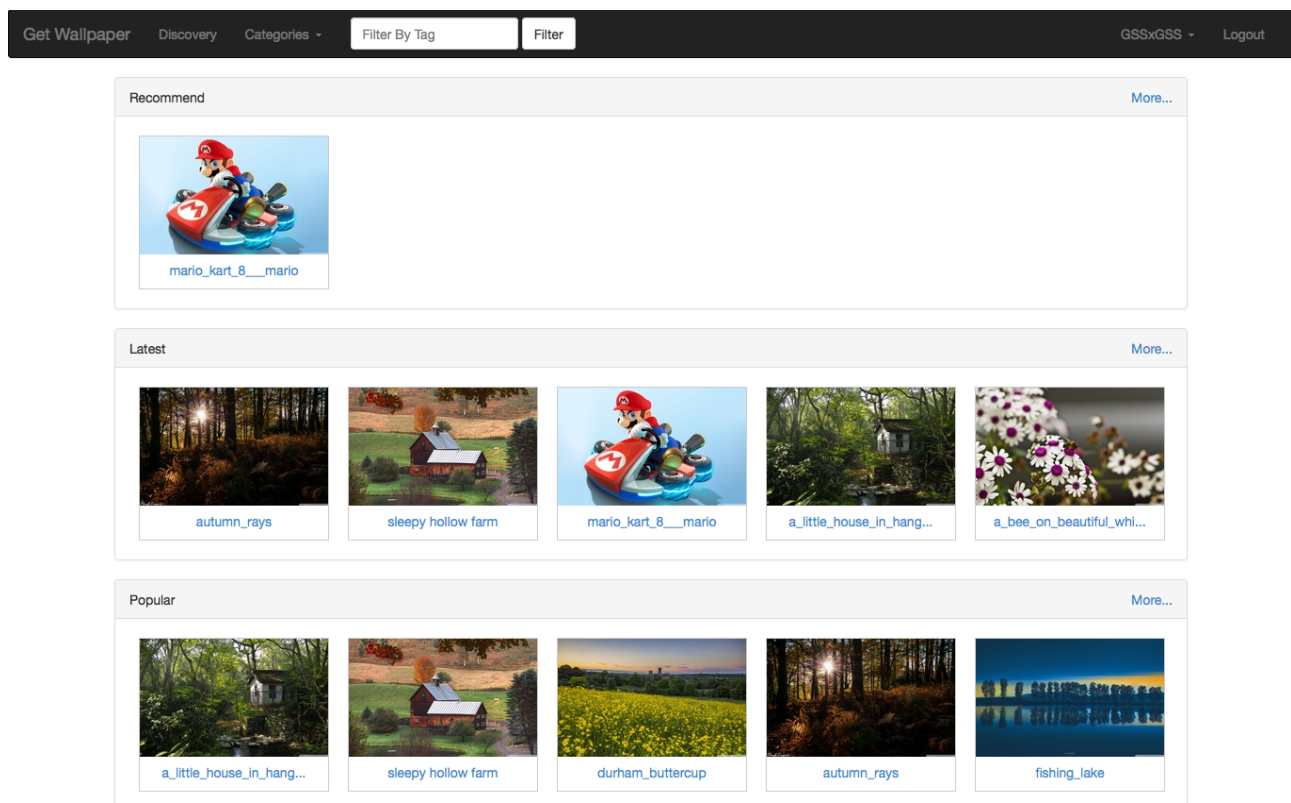
1.2 introduction

簡単にいえば、これはパソコンの壁紙画像のサイトです。普通のユーザーは最近の画像や人気ある画像やランダム画像などで選んでダウンロードできます。ログインしたユーザーは好みの画像を自分のコレクションに入れることができます。そして、システムからのおすすめ画像もあります。

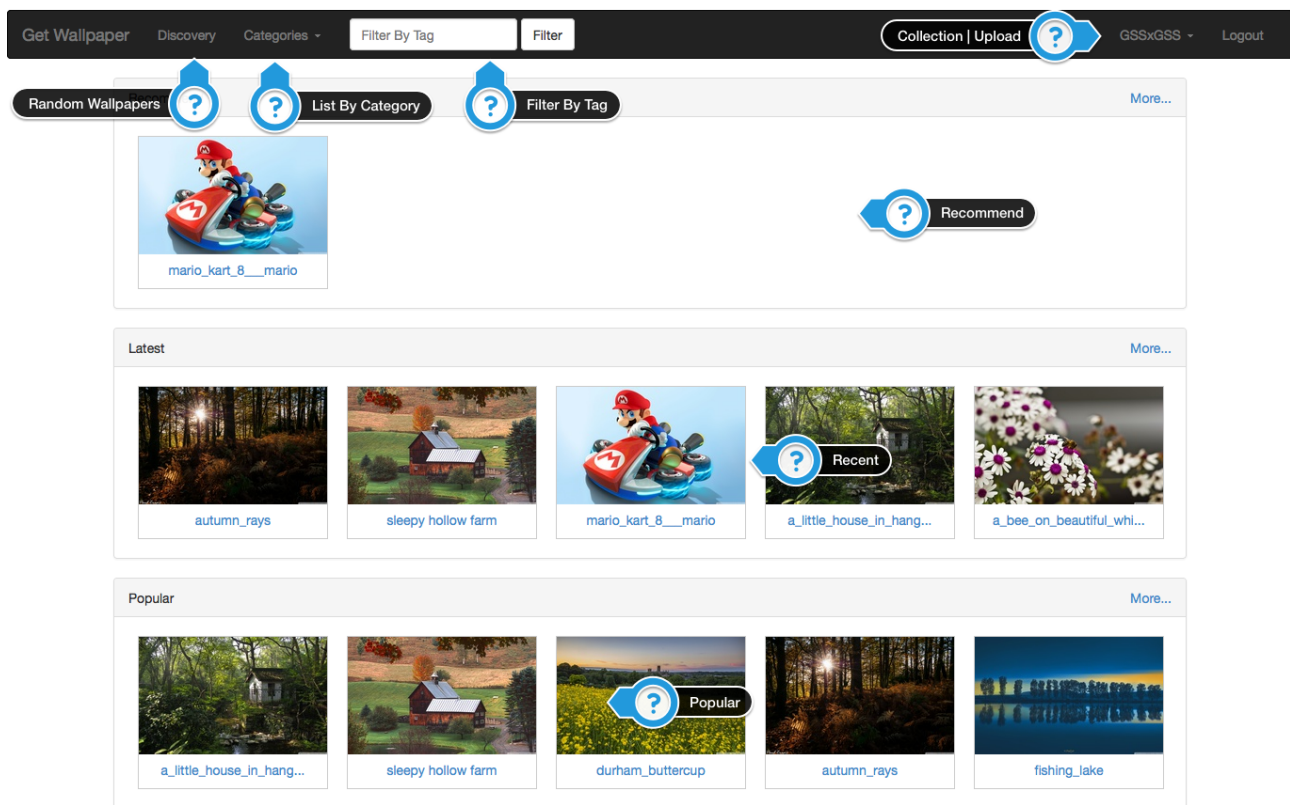
なお、ダウンロードする時、自分のパソコンに相応しいサイズの画像を選ぶことができます。

source: <https://github.com/xnnyygn/get-wallpaper>

1.3 presentation



Copyright © XnnYygn 2014 Twitter GitHub Blog



Copyright © XnnYygn 2014 Twitter GitHub Blog

```

iTerm Shell Edit View Profiles Toolbelt Window Help
1. tmux

User Load (0.1ms) SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT 1 [["id", 2]]
Wallpaper Load (0.2ms) SELECT "wallpapers".* FROM "wallpapers" WHERE "wallpapers"."id" = ? LIMIT 1 [["id", 56]]
Sent file /Users/xnnyygn/Projects/get-wallpaper/rails/public/images/wallpapers/target/4fe501e3-5a82-4ba0-a138-401638be4215-200x125.jpg (0.1ms)
Completed 200 OK in 4ms (ActiveRecord: 0.4ms)

Started GET "/wallpapers/61/thumbna?height=125&width=200" for 127.0.0.1 at 2014-12-03 07:06:44 +0800
Processing by WallpapersController#thumbnail as HTML
Parameters: {"height"=>"125", "width"=>"200", "id"=>"61"}
User Load (0.3ms) SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT 1 [["id", 2]]
Wallpaper Load (0.3ms) SELECT "wallpapers".* FROM "wallpapers" WHERE "wallpapers"."id" = ? LIMIT 1 [["id", 61]]
Sent file /Users/xnnyygn/Projects/get-wallpaper/rails/public/images/wallpapers/target/f0b00df7-1dfb-42a8-a5d1-b3c0f54f2133-200x125.jpg (0.1ms)
Completed 200 OK in 7ms (ActiveRecord: 0.6ms)

Started GET "/wallpapers/preferences" for 127.0.0.1 at 2014-12-03 07:07:40 +0800
Processing by WallpapersController#preferences as HTML
WallpaperStat Load (0.3ms) SELECT "wallpaper_stats".* FROM "wallpaper_stats"
Rendered text template (0.0ms)
Completed 200 OK in 3ms (Views: 0.4ms | ActiveRecord: 0.3ms)

INFO [main] DefaultAnnotationHandlerMapping.registerHandler(399) | Mapped URL path [/helloworld.*] onto handler [in.xnnyygn.getwallpaper.webapp.HelloWorldController@41db0505]
INFO [main] DefaultAnnotationHandlerMapping.registerHandler(399) | Mapped URL path [/helloworld/] onto handler [in.xnnyygn.getwallpaper.webapp.HelloWorldController@41db0505]
INFO [main] AutoRefreshableRemoteDataModelWrapper.refresh(59) | refresh data model
INFO [main] AutoRefreshableRemoteDataModelWrapper.refresh(83) | reload data model
INFO [main] DefaultAnnotationHandlerMapping.registerHandler(399) | Mapped URL path [/recommend-wallpaper] onto handler [in.xnnyygn.getwallpaper.webapp.ItemCfController@3e4854d8]
INFO [main] DefaultAnnotationHandlerMapping.registerHandler(399) | Mapped URL path [/recommend-wallpaper.*] onto handler [in.xnnyygn.getwallpaper.webapp.ItemCfController@3e4854d8]
INFO [main] DefaultAnnotationHandlerMapping.registerHandler(399) | Mapped URL path [/recommend-wallpaper/] onto handler [in.xnnyygn.getwallpaper.webapp.ItemCfController@3e4854d8]
INFO [main] DispatcherServlet.initServletBean(321) | FrameworkServlet 'dispatcher': initialization completed in 695 ms
2014-12-03 07:05:40.557:WARN:oejs:RequestLogHandler:RequestLog
2014-12-03 07:05:40.580:INFO:oejs:AbstractConnector:Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
INFO [pool-1-thread-1] AutoRefreshableRemoteDataModelWrapper.refresh(59) | refresh data model
INFO [pool-1-thread-1] AutoRefreshableRemoteDataModelWrapper.refresh(83) | reload data model
INFO [pool-1-thread-1] AutoRefreshableRemoteDataModelWrapper.refresh(59) | refresh data model
INFO [pool-1-thread-1] AutoRefreshableRemoteDataModelWrapper.refresh(83) | reload data model

xnnyygn 0:goagent 1:gn-server* 2:bash- 2014-12-03 07:07

```

サーバー側
上のはRails、下のはおすすめセンター（Java）

Upload Wallpaper

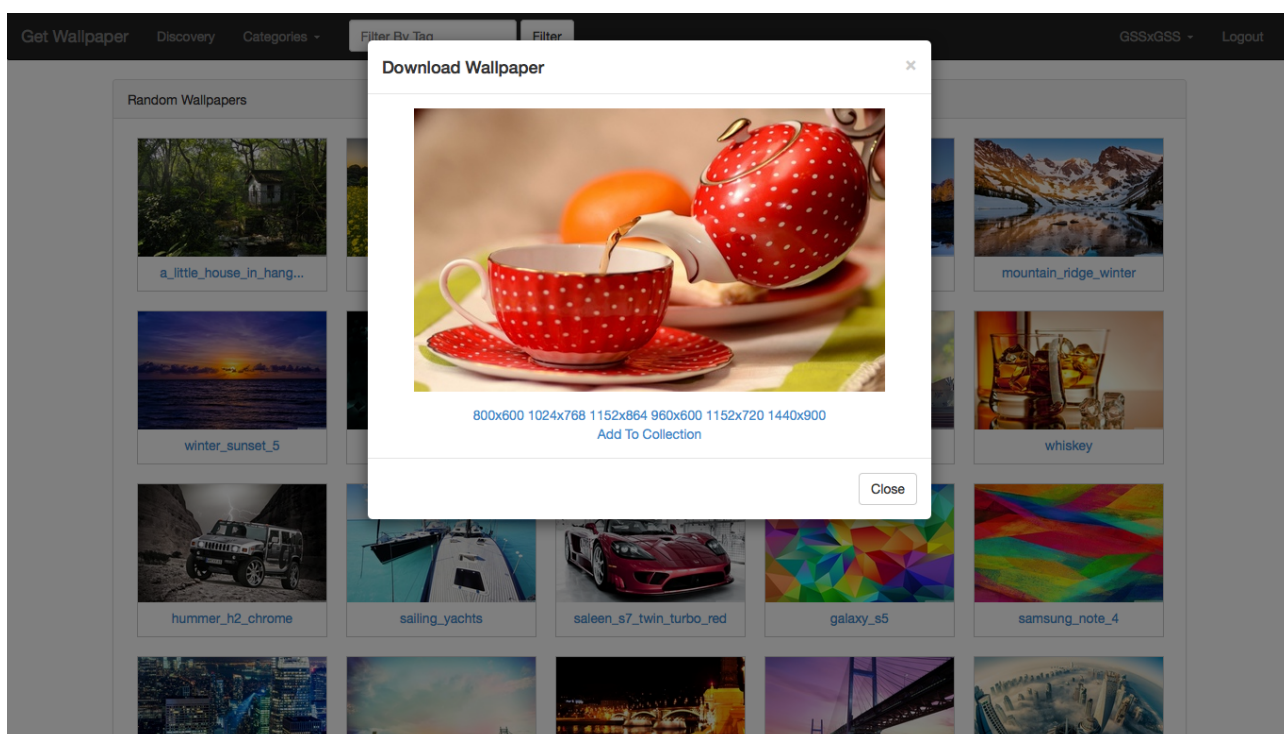
Title

Category

Wallpaper No file selected.

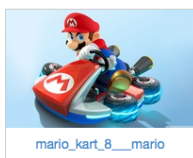
Tags

Copyright © XnnYygn 2014 [Twitter](#) [GitHub](#) [Blog](#)



ご覧通り、サイズを選べます。

Recommend Wallpapers



mario_kart_8__mario

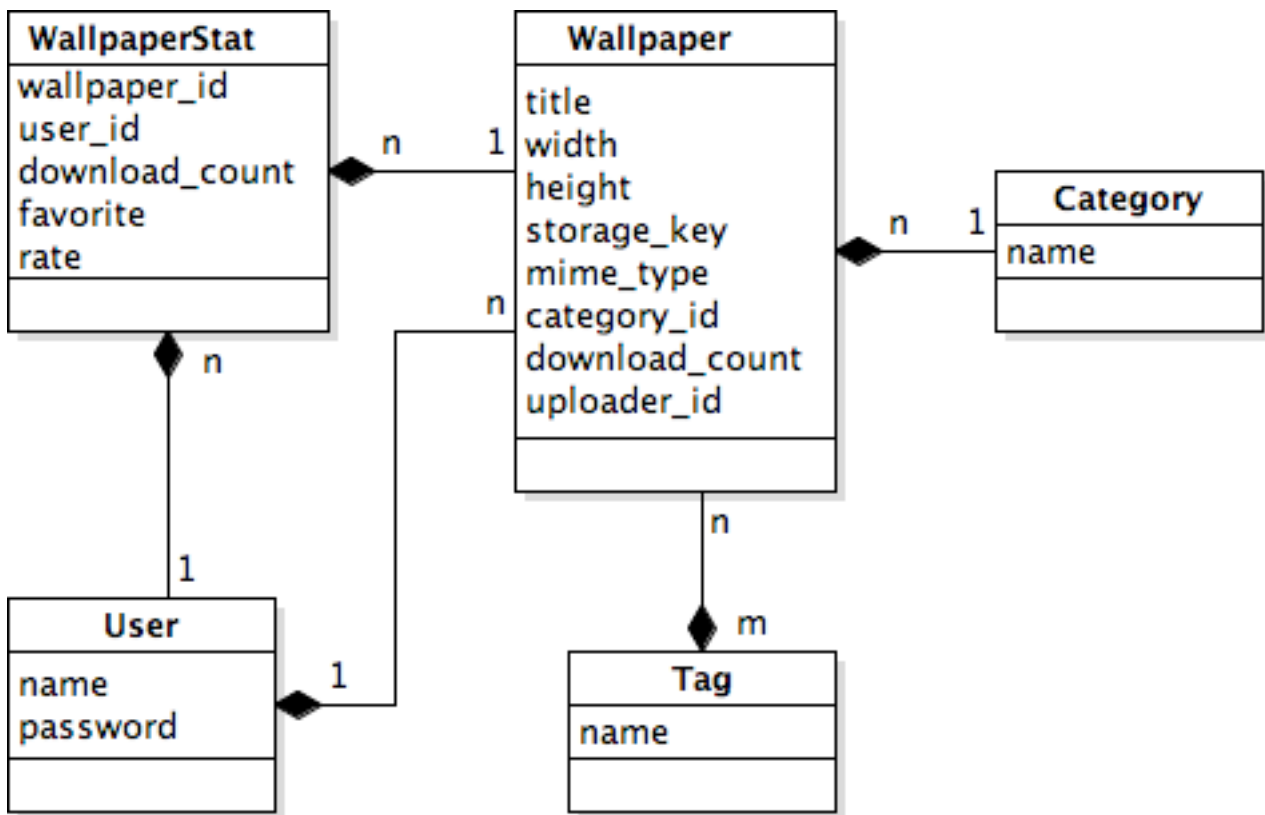
Copyright © XnnYygn 2014 [Twitter](#) [GitHub](#) [Blog](#)

お勧め画像

確かに、これは複雑なページに見えませんが、でも、直接SQLなどでこんな結果になりません。せめて「似ている画像」程度が必要でしょう。もつと多いデータがあるほど、ユーザーにもつと意味のある画像がおくれるでしょう。

2. design

2.1 models



2.2 image storage

Path	Description
ROOT	
ROOT/original	original wallpaper
ROOT/target	resized or temporary images, e.g thumbnail

targetの画像は一定時間にアクセスがない場合、削除されるかもしれません。これはRailsのサイトの機能ではない、UNIX/LINUXのfindなどコマンドを利用して定時のスクリプトで処理されるんです。

2.3 image thumbnail and download

`_wallpaper_thumbnail_item.html.erb`

```
<%= image_tag thumbnail_wallpaper_path(wallpaper, width: 200, height: 125), alt: 'Image' %>
```

download_dialog.html.erb

```
<%= link_to "#{r[0]}x#{r[1]}", download_wallpaper_path(@wallpaper, width: r[0], height: r[1]), target: '_blank' %>
```

画像はアップロードされる時、いろんなサイトの画像は生成されません。
だから、小さな画像を表示されるとダウンロードする時、その場で生成される必要があります。
まず、「target」で指定されるサイズの画像を探して、ある場合は直接戻り、ない場合は「imagemagick」のコマンド

```
convert #{original_file_path} -crop #{crop_bound[0]}x#{crop_bound[1]}+#{crop_bound[2]}+#{crop_bound[3]} -resize #{target_width}x#{target_height} #{target_file_path}
```



ここの「crop_bound」は計算式で得られる。
「sr」は「source_ratio」、source_width / source_height
「tr」は「target_ratio」、target_width / target_height

```
def calculate_crop_bound(source_width, source_height, source_ratio, target_ratio)
  if source_ratio >= target_ratio
    crop_width = source_height * target_ratio
    [
      crop_width.to_i,
      source_height,
      ((source_width - crop_width) / 2).to_i,
      0
    ]
  else
    crop_height = source_width / target_ratio
    [
      source_width,
      crop_height.to_i,
      0,
      ((source_height - crop_height) / 2).to_i
    ]
  end
end
```

「imagemagick」のコマンドはまず「crop」、一番大きい利用できるエリアを計算し、そして、このエリアを「resize」、望むサイズに変更します。

コマンド例

```
convert /Users/xnnyygn/Projects/get-wallpaper/rails/public/images/wallpapers/original/9ad1ca53-a856-499a-9922-f30731c3df89 -crop 1440x864+0+18 -resize 500x300 /Users/xnnyygn/Projects/get-wallpaper/rails/public/images/wallpapers/target/9ad1ca53-a856-499a-9922-f30731c3df89-500x300.jpg
```

```
source_width 1440
source_height 900
target_width 500
target_height 300
```

生成したら、「send_file」

```
send_file wallpaper.crop_and_resize(width, height), type: wallpaper.mime_type, disposition: 'inline'
```

```
send_file wallpaper.crop_and_resize(width, height),
  filename: "#{wallpaper.title}-#{width}x#{height}#{wallpaper.determine_extension}",
  type: wallpaper.mime_type,
  disposition: 'attachment'
```

上のほうが「thumbnail」、下のほうが「download」、主な区別のところは「disposition」

2.4 wallpaper upload

```
def check_and_save_wallpaper(wallpaper, wallpaper_tmpfile)
  # check if empty
  if wallpaper_tmpfile.nil?
    @wallpaper.errors[:base] = 'image required'
    return false
  end

  # check size 1M threshold
  logger.debug "wallpaper size #{wallpaper_tmpfile.size}"
  if wallpaper_tmpfile.size > 1024 * 1024
    @wallpaper.errors[:base] = 'image too large, size threshold 1M'
    return false
  end

  # retrieve resolution
  identify_result = `identify -format '%w %h' #{wallpaper_tmpfile.path}`
  if $?.to_i != 0
    @wallpaper.errors[:base] = 'failed to retrieve resolution from image'
    return false
  end

  resolution = identify_result.split
  width = resolution[0].to_i
  height = resolution[1].to_i
  logger.debug "wallpaper resolution #{width}x#{height}"
  storage_key = SecureRandom.uuid()
end
```

```

# check resolution
unless Wallpaper.check_minimum_resolution(width, height)
  @wallpaper.errors[:base] = 'the minimum resolution of wallpaper is 800x600'
  return false
end

dest_path = Wallpaper.determine_storage_path(storage_key)
logger.info "upload wallpaper to #{dest_path}"
File.open(dest_path, 'wb') {
  |df| df.write(wallpaper_tmpfile.read)
}

@wallpaper.storage_key = storage_key
@wallpaper.width = width
@wallpaper.height = height
@wallpaper.mime_type = wallpaper_tmpfile.content_type
return true
end

```

これは中心のコードです。チェックは

- image != nil
- image size < 1M
- get image resolution successfully
- image resolution >= 800x600

「imagemagick」のコマンド

```
identify -format '%w %h' #{wallpaper_tmpfile.path}
```

「storage_key」は「UUID」で、画像の「id」は保存しないとわからないので利用できません。

2.5 random

普通のSQLなら

```
SELECT * FROM wallpapers ORDER BY RANDOM() LIMIT 10
```

でも、データベースによってSQLが違うんですので、実際使ったのはこのSQLです。

```

SELECT id FROM wallpapers ORDER BY created_at DESC LIMIT 100
Array#sample(20)
SELECT * FROM wallpapers WHERE id in [xxx]

```

ここのsampleはrubyのコードなので、一つのSQLではない。

```

@wallpapers = Wallpaper.find(
  Wallpaper.order(created_at: :desc).limit(100).pluck(:id).sample(20)
)

```

2.6 popular

仕組みは簡単、直接ダウンロード数でランキングです。

2.7 recommend

2.7.1 ITEM CF

これは一番難しい部分でしょう。
まずCF。

https://en.wikipedia.org/wiki/Collaborative_filtering

一般的に言えば、CFはUSER CFとITEM CF 2つがあります。USER CFの仕組みは似ている人を探して、他の人のものをあなたにすすめることです。ITEM CFの仕組みは似ているものを探して、あなたにすすめることです。

USER CFとITEM CFは別々の特徴があるので、普通は一緒に使ったほうがいい。ここは簡単のため、ITEM CFを選びました。あなた以前好きなものに基づいて計算されました。実はこのサイトに「random」や「popular」など部分があるので、USER CFがなくても大丈夫でしょう。

直接rubyで計算できますか。多分出来るんでしょう。でも、今回mahout (Java) に任せました。アルゴリズムが分かりますが、実際の問題は多いでしょう。mahoutは一つの機械学習フレームワークです。

<https://mahout.apache.org/>

- User and Item based recommenders
- Matrix factorization based recommenders
- K-Means, Fuzzy K-Means clustering
- Latent Dirichlet Allocation
- Singular Value Decomposition
- Logistic regression classifier
- (Complementary) Naive Bayes classifier
- Random forest classifier
- High performance java collections
- A vibrant community

ITEM CF コード

```
DataModel dataModel = new FileDataModel(new File(filePath));
ItemSimilarity similarity = new PearsonCorrelationSimilarity(dataModel);
Recommender recommender = new GenericItemBasedRecommender(dataModel, similarity);
recommender.recommend(user_id = 123, max = 10);
```

先言ってました「似ているものを探す」に関するアルゴリズムに「距離」の計算が必要です。ここは「PearsonCorrelationSimilarity」です。他には「EuclideanDistanceSimilarity」もあります。

ここまでは原理です。実に応用するのは別のことを考える必要があります。

2.7.2 rails and Java Web

HTTPで通信。Java WebのフレームワークはSpring MVC、簡単のためデータベースなし。Railsの方はfaradayで、JavaのほうがApache commons-http-clientです。

Rails側のコード

```

def recommend(max)
  response = Faraday.get "http://localhost:8080/recommend-wallpaper.htm?user_id=#{@current_user}"
  if response.status == 200
    json = JSON.parse(response.body)
    if json['success']
      return Wallpaper.find(json['item_ids'])
    end
  end
  []
end

```

2.7.3 refreshable data model

ユーザーのデータはいつも変わっていますので、お勧めセンターのデータはリロードの必要があります。ここで定時作業を設定します。1分くらいでデータリロード。リロードコード（Java側）

まずRailsからデータをゲット、ファイルに書き込む、FileDataModelをリロード。

```

public void refresh() throws DataModelRefreshException {
    logger.info("refresh data model");

    // refresh file
    GetMethod method = new GetMethod(sourceUrl);
    boolean dataUpdated = false;
    try {
        client.executeMethod(method);
        int statusCode = method.getStatusCode();
        if (statusCode == HttpServletResponse.SC_OK) {
            IOUtils.write(method.getResponseBodyAsString(), new FileOutputStream(
                filePath));
            dataUpdated = true;
        } else {
            logger
                .warn("failed to retrieve latest data file, server reply error,
                    + statusCode + "]");
        }
    } catch (IOException e) {
        logger.warn("failed to retrieve latest data file, io error", e);
    } finally {
        method.releaseConnection();
    }

    if (!dataUpdated) return;
    logger.info("reload data model");

    // refresh data model
    if (dataModel == null) {
        try {
            dataModel = new FileDataModel(new File(filePath));
        } catch (IOException e) {
            throw new DataModelRefreshException("failed to load data model", e);
        }
    } else {
        dataModel.refresh(null);
    }
}

```

2.7.4 preferences data

テストのデータ

2,65,5,1417522483
2,23,3,1417442145
2,25,3,1417440697
1,23,3,1417514439
1,65,3,1417514442
3,23,5,1417519760
3,27,3,1417519018
3,61,3,1417519021
3,62,2,1417519028
3,65,5,1417521646
3,36,5,1417521640
3,49,3,1417521614
3,50,3,1417521617
1,36,3,1417521664
1,25,5,1417521667

説明させていただきます。第一ラインを見せてください。
2はユーザーID。65は画像ID、5はスコア、1417522483は時間。
Rails側のコード

```
def preferences
  data = ''
  WallpaperStat.all.each do |ws|
    score = 0
    score += 2 if ws.download_count > 0
    score += 3 if ws.favorite
    if score == 0
      next
    end

    data << ws.user_id.to_s << ','
    data << ws.wallpaper_id.to_s << ','
    data << score.to_s << ','
    data << ws.updated_at.to_time.to_i.to_s << "\n"
  end
  render text: data
end
```

2.7.5 invocation chain

普通の時

Rails Request

http://localhost:8080/recommend-wallpaper.htm?user_id=xxx&max=10

Java Web Response

{“success”: true, “item_ids”: [1, 2, 3]}

リロードの時

Java Web Request

<http://localhost:3000/wallpapers/preferences>

Rails Response

2,65,5,1417522483
2,23,3,1417442145
...

2.8 Filter By Tag

これは未完成の機能です。今は一つのタグしか探しません。もし時間があれば、お勧めセンターでluceneとかsolrとかサーチエンジンを作ってみます。NRTなど問題は大丈夫でしょう。

3 How to Run

まずRails

```
$ cd rails && rails s
```

次はrecommend-center、Javaとmavenを確認したら

```
$ cd recommend-centre && mvn jetty:run
```

おすすめデータは /tmp/p.dat にあります。