プラグイン開発ガイド

Railsを知らない人のための Redmine
プラグイン開発入門

吉田光良

目的

Redmine のプラグイン開発方法の概略を説明。
わりと簡単に作れそうだと思ってもらえるようにしたい。

作り方の詳細は r-labs の プラグイン開発ガイド。

ただし、プラグインを作る際に、一番参考になるのは他のプラグインや Redmine のコードを参考にすること。
コードを読む上で必要な情報もなるべく説明に追加。

目次

私が作成したプラグイン

TestLink Link プラグイン →
動機：Redmineから TestLink(テスト管理システム) へのリンクを張る方法がなかったので。
参考：Wiki UNC プラグイン

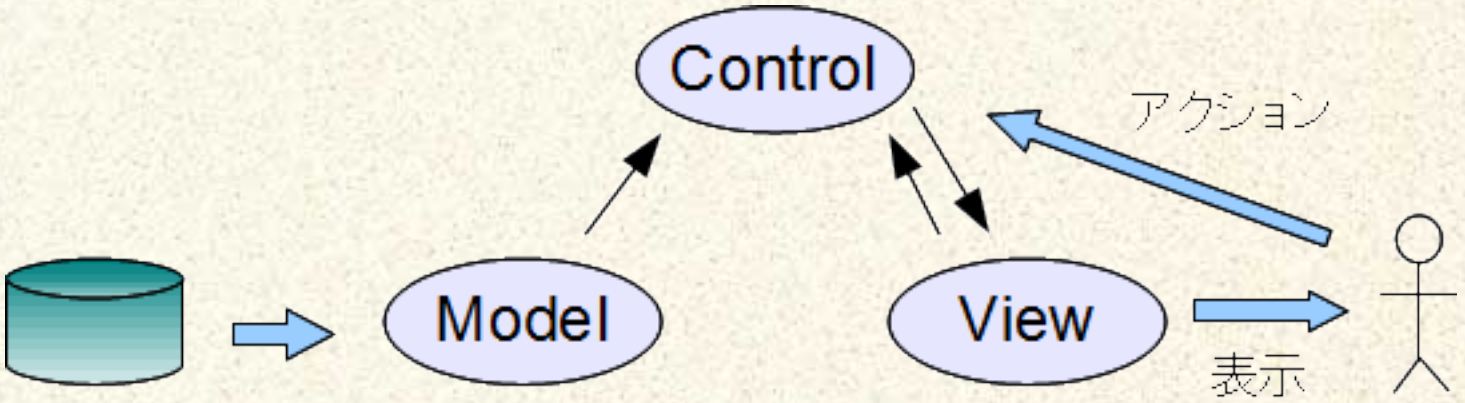
用語集プラグイン →
動機：昔作っていた用語の対訳管理システムを Redmine を利用すれば、用語集にできると思って。
参考：ezFAQ プラグイン

Redmine インフォメーションプラグイン →
動機：フローチャートなど管理者用の情報を見れず、不便だった。
参考：Redmine 本体のソース

プラグインのタイプ

- Wiki マクロの追加
  - {{toc}} のようなマクロで Wiki を拡張
    - Wiki マクロの追加方法(プログラマーズ雑記帳)
- Redmine の機能の拡張
  - 既存の機能を拡張
    - プラグインインターナル(r-labs)
    - プラグインホック(r-labs)
- 管理データの追加
  - チケットのような管理データを追加
  - こちらの方法を紹介

Railsの概要（MVC）→



	用途	**Rals** では
Model	データを扱う部分 （データはファイルやデータベース）	クラスのインスタンスがデータベースの１つのレコードに対応し、テーブルからの取得にはクラスメソッドを使用
View	表示、ユーザインターフェース部分	html を作成する部分。Erb で埋め込み式の ruby コードを評価して作成する
Control	ユーザの入力に応答して処理する部分	アクションに対応したメソッドを実行する

Rails の方針(CoC, DRY)

- 設定よりも規約　CoC (Convention Over Configuration)
  - 名前に規則を持たせることで設定の手間を省き、簡単に作成
    - ファイルを置く位置やファイル名に決まりがある。
    - FooBar クラスの定義ファイルは foo_bar.rb

- 繰り返しの禁止　DRY (Don't Repeat Yourself)
  - 同じことを何度も書かない
    - 表示の部分描画
    - レイアウト

プラグインのスケルトン →

最初にスクリプトを使ってテンプレートを生成。

```
$ ruby script/generate redmine_plugin プラグイン名
$ ruby script/generate redmine_plugin Standard
```

vender/plugins 以下に生成

```
init.rb                最初に読み込まれるファイル
app/
  ├ controllers/       コントロール
  ├ models/             モデル
  └ views/               ビュー
db/migrate/            データベース生成
config/locales         国際化
```

モデル(Model) →

プロジェクトID、題名、説明をメンバー に指定して生成

```
$ ruby script/generate redmine_plugin_model プラグイン名 モデル名
[カラム名:型 ...]
$ ruby script/generate redmine_plugin_model Standard Foo project_id:
integer subject:string description:text
```

```
app/models/foo.rb
db/migrate/20110612021254_create_foos.rb
```

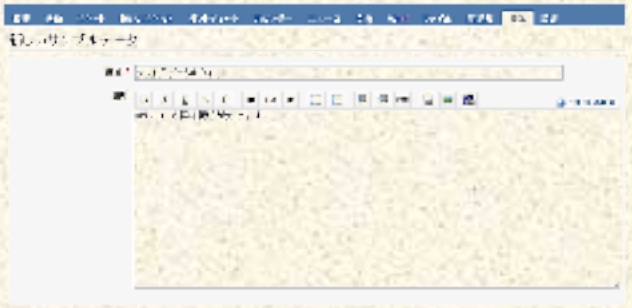
文字数制限などをモデルに記述。
DBのテーブル作成用のrake コマンドファイルも生成。

```
$ rake db:migrate_plugins
```

コントロール(Control) →

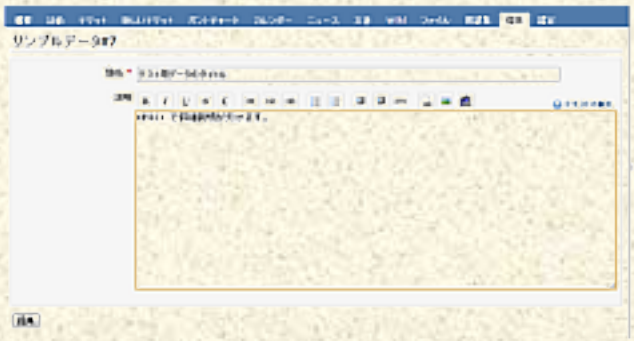
コントロールクラスでアクションごとのメソッドを生成

一覧表示



新規作成

詳細表示



編集

よく使われるアクション

index	一覧表示	DBからデータリストを取得し、表で表示
show	詳細表示	一つのデータの情報を表示
new	新規作成	入力を受け取り、データを追加
edit	編集	入力を受け取り、データを更新
destroy	削除	データを削除

コントローラクラスと各アクションの画面を生成

```
$ ruby script/generate redmine_plugin_controler プラグイン名 コントローラ名 [アクション名 ...]
$ ruby script/generate redmine_plugin_control Standard Foos index new show edit
```

```
app/controllers/foos_controller.rb
app/views/foos/index.html.erb
                :
```

ビュー(View)

View はコントロールと一緒に生成。
一つのアクションにつき、一つの html のソース

- 新規作成(new) アクション → app/views/foos/new.html.erb

Erb を使って html 内に埋め込まれたスクリプトを評価してページが表示

- **<% ... %>** : ruby スクリプトとして評価
- **<%= ... %>** : 評価の結果になる

```
<% for val in [1, 2, 3] %>
  <%= val %> <br />
<% end %>
```

```
1 <br />
2 <br />
3 <br />
```

init.rb

Redmine の起動時に読み込まれる。
プラグインを Redmine から利用するための処理を記述。

- プラグイン情報の登録

- プロジェクトモジュールの登録 project_module
- アクション権限を設定
  - 表示(index, show)
  - 管理(new, edit, delete)

- 最初のアクション(一覧表示)をプロジェクトメニューに追加

国際化 →

config/locals 以下に ja.yml, en.yml などの言語ごとのファイルを作成

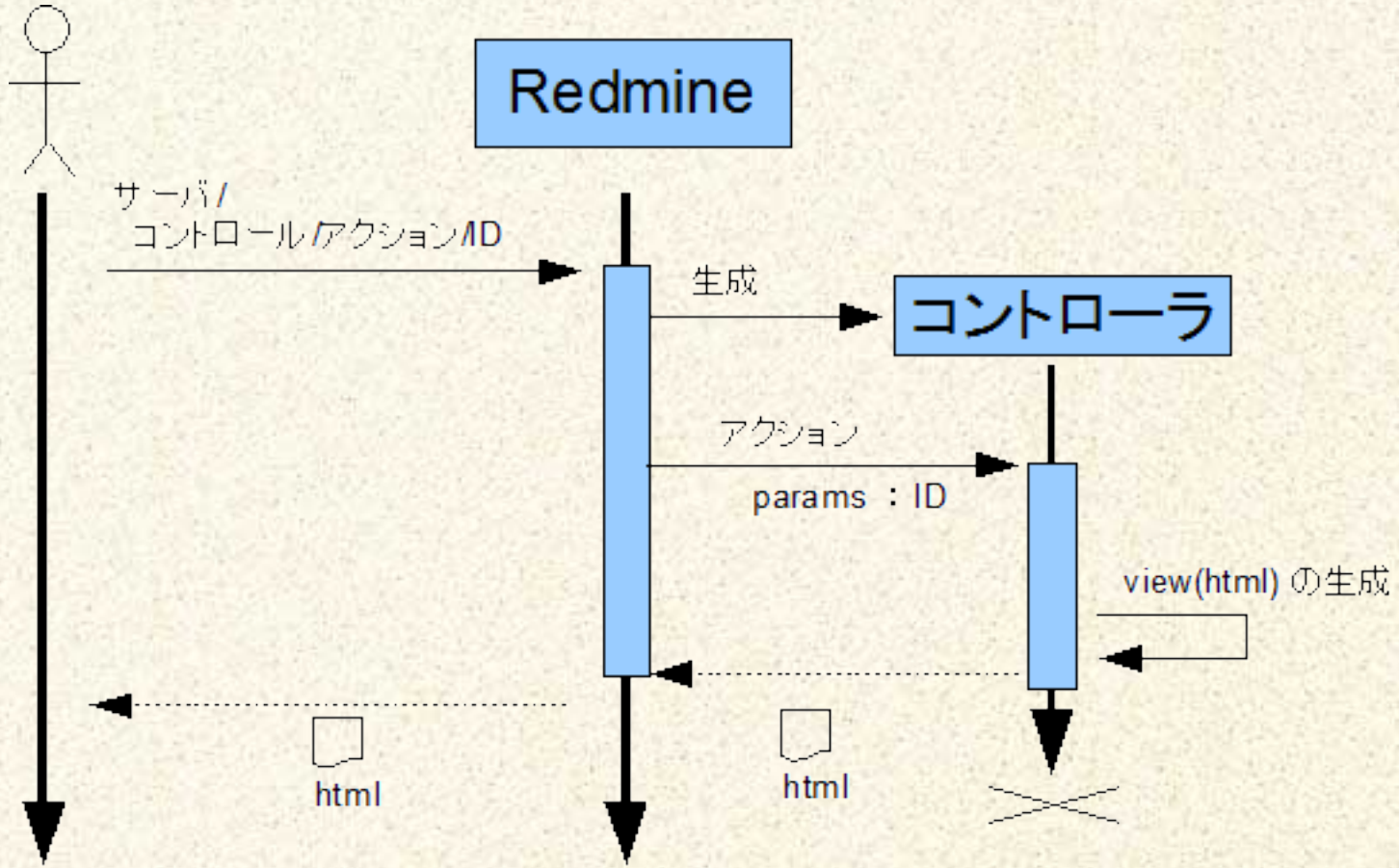
```
言語の識別子:
  シンボル名: メッセージ
```

```
ja:
  label_standard: 標準
  project_module_standard: 標準プラグイン
```

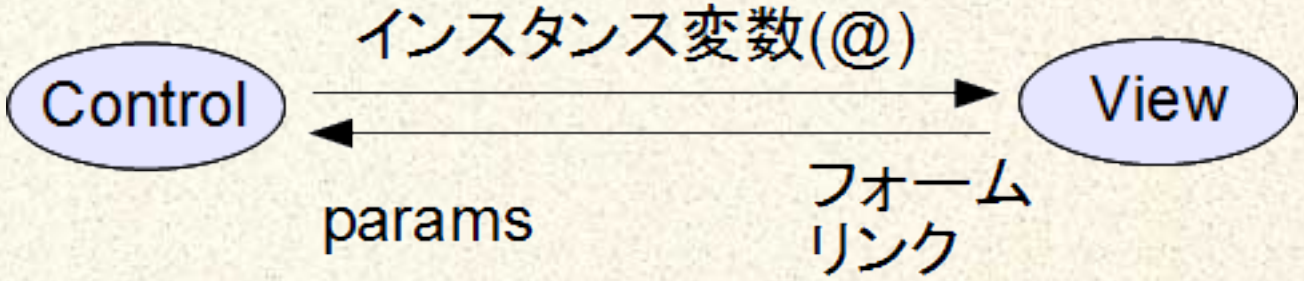
使う場合は **l** メソッドを使用

```
<h2> <%=h l(:label_standard) %> </h2>
```

データの流れ



リンクをクリック、フォームでデータ送信
→ コントローラ生成、アクション実行
→ html を生成



- ビュー → コントロール
  - リンク、フォームの引数が params 変数に格納

- コントロール <- モデル
  - コントロール内でモデルのメソッド呼び出し
    - 取得 : クラスメソッド(find)でインスタンスを取得
    - 操作 : インスタンスのメソッド(save, destroy)

- コントロール → ビュー
  - コントロールクラスのメンバー変数(@)

各アクションの処理

- 共通処理
  - Control : before_filter
    - ログインチェック authorize
    - @project の設定 find_project
      - **Project**.find(params[:project_id])
    - @foo の設定 (show, edit, delete) find_foo
      - **Foo**.find_by_id(params[:id])
  - View : レイアウト　タイトルの設定

- index 一覧表示
  - Control :一覧を取得　　@foos = **Foo**.find(:all, ...)
  - View : ループを回して、項目(show へのリンク) を表示
    - link_to(..., {:action=>:show, :project_id=> ... , :id => ...)

- show 詳細表示
  - Control : 何もしない（共通処理 で @foo を取得）
  - View : @foo の中身を表示

- new 新規作成、edit 編集
  - リンクからきた場合 (GET)
    - Control : @foo の用意（new は新規作成）
    - View : フォームの表示 → 結果は params[:foo] へ
  - フォームからきた場合 (POST)
    - Control : params[:foo] で @foo を作成
    - 保存 @foo.save して、show へ移動
- delete 削除
  - Control : 削除 @foo.destroy して、index へ移動

終わりに

これから Redmine に不足を感じたら、自分でプラグインを作って満足のいく Redmine に仕上げていってください。
そしてプラグインができたらどんどん公開して、Redmine をいいソフトにしていきましょう。

プラグイン作成方法のページ

- プラグイン開発ガイド
- redmine.org のデベロッパーガイド (日本語訳)
- プラグイン Tips
- Redmineプラグイン開発 (gihyo.jp)