

## jpmobile + Rails 2.3.4 で作る携帯サイト入門 【前編】

更新日時:2009/12/07 00:21:31

- [概要](#)
- [jpmobile とは](#)
- [Rails プロジェクトのセットアップ](#)
  - [Debian パッケージのインストール](#)
  - [gem のアップデート](#)
  - [Rails のインストール](#)
  - [Rails プロジェクトの作成](#)
- [jpmobile のインストール](#)
- [初期設定](#)
  - [セッションの設定](#)
- [動かしてみよう](#)
  - [コントローラの作成と設定](#)
- [カスタマイズしよう](#)
  - [ビューファイルの追加](#)
  - [確認するには](#)
- [絵文字を使ってみる](#)
  - [jpmobile での絵文字の埋め込み](#)
- [まとめ](#)
- [次回予告](#)
- [リンク](#)
- [著者について](#)
  - [Rust/OGAWA](#)
- [jpmobile + Rails 2.3.4 で作る携帯サイト入門 連載一覧](#)

### 概要

jpmobile + Ruby on Rails 2.3.4 で携帯サイトを作る際には、これさえ押さえておけば大丈夫という、ベストプラクティスを紹介します。意外にはまるところや、基本的な設定方法、Ruby on Rails 2.3.4 での変更による影響点などを紹介します。

「これで誰でもお手軽に携帯電話対応サイトを構築できる！」

### jpmobile とは

[Ruby 札幌](#) 所属の [darashi](#) さんが制作されている、携帯電話特有の機能を Ruby on Rails で利用するためのプラグインです。

ここで私は主に Rails 2.3 以降の対応を担当しています。

## Rails プロジェクトのセットアップ

ここでは Debian 5.0.3 がインストールされたシステムをターゲットとします。またデータベースには sqlite3 を用います。

### Debian パッケージのインストール

まず Ruby と関連パッケージをインストールします。

```
$ sudo aptitude install ruby ruby1.8 ruby1.8-dev sqlite3 libsqlite3-dev rubygems git-core libopenssl-ruby
```

### gem のアップデート

Debian 5.0.3 のパッケージは gem 1.2.0 なのですが、Rails 2.3.4 は gem 1.3.2 以降が必要となるので下記の手順でアップデートします。

gem 1.2.0 で rubygems-update がインストールされていない場合、gem 最新版への直接の update はうまく行えません。そのため、まず 1.3.1 へ update を行ない、その後最新版へ update します。

```
$ sudo gem install rubygems-update -v 1.3.1
$ sudo /var/lib/gems/1.8/bin/update_rubygems
$ sudo gem install rubygems-update
$ sudo /var/lib/gems/1.8/bin/update_rubygems
```

### Rails のインストール

Rails と必要な gem パッケージをインストールします。

```
$ sudo gem install rails sqlite3-ruby
```

### Rails プロジェクトの作成

プロジェクト名は「jpmobile-rails」とします。

```
$ mkdir ~/rails-projects/
$ cd ~/rails-projects/
$ rails jpmobile-rails
  create
  create  app/controllers
  create  app/helpers
  create  app/models
  create  app/views/layouts
  create  config/environments
  create  config/initializers
  create  config/locales
  create  db
  create  doc
  create  lib
  create  lib/tasks
  create  log
  create  public/images
  create  public/javascripts
  create  public/stylesheets
  create  script/performance
  create  test/fixtures
  create  test/functional
  create  test/integration
  create  test/performance
```

```
create test/unit
create vendor
create vendor/plugins
create tmp/sessions
create tmp/sockets
create tmp/cache
create tmp/pids
create Rakefile
create README
create app/controllers/application_controller.rb
create app/helpers/application_helper.rb
create config/database.yml
create config/routes.rb
create config/locales/en.yml
create db/seeds.rb
create config/initializers/backtrace_silencers.rb
create config/initializers/inflections.rb
create config/initializers/mime_types.rb
create config/initializers/new_rails_defaults.rb
create config/initializers/session_store.rb
create config/environment.rb
create config/boot.rb
create config/environments/production.rb
create config/environments/development.rb
create config/environments/test.rb
create script/about
create script/console
create script/dbconsole
create script/destroy
create script/generate
create script/runner
create script/server
create script/plugin
create script/performance/benchmark
create script/performance/profiler
create test/test_helper.rb
create test/performance/browsing_test.rb
create public/404.html
create public/422.html
create public/500.html
create public/index.html
create public/favicon.ico
create public/robots.txt
create public/images/rails.png
create public/javascripts/prototype.js
create public/javascripts/effects.js
create public/javascripts/dragdrop.js
create public/javascripts/controls.js
create public/javascripts/application.js
create doc/README_FOR_APP
create log/server.log
create log/production.log
create log/development.log
create log/test.log
$ cd jpmobile-rails
```

## jpmobile のインストール

次に [github](#) から、jpmobile をプラグインとしてインストールします。

```
$ git clone git://github.com/darashi/jpmobile.git vendor/plugins/jpmobile
$ rm -rf vendor/plugins/jpmobile/.git
```

## 初期設定

### セッションの設定

cookie が使えない携帯電話でセッション管理するための設定です。以前は config/environment.rb に書きましたが、Rails 2.3 からは config/initializers/session\_store.rb というセッションまわり用

の初期設定ファイルができていますので、こちらに書くことにします。

```
$ emacs config/initializers/session_store.rb
```

変更内容はおおまかに言って下記の 3 点です。

- :key を書き換える
  - ここが長いと URL が長くなってしまい、携帯端末によっては欠落する可能性があるので、少し短くしておきます。

```
ActionController::Base.session = {  
  :key      => '_session_id',  
  :secret   => 'hoge'hoge'  
}
```

- ActionController::Base.session\_store = :active\_record\_store を有効にする。
  - Cookie が使えない携帯端末でセッション管理をするために session\_store に :active\_record\_store を設定する必要があります。

```
ActionController::Base.session_store = :active_record_store
```

- Cookie よりもセッションパラメータを先に見る設定をする
  - Rails 2.3 以降では、Cookie が使える場合にはそちらが優先されてしまいます。このため後述する trans\_sid で :always (PC でも有効にする設定) が有効になりません。設定方法は下記のように :cookie\_only => false を追加します。
    - 実はこの設定だけでは有効になりません。jpmobile では ActionController に手を入れることで、先にセッションパラメータを見るように変更しています。

```
ActionController::Base.session = {  
  :key      => '_session_id',  
  :cookie_only => false,  
  :secret   => 'hoge'hoge'  
}
```

最終的には以下ようになります。:key と :secret は適宜置き換えてください。

```
# Be sure to restart your server when you modify this file.  
  
# Your secret key for verifying cookie session data integrity.  
# If you change this key, all old sessions will become invalid!  
# Make sure the secret is at least 30 characters and all random,  
# no regular words or you'll be exposed to dictionary attacks.  
ActionController::Base.session = {  
  :key      => '_session_id',  
  :cookie_only => false,  
  :secret   => 'hoge'hoge'  
}  
  
# Use the database for sessions instead of the cookie-based default,  
# which shouldn't be used to store highly confidential information  
# (create the session table with "rake db:sessions:create")  
ActionController::Base.session_store = :active_record_store
```

- セッションテーブルを作成する
  - 準備ができたなら実際にセッションを管理するテーブルを作成します。

```
rake db:sessions:create  
rake db:migrate
```

## 動かしてみよう

では実際に動かしてみましょう。

## コントローラの作成と設定

まずはコントローラを作成します。ここでは TopController を作ってみましょう。

```
$ ruby script/generate controller Top
exists app/controllers/
exists app/helpers/
create app/views/top
exists test/functional/
create test/unit/helpers/
create app/controllers/top_controller.rb
create test/functional/top_controller_test.rb
create app/helpers/top_helper.rb
create test/unit/helpers/top_helper_test.rb
```

次にオプションを設定します。ここではセッションパラメータが有効に働くか確認するために trans\_sid を設定します。また簡単なビューも作成してみましょう。

- app/controllers/top\_controller.rb

```
class TopController < ApplicationController
  trans_sid :always

  def index
    session[:count] ||= 0
    session[:count] += 1
    @count = session[:count]
  end
end
```

- app/views/top/index.html.erb

```
<%= @count -%><br />
<br />
<%= link_to "Go to index", :action => "index" -%>
```

ではブラウザで確認してみましょう。

- サーバの起動

```
$ ruby script/server
```

サーバを起動してブラウザで <http://localhost:3000/top/> にアクセスすると、下記のような画面になります。

- 初回アクセス

1

[Go to index](#)

ではリンクをクリックして見ましょう。数字が 1 つ増えて 2 になっていて、リンクの URL に指定したセッションパラメータがついていることがわかります。

- 2 回目のアクセス

2

[Go to index](#)

“http://10.211.55.3:3000/top?\_session\_id=2ce61e7ab75fc6cb949a7ebc164eb0da” を新規タブで開く

trans\_sid :always を指定することでセッション ID を URL のクエリーパラメータから取得するようになりました。Cookie が使えない携帯端末では、この機能を使うことでセッション管理が可能となります。

## カスタマイズしよう

次に PC と携帯端末とでビューを切り替えてみましょう。同じコントローラとアクションを使ってみます。

### ビューファイルの追加

jpmobile では、端末のユーザエージェントに応じてビューファイルを切り替えて表示することができます。ビューファイルと端末の対応関係は以下の通りです。

- アクション index に対するビューの選択リスト

ファイル名	対応端末
index_mobile_docomo.html.erb	NTT ドコモ携帯全般
index_mobile_au.html.erb	au 携帯全般
index_mobile_softbank.html.erb	SoftBank 携帯全般
index_mobile_willcom.html.erb	WILLCOM 携帯全般
index_mobile_emobile.html.erb	イー・モバイル携帯全般
index_mobile.html.erb	携帯全般
index.html.erb	上記以外の全て

- 優先順位は上記の順で、ファイルが存在しなければ飛ばされます。
  - たとえば index\_mobile.html.erb だけがある場合は下記ようになります。
    - 携帯であれば index\_mobile.html.erb が表示される。
    - それ以外は index.html.erb が表示される

ここでは携帯端末と PC とでビューを切り替えてみましょう。index\_mobile.html.erb を追加して script/server を再起動します。（新規ファイルの追加なので再起動が必要です。）

- app/views/top/index\_mobile.html.erb

```
mobile !!<br />
<%= @count -%><br />
<br />
```

```
<%= link_to "Go to index", :action => "index" -%>
```

## 確認するには

さて確認しようと思っても、PC のブラウザで閲覧する限りはずっと index.html.erb が表示されてしまいます。jpmobile はユーザエージェントで判別しているので、これを偽装する必要があります。主な偽装の方法は下記の通りです。

- Firefox のアドオン [FireMobileSimulator](#)
- [ssb \(server side browser\)](#)

今回は前者の FireMobileSimulator を使って確認します。Firefox で [ツール]->[FireMobileSimulator]->[DC P903i] を選択して <http://localhost:3000/top/> にアクセスしてみてください。"mobile !!" と追加されているのがわかります。

- mobile !!と表示されている


```
mobile !!  
1  
Go to index
```

このように jpmobile では携帯キャリアごとにビューを切り替えることで、ビューの中に if else end などの条件分岐を少なく済ませることができます。

## 絵文字を使ってみる

携帯といえば絵文字です。次は絵文字を表示してみましょう。

### jpmobile での絵文字の埋め込み

ビューファイルへの絵文字の埋め込みは HTML の実体参照を利用しています。たとえば [NTT](#)  [ドコモの絵文字](#) を埋め込む場合を考えます。表の中でサッカーの Unicode のコードは E656 となっています。これを先ほどの携帯用ビュー index\_mobile.html.erb に埋め込んでみましょう。\*1

- app/views/top/index\_mobile.html.erb

```
&#xe656;<br />  
mobile !!<br />  
<%= @count -%><br />  
<br />  
<%= link_to "Go to index", :action => "index" -%>
```

先ほどと同じく FireMobileSimulator で確認します。FireMobileSimulator は絵文字を適切に表示してくれるので、絵文字の確認にも非常に有用です。

- NTT ドコモで絵文字を表示したところ

```
 mobile !!  
1  
Go to index
```

さて一見よさそうに見えるのですが、ブラウザで文字コード (Firefox の場合は [表示]->[文字エンコーディング]) を確認すると、文字コードが UTF-8 になっています。携帯電話ではそれぞれのキャリアに応じた文字コードを用いる必要があります。また絵文字の埋め込みに関しても、NTT ドコモのコードで埋め込んだものはそのままでは他のキャリアでは見えません。

そこで出力変換用のフィルターを適用します。適用したいコントローラに `mobile_filter` と追加します。ここでは Top コントローラに追加してみましょう。

- `app/controllers/top_controller.rb`

```
class TopController < ApplicationController
  trans_sid :always

  # add filter
  mobile_filter

  def index
    session[:count] ||= 0
    session[:count] += 1
    @count = session[:count]
  end
end
```

さてもう一度アクセスして文字コードを確認すると、今度は Shift\_JIS に変わっているのが確認できます。では続いて他のキャリアで見てみましょう。[ツール]->[FireMobileSimulator]->[AU W53CA] と選んで au で見てみることにします。

- au で見たとき



mobile !!  
2  
[Go to index</>](#)

若干分かりにくいですが、絵文字が変わっています。また文字コードも Shift\_JIS に変わっているのがわかります。次に [ツール]->[FireMobileSimulator]->[SB SoftBank 930SH (3GC 型)] と選んで SoftBank で見てみます。

- SoftBank で見たとき



mobile !!  
3  
[Go to index](#)

今度は絵文字の変化も少しわかりやすいですね。また文字コードも UTF-8 に変わっているのがわかると思います。

## まとめ

今回は jpmobile の主な機能のうち

- セッション ID の URL パラメータへの追加
- ビューの切り替え
- 文字コード変換



- 絵文字変換

の 4 つを紹介しました。この他にも

- 端末情報の取得
- 位置情報取得
- 全角半角変換

など携帯サイト制作には欠かせない機能が含まれています。不明点や問題点がありましたら、下記のメーリングリストか IRC でお気軽にお尋ねください。

## 次回予告

今回は jpmobile を拡張して iPhone や Android といった新しい端末を判別できるようにしてみる予定です。

## リンク

RDoc Documentation

<http://jpmobile.rubyforge.org/rdoc>

GitHub

<http://github.com/darashi/jpmobile/>

RubyForge Project Page

<http://rubyforge.org/projects/jpmobile>

Mailing List

<http://groups.google.com/group/jpmobile>

IRC Channel

#jpmobile@freenode.net

## 著者について

Rust/OGAWA

300 万人規模の携帯向けメーリングリストサービスを Ruby on Rails で構築・運用している人。  
jpmobile / termtter のコミッターでもある。

[Tokyu.rb](http://Tokyu.rb) 所属

## jpmobile + Rails 2.3.4 で作る携帯サイト入門 連載一覧

- jpmobile + Rails 2.3.4 で作る携帯サイト入門 【前編】

---

\*1 サッカーの絵文字は [NTT ドコモのホームページ](#)「基本絵文字」より引用。

Powered by [Ruby](#) 1.8.7 (2008-08-11).

Founded by RubiMa Editors.