概要

このモジュールは幾つかのメンバ変数によりデータ構造を定義し HTML::Template によるフォームを作成するだけで、環境設定のような設定値の保存ができる admin_menu のアクションハンドラが作れるベースクラスです。

例

■ plugin/hoge/Admin.pm

```
package plugin::hoge::Admin;
use Wiki::AdminTemplate;
use vars qw(@ISA);
@ISA = qw(Wiki::AdminTemplate);
sub new {
        my $class = shift;
        my $self = new Wiki::AdminTemplate('hoge');
        $self->{default_value} = {
                name    => "hoge",
                value   => "10",
                kind    => "alpha",
        };
        $self->{config_template} = {
                name    => '^\w+$',
                value   => '^\d*$',
                kind    => ["alpha", "beta"],
        };
        return bless $self,$class;
}
1;
```

■ plugin::hoge::Install.pm

```
package plugin::hoge::Install;
sub install {
        my $wiki = shift;
        $wiki->add_admin_menu(
                "hoge",
                $wiki->config('script_name')."?action=ADMINHOGE",
                200,
                "AdminTemplateTest",
        );
        $wiki->add_admin_handler('ADMINHOGE',"plugin::hoge::Admin");
}
1;
```

■ tmpl/hoge.tmpl

```
<h2>hoge プラグインサンプル設定 </h2>
<h3> 名前 </h3>
<p><input type="text" name="name" size="40"
          value="<!--TMPL_VAR NAME="NAME" ESCAPE="HTML"-->"></p>
<h3> 閾値 </h3>
<p><input type="text" name="value" size="4"
          value="<!--TMPL_VAR NAME="VALUE" ESCAPE="HTML"-->"></p>
<h3> 分類 </h3>
<p>
  <input type="radio" name="kind" value="alpha" id="kind_alpha"
         <!--TMPL_IF NAME="TYPE_ADMIN"-->checked<!--/TMPL_IF-->>
  <label for="kind_alpha">alpha</label>
  <input type="radio" name="kind" value="beta" id="kind_beta"
         <!--TMPL_IF NAME="TYPE_USER"-->checked<!--/TMPL_IF-->>
  <label for="kind_beta">beta</label>
</p>
<p>
  <input type="submit" name="SAVE" value=" 保存 ">
  <input type="reset" value=" リセット ">
</p>
```

詳細

このクラスを継承し、コンストラクタで幾つかのメンバ変数を初期化します。初期化が必要なメンバ変数は HTML::Template ファイルや値を保存する設定ファイル名のほか、データ構造を表すハッシュ変数、初期値を保持するハッシュ変数等があります。

ベースクラスのメソッドはそれらのメンバ変数にしたがって動作し、admin_menu 用のオブジェクトとして使用できます。

具体的には、

- 設定ファイル読み込みと HTML::Template 変数への反映
- 設定ファイルまたは POST されたデータの正当性チェックと設定ファイルへの反映

を行ないます。このため、このクラスを継承し、メンバ変数の初期化を行なうコンストラクタと HTML::Template 用のファイルを作れば環境設定機能を加わえる事ができます。

コンストラクタでは、ベースクラスのコンストラクタにプラグイン名を渡して呼び出す事で、そのプラグイン名の名前を元にページタイトルや主要ファイルの名前が設定できます。もちろん、その後に個別に変更する事も可能ですし、ベースクラスのコンストラクタは呼び出さず、独自に初期化しても問題ありません。

環境設定で設定できる様にするデータ名はメンバ変数 config_template で行ないます。このメンバ変数はハッシュ変数 (のリファレンス) となっており、キーが変数名、値がその変数が許容する文字列の正規表現または (ラジオボタンなどで) 選択すべき値となっています。また、メンバ変数 default_value にハッシュ変数を設定する事で初期値を設定できます。

HTML::Template 用のファイルは別途作る必要がありますが、メンバ変数 config_template の値を元に HTML::Template 用変数が作成され、config ファイルの値が設定されます。

これら保存された設定値を使用する他のオブジェクトは、継承して作成したこのクラスのオブジェクトを作成し、load_config メソッドを呼び出す事により、ファイルから読み出されてハッシュ変数へ格納された、バリデーションチェックを行なった値を読み出せます。

これらにより、環境設定機能を比較的簡単・安全に実装できます。

初期化すべきメンバ変数

■ config_template

```
$self->{config_template} = {
        name    => '^\w+$',
        value   => '^\d*$',
        kind    => ["alpha", "beta"],
};
```

■ default_value

```
$self->{default_value} = {
        name    => "hoge",
        value   => "10",
```

```
        kind    => "alpha",
};
```

コンストラクタにより初期化されるメンバ変数

■ title

環境設定ページに表示されるタイトル

■ file_config

設定値を保存するファイル名を指定する際の setup.dat での key 名

■ file_default

setup.dat に指定がない場合のデフォルト設定ファイル名

■ tmpl_name

テンプレートファイル名

■ action_name

環境設定を行なう際の action パラメータの値

load_config メソッドにより設定されるメンバ変数

■ flat

設定ファイルまたはフォームにより送られた値を操作するために、主に内部で使われるハッシュ変数です。主に操作を行なうのは、プライベートメソッドである _check_conf と _make_flat です。

load_config 終了時、通常は空になりますが、正しくパース出来なかった項目があれば、それが残されます。

■ conf

設定値を保持するハッシュ変数です。形式は default_value と同じようになります。load_config メソッド、set_config_form_param メソッドにより設定され、通常は load_config メソッドを呼び出した後に使用します。

■ tmpl

環境設定フォーム用の Template ファイルに渡す値を保持するハッシュ変数です。load_config メソッドにより設定されます。

コンストラクタ

```
my $self = new Wiki::AdminTemplate($plugin_name);
```

プラグイン名を引数にとります。ファイル名の一部に使用されるため英数字が前提です。

渡されたプラグイン名を元に以下の幾つかのメンバ変数を初期化します。

title
　プラグイン名 +" プラグインの設定 "

file_config
　　プラグイン名 +"_file"
file_default
　　プラグイン名 +".dat"
tmpl_name
　　プラグイン名 +".tmpl"
action_name
　　"ADMIN"+ 大文字のプラグイン名

メソッド

■ change_value

設定が変更された値毎に呼ばれるフックメソッドです。

■ after_save_hook

設定がセーブされたあとに呼ばれるフックメソッドです。

■ do_acton

アクションハンドラメソッドです。

SAVE パラメータの有無により設定値の保存 + リダイレクトとフォーム表示を切替えます。実際の動作は、保存 + リダイレクトは save_config メソッド、フォーム表示は config_form に引き継がれます。

■ config_form

設定フォームの表示を行ないます。

load_config メソッドにより設定値の読み込みとパラメータ設定を行なったあと、load_template メソッドによりテンプレートの読み込みとパラメータセットを行ない、最後に get_html メソッドにより HTML を生成して返します。

■ load_config

保存されている設定値の取得とバリデーションチェックを行ないます。

まず、設定値を flat メンバ変数に読み込み、_check_config プライベートメソッドにより、config_template にしたがってパースしつつ tmpl と conf に結果を返します。

管理メニュー時以外でも、オブジェクトを生成してこのメソッドを呼び出す事によりデフォルト値も含めたバリデーションされた設定値をハッシュによってアクセスするために使用します。

■ config_file

設定ファイル名をデフォルト名または setup.dat から取得して返します。

■ default_value

default_value メンバ変数からデフォルト値を取得して返します。

■ load_template

テンプレートファイルからテンプレートオブジェクトを生成し、load_config メソッドで生成されたテンプレート用パラメータを設定します。このテンプレートオブジェクトは後で get_html により使用されます。

■ get_html

すでに作成されているテンプレートオブジェクトからから実際に HTML を生成します。このメソッドの戻り値がこのプラグインの戻り値 (実際に出力される HTML) になります。

■ save_config

フォームによる更新を処理します。その際に、変更された値毎と、値がファイルに保存されたあとにフックメソッドを呼びます。

■ set_config_from_param

受け取った CGI パラメータから設定を更新し、保存用データを作成します。