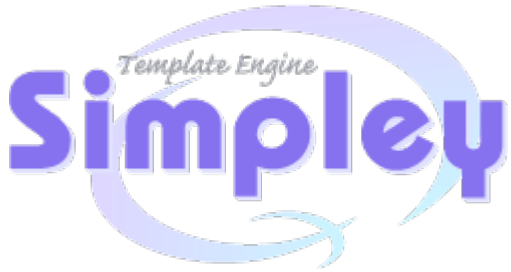

Template Engine
Simpley

2005/11/03	初版	Ver 0.1
2006/03/06	ダイアフラム・ディレクトリ構成追加（まだまだ作りかけ）	Ver 0.2
2006/03/07	モジュール実装・skin実装追加（まだまだ作りかけ）	Ver 0.3
2006/03/08	skin実装追加・Simpleyコア標準メソッド関数（まだまだ作りかけ）	Ver 0.4
2006/03/10	Simpleyコア標準メソッド関数追加・DBアクセス追加・DBセッション追加	
	携帯電話絵文字変換追加・その他参考資料追加	Ver 0.5
2006/03/14	Simpleyコア標準メソッド修正・Skin実装修正	Ver 0.6
2006/03/20	Simpleyモジュール実装修正	Ver 0.7
	Todo：携帯電話向け説明次回追加・・・	

この電子マニュアルはOpenOffice.org Ver2で作成しています。

PDFリーダーの選択ツールでテキスト選択できます。

Simpley とは

Simpleyは、PHPプログラミングにおいて、画面（Skin）とプログラム（Class）の分離(MVC)を容易に実現し、かつ汎用的なClass実装を目指すテンプレートエンジンです。

■特徴

Skinキャッシュ

スキン変換を行う際のオーバーヘッドを減らすために、Skinのキャッシュ機能を搭載しています。
既に変換されたページは次回からキャッシュを利用する事により、劇的な高速化が望めます。

DBアクセス

便利なDBアクセス関数群をSimpleyコアに内蔵しています。
これによりDBの種類を意識する事なく、DBと連携したWebアプリケーションの作成環境を提供します。
その他SQLインジェクション等を防止するための専用メソッドも装備されていますので、面倒なコードを意識することなく安全にDBを利用できます。
※対応DB：MySQL／PostgreSQL／Oracle／MS-SQL／SQLite

携帯電話絵文字変換

携帯電話端末向けにキャリア相互の絵文字変換機能を内蔵しました。これによりキャリアを意識することなくページのデザインが可能です。また、キャリア毎のスキンの切り分けも容易に出来るよう実装しています。

独自タグによるデザインビュー

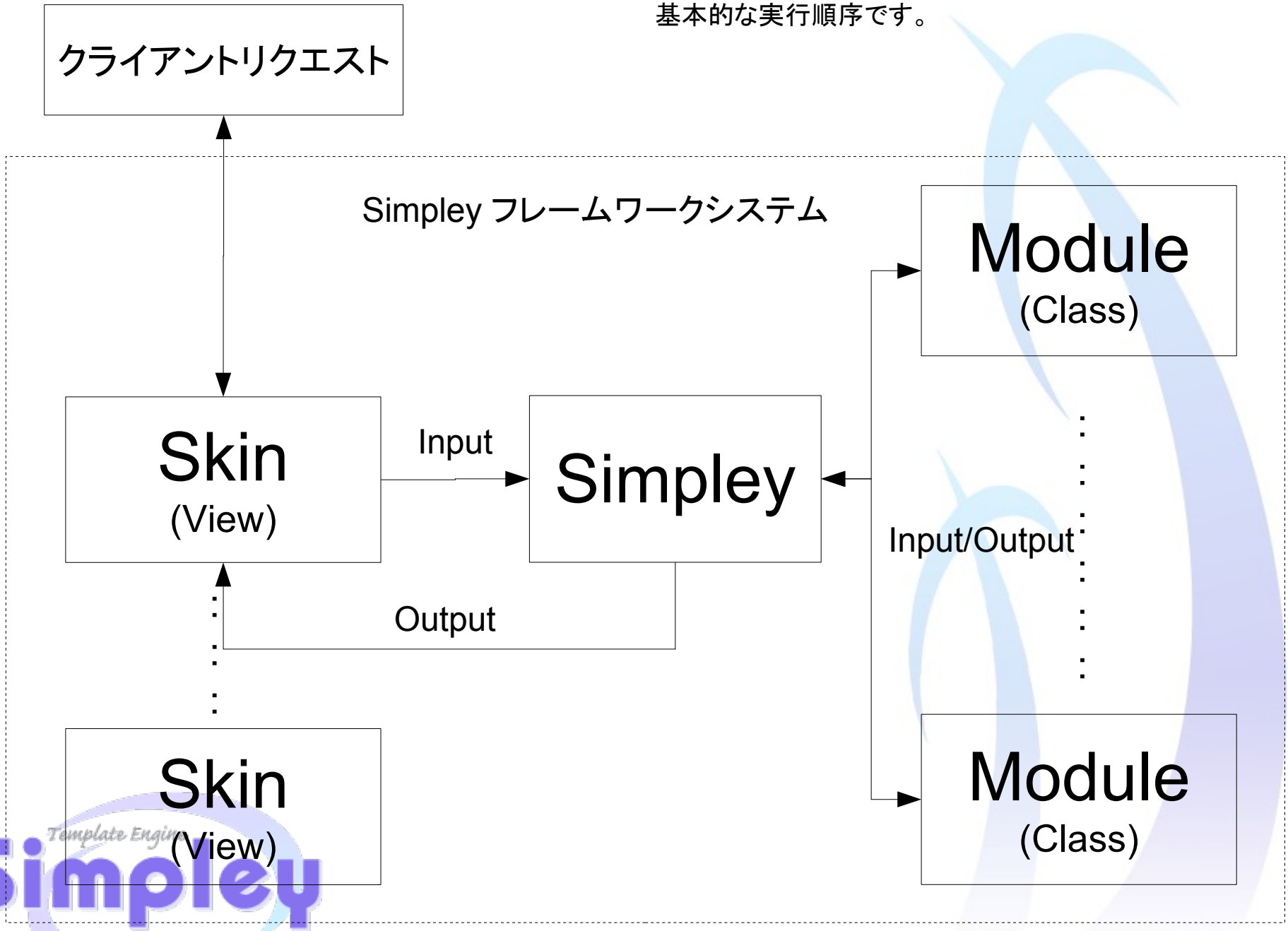
画面（Skin）はSimpley専用の簡単なタグを用いて市販のHTML編集ソフトでデザインを崩すことなく読み込む事が可能です。これによりプログラマとページデザイナーの完全分業を実現します。

■目次

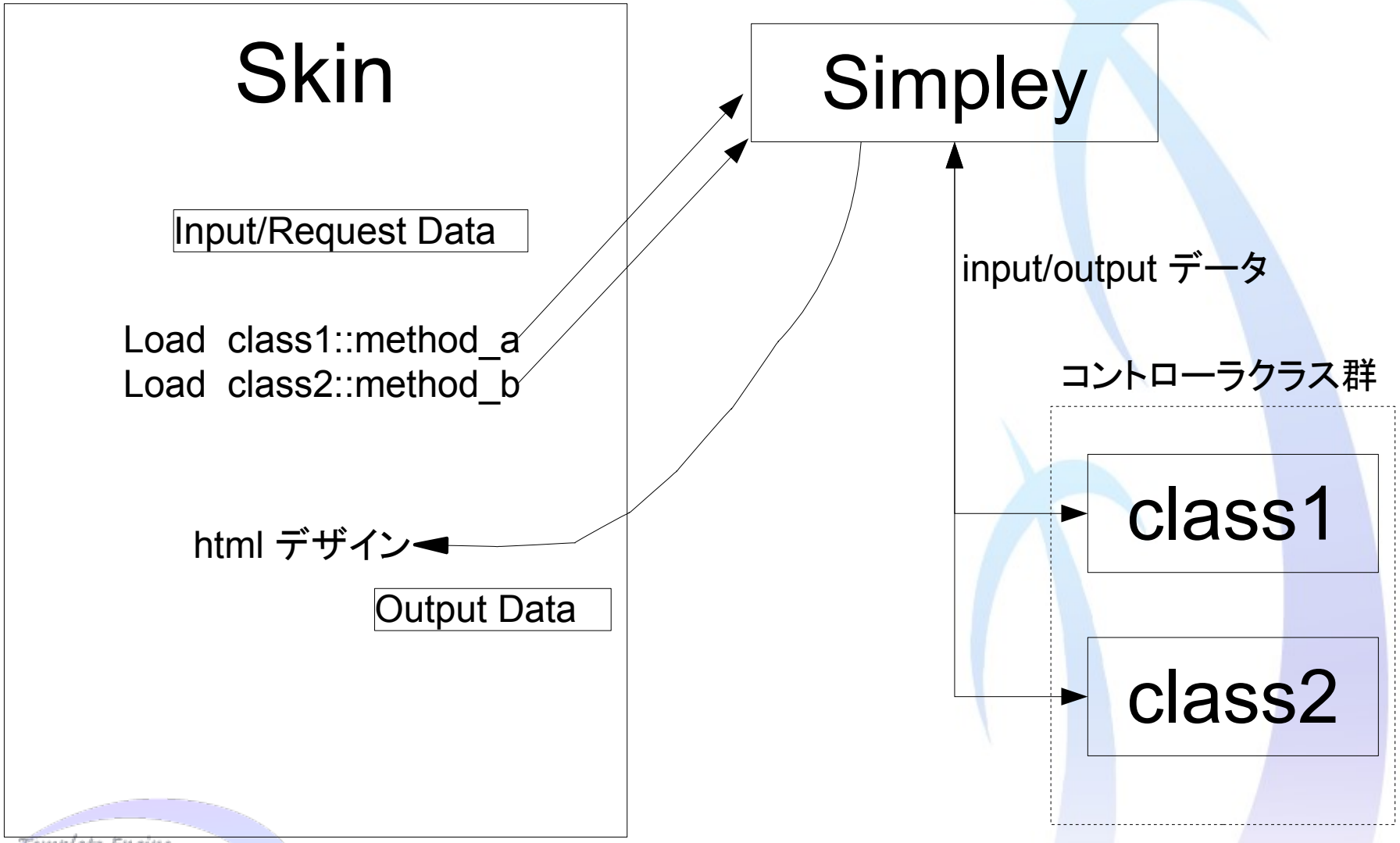
Template Engine Simpley

1.ブロックダイアフラム

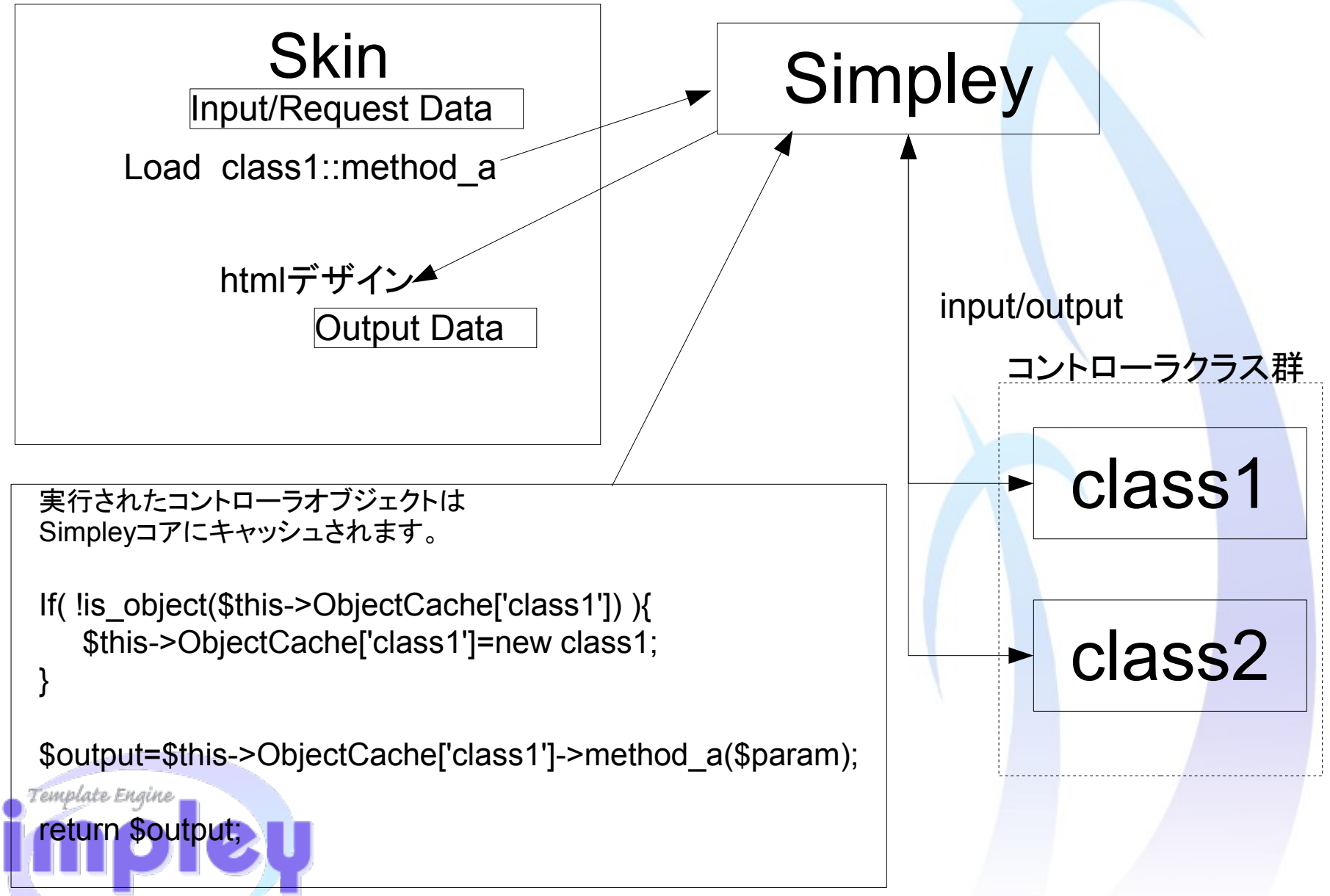
基本的な実行順序です。


クライアントリクエスト
Simpley フレームワークシステム
Module
(Class)
Skin
(View)
Input
Simpley
Input/Output
Output
Skin
(View)
Module
(Class)

コントローラクラスの読み込み

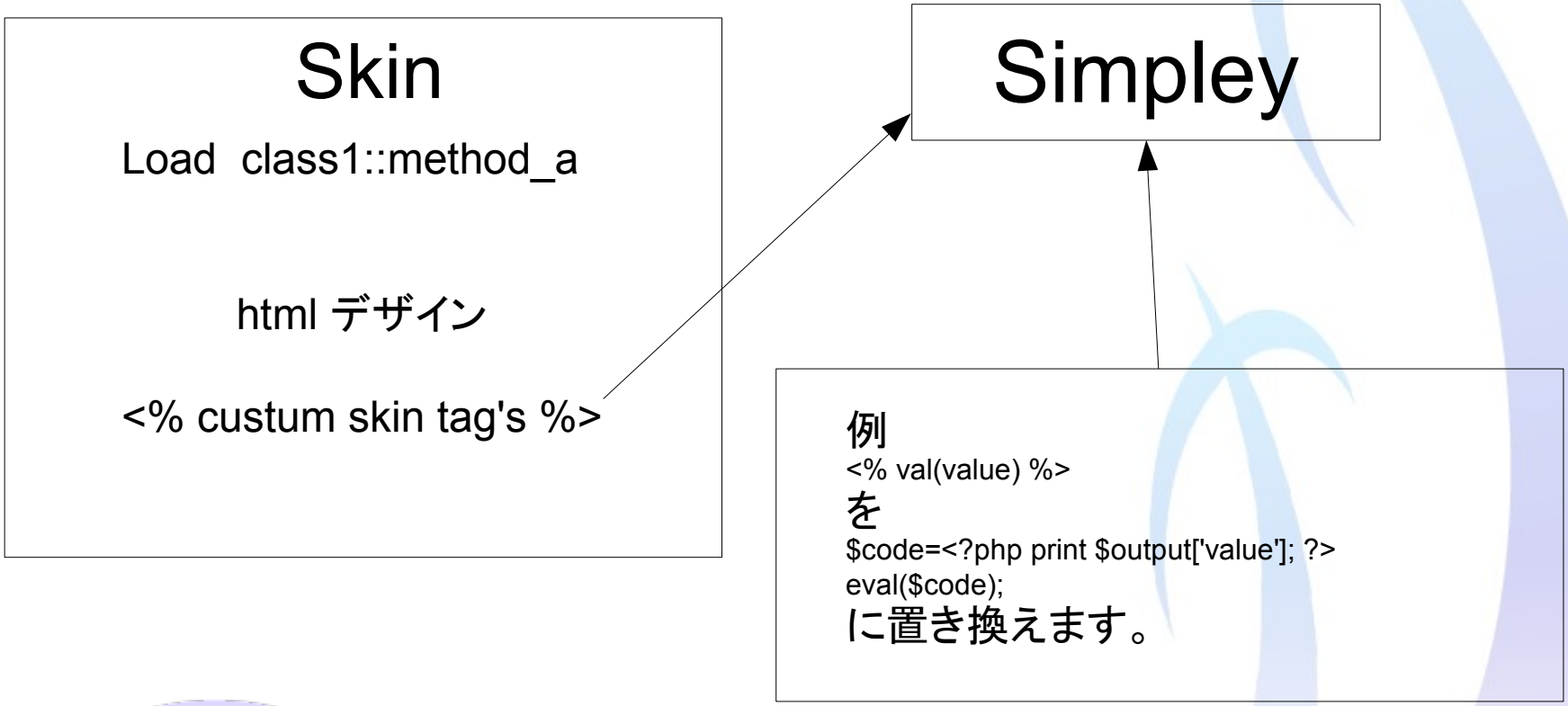


オブジェクトキャッシュ



Skinパースと出力の実行

クライアントのリクエスト要求より、skinタグを解釈します。
解釈された結果はPHPのコードに置き換えられます。



Skin パース＆ Skinキャッシュの実行

クライアントのリクエスト要求より、skinタグを解釈します。
解釈された結果はPHPのコードに置き換えられ、PHPコードとしてファイルに保存します。

Skin

Load class1::method_a

html デザイン

<% custum skin tag's %>

Simpley

例
<% val(value) %>
を
<?php print $output['value']; ?>

に置き換えキャッシュファイルに保存します。
次回リクエストよりキャッシュファイルを自動的に選択し実行します。

2.Simpley設置

■Simpleyを動作させるのに必要なサーバリソース

- Linux / Windows OS
- Apache Ver1/2　その他のPHPが動作可能webサーバ
- PHP4/5(mbstringが利用可能な事)
- 必要ならデータベースサーバ

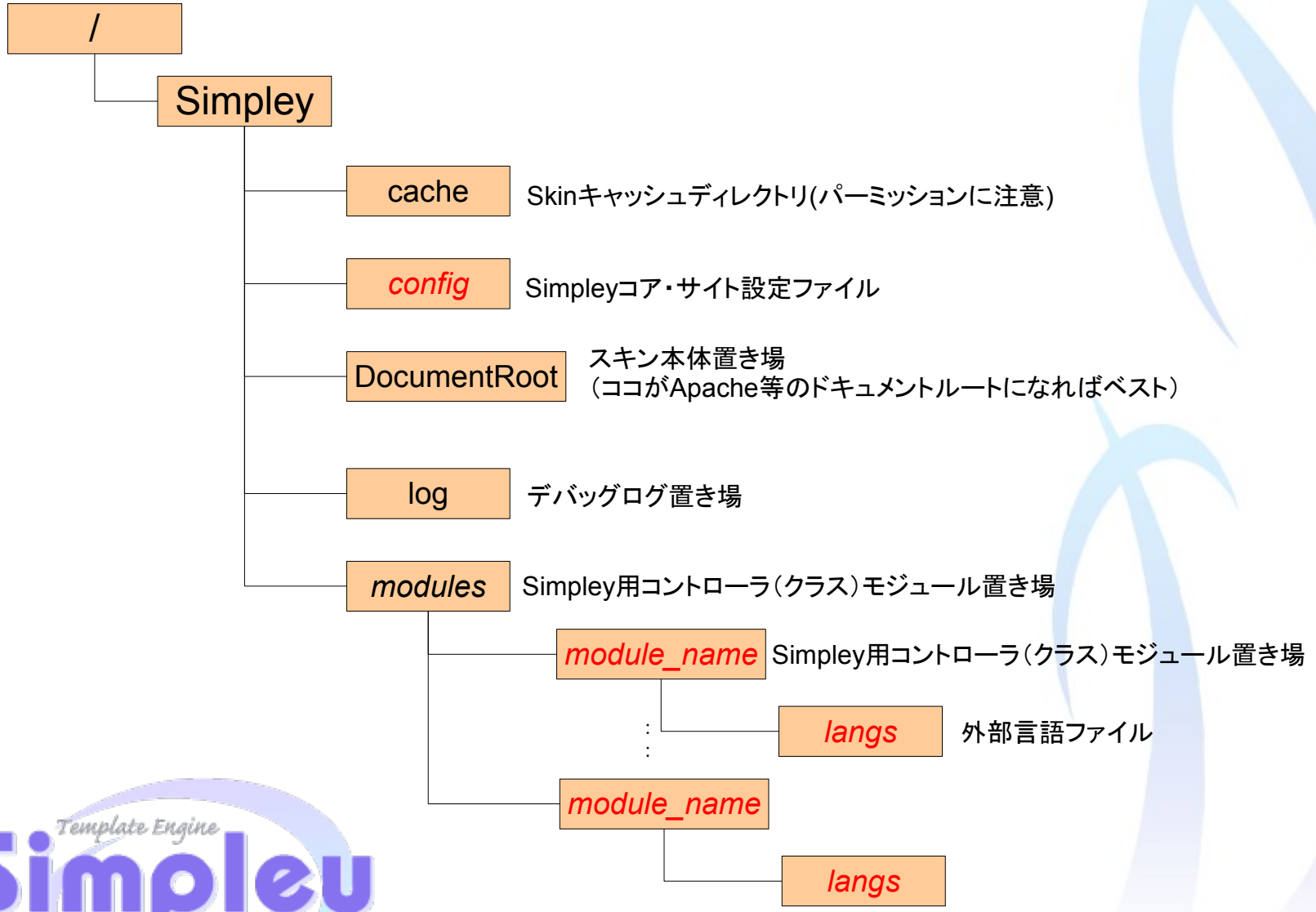
■Simpleyのディレクトリ構成について

- ディレクトリ構成はあらかじめ決められた場所に決められたソースファイルを設置し、明確的にソースの集中管理を目的としています。
- コントローラークラス（以下モジュール）にも命名規則を設け、skinから実行される、クラス・メソッドを明確にし、デバッグの向上をねらいます。

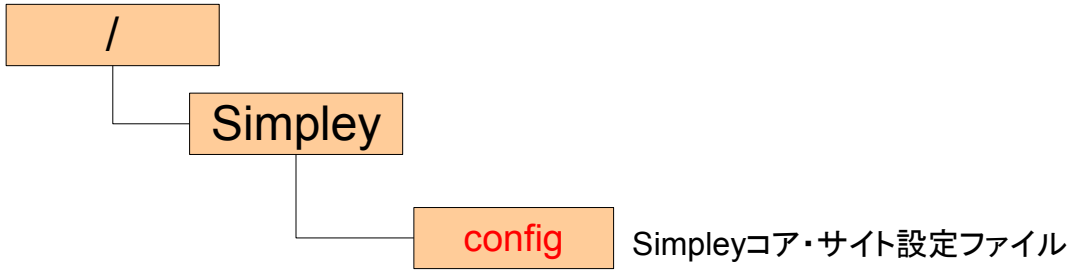
Template Engine Simpley

■Simpleyのディレクトリ構成

標準的なディレクトリ構成です。



■Simpley　設定ファイル等



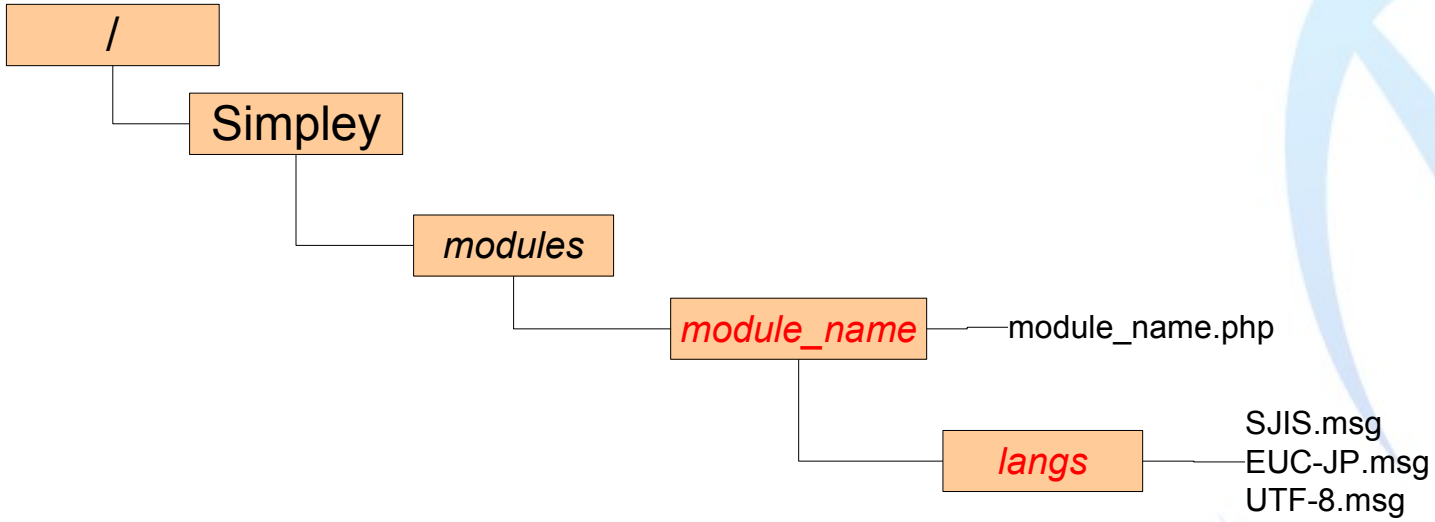
Config ディレクトリ以下にサイト設定ファイル・Simpleyコアクラス等が有ります。

サンプルに含まれる設定ファイルは *def.php* がサイト設定ファイルとなります。

ドキュメントルート以下に複数サイトを設置する場合でかつ設定が複数必要な場合は *def.php* を別名にコピーして設定内容を編集します。

設定項目は後のページで解説します。

■Simpley　モジュールの設置



modules ディレクトリ以下に、決められた命名規則に従って、コントローラ（クラス）モジュールを順に設置していきます。

module_name 以下に *module_name.php* となるようにモジュールを作成し、設置します。
具体的な実装方法は、後のページ（モジュール実装）で解説します。

*langs*ディレクトリ以下には各種内部文字コードに対応した言語ファイルを用意し設置します。
それぞれ文字コードに対応したファイル名は、*SJIS.msg EUC-JP.msg UTF-8.msg* 等になるようにします。

あらかじめマルチバイトを含むメッセージ類を言語ファイルに記述しておけば、サイト設定による、文字コードの違いをモジュールソースレベルでの修正ではなく、メッセージファイルベースでの修正（最小コストでの修正）にとどめることが可能となります。

■Simpley　サイト設定1

サンプルに含まれる設定ファイルは *def. php* がサイト設定ファイルとなります。

順に設定項目を解説します。

```
<?php
/**
 *  Simpley Site Configuretion File
 *  サイト毎にエンコーディング設定等が違う場合はこのファイルを別名にコピーして
 *  そのサイトに合うように設定してスキンでinclude してください。
 *  最後の方に補足情報も有りますので御一読ください。
 */

$MAIN_OPTION=array();
// いぢるなbenchi mark function
function getmicrotime(){
                        list( $usec, $sec ) = explode(' ',microtime());
                        return ( (float)$sec + (float)$usec );
}


/******************** Simpley Site Configuretion File ********************/

// キャッシュ対策(毎回リロードされるDebugやキャッシュされたくない用)
//header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
//header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

//メンテナンス画面に切り替える場合はコメントをはずす.中身は適当に編集してね。
//include_once("mente.html");
//exit;


/**
予備
define( 'SJIS'  , 'SJIS' );                 //Shift_JIS
define( 'EUC-JP', 'EUC-JP' );               //EUC-JP
define( 'UTF-8' , 'UTF-8' );                //Unicode
**/
```

次に続きます

■Simpley　サイト設定2

```
ignore_user_abort(false);                    //コネクションが切れてもスクリプトを実行するかどうか.
                                             //trueに設定した場合永久ループ動作するようなコーディングが有る場合危険です。タイ
ムアウトしません！！


ini_set("display_errors","1");               //エラーを画面に出力するかどうか
//ini_set("error_reporting","(E_ERROR | E_WARNING | E_PARSE)");
ini_set( "error_reporting" , E_ALL );        //エラー0を目指したい人向けw


//Bench Mark echo  スクリプトベンチマークタイム表示(あてにならん・・・)
$MAIN_OPTION['MAIN_DB']['BENCHI']  =true;                  // 表示
//$MAIN_OPTION['MAIN_DB']['BENCHI']  =false;               // OFF
if($MAIN_OPTION['MAIN_DB']['BENCHI']==true){ $MAIN_OPTION['MAIN_DB']['B_START']=getmicrotime(); }


/***** ディレクトリ設定 *****/
/*
  ドキュメントルートとは違います！！
  設定ディレクトリ以下に configや modulesが入るように設定します！
  推奨はドキュメントルートの1つ前のディレクトリに設置がいいかな。
*/
$MAIN_OPTION['MAIN_DIR']                ='/var2/php';          //コアメインディレクトリ
define('MAIN_DIR',$MAIN_OPTION['MAIN_DIR']);
```

次に続きます

■Simpley　サイト設定3

```
/***** 内部エンコード設定 *****/
$MAIN_OPTION['MAIN_LANG']='EUC-JP';                    //内部エンコーディング
//$MAIN_OPTION['MAIN_LANG']='SJIS';                    //内部エンコーディング
//$MAIN_OPTION['MAIN_LANG']='UTF-8';                   //内部エンコーディング
define('MAIN_LANG',$MAIN_OPTION['MAIN_LANG']);


/***** 出力エンコード設定 *****/
$MAIN_OPTION['MAIN_DB']['OUTPUT_LANG'] = 'SJIS';      //出力エンコード
//$MAIN_OPTION['MAIN_DB']['OUTPUT_LANG']= 'EUC-JP';//出力エンコード
//$MAIN_OPTION['MAIN_DB']['OUTPUT_LANG']= 'UTF-8';   //出力エンコード
define('OUTPUT_LANG',$MAIN_OPTION['MAIN_DB']['OUTPUT_LANG']);


/***** 携帯絵文字エンコード設定 *****/
/*   !!!!!!!!!注意!!!!!!!!!!
    絵文字変換を利用する場合は出力コードは必ずSJISに設定してください（絵文字変換きかなくなるので）.
    絵文字変換を使う場合は文字コードが内部と外部が違う場合は絵文字モジュールでout-in in-out変換します。
    絵文字変換を使わない場合は false 使う場合は true に設定してください
*/
$MAIN_OPTION['MAIN_DB']['EMOJI_ENCOD'] = false;
$MAIN_OPTION['MAIN_DB']['EMOJI_ENCOD'] = true;


/***** テンプレートエンコード変換設定 *****/
/* htmlも内部エンコードと同じ場合(内部エンコードEUC 外部SJIS等でテンプレートもEUCにしてある場合)
   出力する際内部から出力エンコードに変換
   false = しない
   true = する
   phpのoutput_handler に mb_output_handler を設定して自動変換させている場合は false 設定にしてください.
   (変数名がおかしいけど気にしないで・・・)
*/
$MAIN_OPTION['MAIN_DB']['EMOJI_ENCOD_HTML'] = false;
$MAIN_OPTION['MAIN_DB']['EMOJI_ENCOD_HTML'] = true;
```

ここで行う文字コードの設定は自動的に変換される事を意味している訳ではありません。
あくまで入力と出力・DBの文字コードの差異をSimpleyコアに情報として与えるためです。

入出力に対して文字コード変換を行うアプローチは2つ有ります。

・PHPで自動変換させる（php.ini等の設定変更要）。
・コントローラモジュールで入出力に対するコード変換を書いておく。

また、Simpleyは内部文字コードと出力文字コードが違う場合、かつphpで自動変換させてない場合は自動変換して出力します。phpでさせる場合は変換をしません。

入力文字コードに関しては、Simpleyコアメソッドを利用することにより透過的処理を行います（上記の逆の動作）。

詳しくはSimpleyコアメソッド解説参照。

次に続きます

■Simpley　サイト設定4

```
/***** テンプレートキャッシュ設定 *****/
/* !!!!!!!!!注意!!!!!!!!!!
   phpがsalfモードで動作しているときは利用出来ません！！
*/
$MAIN_OPTION['CACHE']=true;                                    //キャッシュ有効
//$MAIN_OPTION['CACHE']=false;                                 //キャッシュ無効
$MAIN_OPTION['CACHE_FILETIME']=86400;                          //キャッシュの有効時間（秒）86400=1日
define('SIMPLEY_CACHE_CHMOD',0777);                            //キャッシュ書込時のパーミッション(chmod)
/* キャッシュディレクトリ
  (デフォルトは メインディレクトリ以下のサブディレクトリ設定) */
$MAIN_OPTION['CACHE_DIR']=MAIN_DIR.'/cache';
//$MAIN_OPTION['CACHE_DIR']='/tmp/php';


/***** データベースセッション設定 *****/
                                        //セッションにDBを利用する
$MAIN_OPTION['SID_DB']=true;            //DBセッション利用
//$MAIN_OPTION['SID_DB']=false;         //利用しない(php のファイルモード)
$MAIN_OPTION['MAIN_SESSION_TIMEOUT']=3600;          //セッションタイマー(指定は秒 3600sec=1h) DBセッションの場合


/***** DBエラーメール通知設定 *****/
$MAIN_OPTION['MAIN_DB']['ERROR_MAIL']=true;             //DBで何らかのエラーが発生した場合
//$MAIN_OPTION['MAIN_DB']['ERROR_MAIL']=false;          //メールを送信するかどうか(mailモジュール必須)
$MAIN_OPTION['MAIN_DB']['ERROR_MAIL_TO_ADDRESS']   ='to_mail@domain.com';          //送信先メールアドレス
$MAIN_OPTION['MAIN_DB']['ERROR_MAIL_FROM_ADDRESS'] ='from_mail@domain.com';        //差出人メールアドレス
$MAIN_OPTION['MAIN_DB']['ERROR_MAIL_SUBJECT_CONNECT']='DBコネクションエラー';      //接続出来なかった時のサブジェクト
$MAIN_OPTION['MAIN_DB']['ERROR_MAIL_SUBJECT_QUERY']  ='DBクエリエラー';            //クエリエラー時のサブジェクト
```

次に続きます

■Simpley　サイト設定5

```
/***** デバッグ用設定 *****/
$MAIN_OPTION['MAIN_DEBUG']['EXEC']=false;                //デバッグログに出力するかどうか(PHP5だとなぜか動かないので注意)
//$MAIN_OPTION['MAIN_DEBUG']['EXEC']=true;               //false 無効 , true 有効
$MAIN_OPTION['MAIN_DEBUG']['LOG_PATH']=MAIN_DIR.'/log/debug_log'.date("Ymd").'.log';        //ログファイルの場所

/***** データベース設定 *****/
/* 利用したいDBを選択(Postgres/MySQL/oracle/SQLite/MS-SQL) */
//$MAIN_OPTION['MAIN_DB']['DB']='postgres';              //PostgreSQL
$MAIN_OPTION['MAIN_DB']['DB']='mysql';                   //MySQL
//$MAIN_OPTION['MAIN_DB']['DB']='oracle';                //ORACLE
//$MAIN_OPTION['MAIN_DB']['DB']='sqlite';                //SQLite
//$MAIN_OPTION['MAIN_DB']['DB']='mssql';                 //MS-SQL
//$MAIN_OPTION['MAIN_DB']['DB']='';                      //DB無しの場合は必ず空文字を！！
define('MAIN_DB', $MAIN_OPTION['MAIN_DB']['DB'] );

$MAIN_OPTION['MAIN_DB']['HOST']='192.168.1.111';         //DBホスト名
$MAIN_OPTION['MAIN_DB']['PORT']='3306';                  //DB接続ポート
//$MAIN_OPTION['MAIN_DB']['PORT']='5432';                //DB接続ポート
$MAIN_OPTION['MAIN_DB']['DB_NAME']='simpley';            //DB名
//$MAIN_OPTION['MAIN_DB']['DB_NAME'] = MAIN_DIR.'/config/Simpley.sqlite2';                    //SQLiteの場合はファイルパスを
$MAIN_OPTION['MAIN_DB']['DB_USER'] = 'simpley';          //DBユーザー名 SQLiteの時は空文字
$MAIN_OPTION['MAIN_DB']['DB_PASS'] = 'db_password';      //DBパスワード SQLiteの時は空文字
$MAIN_OPTION['MAIN_DB']['DB_LANG'] = 'EUC-JP';           //DBエンコーディング(PostgreSQL用)
$MAIN_OPTION['MAIN_DB']['SQL_DEBUG'] = true;             //SQLデバッグの設定（デバッグログに出力されます）
$MAIN_OPTION['MAIN_DB']['SQL_DEBUG'] = false;            //true=有効 false=無効

define('DB_LANG', 'EUC-JP' );            //Simpleyで自動変換したい場合の設定　使わない場合は内部エンコードと同じ設定にしておく！
//define('DB_LANG', 'SJIS' );            //DBのエンコードタイプを選択してください.中身はいじらないでね。
//define('DB_LANG', 'UTF-8' );           //(PostgreSQLの設定とは違います！！) SQLite利用時はたぶんUTF-8の方がいいかもしれない。。。



/***** magic_quotes_gpcの状態を保存しておく(いじらなくてよし) *****/
$MAIN_OPTION['QUOTES']=get_magic_quotes_gpc();       //オフの場合 0、オンの場合 1
```

お疲れ様でした。以上でSimpleyのサイト設定は完了です。
次はskinパースを自動で行う為の設定（Apache)と、推奨する
php.iniの設定です。

■Simpley　サイト設定6

この章では、skinパースする際、どのサイト設定を使うかを自動的に行う設定です。

この設定をしない場合はskinの行頭に include_once() 又は require_once()にて、サイト設定ファイルを読み込む必要が有ります。これは、あまり良いとは言えません。なぜならば、同じスキンを異なるサイトで使い回す場合、サイト設定ファイルの指定を全て書き直さなければならないからです（とても面倒）。

今回の場合は以下の想定で設定します。

内部文字コード EUC-JP
出力文字コード　SJIS
magic_quotes_gpc ON
php による入力文字自動変換ON
サイト設定ファイルに site_one.php を利用
以上の想定で Apache の *.htaccess* を用意します。

```
php_value magic_quotes_gpc on
php_value mbstring.http_input auto
php_value mbstring.internal_encoding EUC-JP
php_value mbstring.http_output SJIS
php_value auto_prepend_file “/Simpley/config/site_one.php"
```

以上で 上記想定を実現する .htaccess となります。
設定確認には、ドキュメントルートに phpinfo(); を記述したファイルを用意し、上記設定項目と一致しているか確認してください。

詳しくは Apacheのマニュアル参照の上、各自の環境に合うように設定してください。

■Simpley　推奨するphp.iniの項目

この章では、phpでマルチバイト文字（日本語）を扱う上で重要な設定となります。

php.ini

output_handler = “mb_output_handler"
output_buffering = 8192
mbstring.detect_order = “auto"
mbstring.encoding_translation = “On"
mbstring.language = “Japanese"

以下項目は サイト設定により.htaccess等で変更してください。
mbstring.http_input = “auto"
mbstring.http_output = “SJIS"
mbstring.internal_encoding = “EUC-JP"

絵文字変換モジュールを利用する場合は必ず
magic_quotes_gpc = off
mbstring.http_input = “pass”
mbstring.http_output = “pass”
mbstring.internal_encoding = 任意
の設定にしてください。

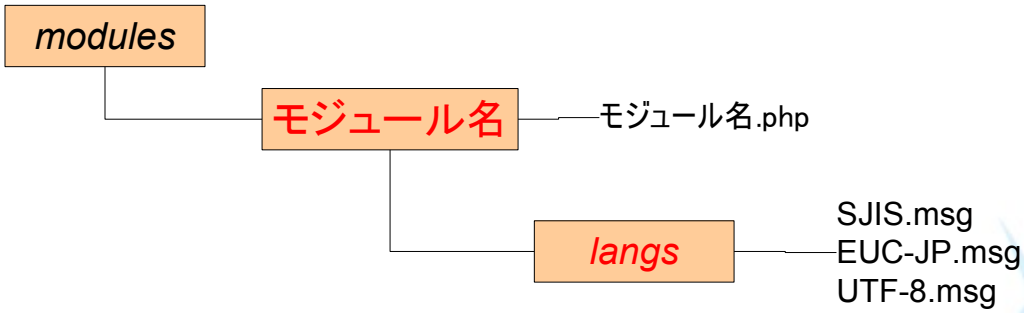
詳しくは phpマニュアル参照の上、各自の環境に合うように設定してください。

3.モジュール実装

■Simpley　モジュール（コントローラクラス）実装

Simpleyではコントローラクラスを*モジュール*と呼びます。
モジュールの設置方法にて先ほど述べましたが実際に実装する際の方法を解説します。

前提条件として、下記ディレクトリ構成を必ず厳守します。
Skinから呼び出すモジュールを決定しますので、順序が正しくないと正常に実行されません。



例として、以下のモジュール作成することとします。

Class名　　example
ファイル名　example.php
格納ディレクトリ名 example
言語ファイルディレクトリ名 langs（固定）とし、各文字コードに対応するファイル名を設置
。
（言語ファイルについては後に解説します）

■Simpley　モジュール実装

通常クラス実装する場合は以下のようにします。(PHP4での書き方)

```
<?php

class example{

  //コンストラクタ
   function example(){
   }

   function method_a(){
   }

   function method_b(){
   }

}

?>
```

■Simpley　モジュール実装

Simpleyで実装する場合は以下のようにします。(PHP4での書き方)
コアに内蔵している標準メソッド等を利用したりするのに役立ちます。

```
<?php

class example{
   var $module="example";    //モジュール(クラス)名をここで決め打ちしておく(Simpley利用時必須)
   var $message;             //モジュール用外部言語ファイルメッセージ(Simpley利用時必須)
   var $message_main;        //コアメッセージ類(Simpley利用時必須)
   var $main_option;         //オプション設定情報(Simpley利用時必須)
   var $simpley;             //コアオブジェクト標準メソッド等(Simpley利用時必須)
  //コンストラクタ
   function example(){
      $this->init();
   }
   /*
       初期化
      init()がかぶってたら他の名前に変更しコンストラクタへ追加
   */
   function init(){
      clearstatcache();
      global $MESSAGE_MAIN;                //コアメッセージ(Simpley利用時必須)
      global $MAIN_OPTION;                 //全設定(Simpley利用時必須)
      $this->main_option  =$MAIN_OPTION;    //全設定(Simpley利用時必須)
      $this->message_main =$MESSAGE_MAIN;    //コアメッセージ(Simpley利用時必須)
      if( is_file(MAIN_DIR.'/modules/'.$this->module.'/langs/'.MAIN_LANG.'.mes') ){
         include(MAIN_DIR.'/modules/'.$this->module.'/langs/'.MAIN_LANG.'.mes');
      } else {
         echo $this->message_main["ERR_LANG"].':'.MAIN_DIR.'/modules/'.$this->module.'/langs/'.MAIN_LANG.'.mes';
      }
     $this->simpley = new Simpley;
   }

   function method_a( $param ){
   }

   function method_b( $param ){
   }

}
?>
```

クラスのコンストラクタにSimpleyを利用するにあたって必要な情報をメンバ変数にセットします。

この例ではinit()としていますが他の名前に変更してもかまいません。又その他必要なメンバ変数等も自由に設定できます。

ここで内部の文字コードに対応する言語ファイル等も読み込まれます(P29参照)。

こうした背景は以下の理由があります。

- 異なる文字コードで同じモジュールを使い回す場合等。
- 固定的なメッセージ出力をソースに直接書き込むことなく、言語ファイルのみの修正にとどめるため。
- モジュール固有の設定値などもここに設定することによりソースの無用な改変を行うことを減らす。

上記の事により、無駄な時間や、無用なバグ等を削減することが出来ます。

■Simpley　モジュール実装

モジュール実装する際、現在の設定を知るための定義済みの定数が有ります。
定数定義は、サイト設定ファイルで設定された物が定義されます。
主に利用する定数は現在利用しているDBや文字コードといった場合が多くなるでしょう。
以下に定義済みの定数を記します。

■定義済み定数

MAIN_DIR	メインディレクトリの内容が定義されています。
MAIN_LANG	内部エンコーディングが定義されています。
OUTPUT_LANG	出力エンコーディングが定義されています。
SIMPLEY_CACHE_CHMOD	スキンキャッシュ生成時のアクセス権が定義されています。
MAIN_DB	利用するデータベースが定義されています。
DB_LANG	利用するデータベースのエンコーディングが定義されています。
TAGCONV	スキンのタグ解析用正規表現が定義されています。

MAIN_DIR フルパスでメインディレクトリを取得できます。/modulesを追記することでモジュールディレクトリを取得するのに利用できます。

MAIN_LANG 内部文字コードが定義されています。SJIS/EUC-JP/UTF-8等PHPに設定する項目が定義されています。

OUTPUT_LANG 出力文字コードが定義sれています。MAIN_LANGと同様です。

SIMPLEY_CACHE_CHMOD　skinキャッシュディレクトリが定義されています。

MAIN_DB　現在そのサイトで利用しているDB名が取得出来ます。mysql/postgres/oracle/mssql/sqlite等が定義されています。

DB_LANG　データベースで利用している文字コードが取得できます。DBを参照する際、内部文字コードと差異が有る場合変換する必要がある場合に利用できます。

TAGCONV　通常は意識しません。skinパースするさいのskinタグの正規表現が定義されています。

■Simpley　モジュール実装

前準備が出来たら実際にコントロールメソッドを実装します。Simpleyでは、これらを***ブロック***と呼ぶこととします。

本ケースの場合は *example モジュールの method_a ブロック* といった感じになります。

ブロックはskinに対して値を出力したり、リクエストの値を受取モデルクラスを利用しデータセットを返したり登録したりする部分です。PHPべた書きの場合は画面等にこれらを実装しますが、Simpleyでは全てモジュール・ブロックがそれらに該当するように実装します。
出力したい値は全て配列にしreturnします。
また各ブロックのオーギュメントはskinパラメータとして配列で値を取得することができます。
(skinパラメータ等はskin実装編にて解説します)

Skin から呼び出されたメソッドは以下のように実装します。

例

```
// $paramはskinパラメータを受け取ることが出来ます
function method_a( $param ){
        $out=array();    //出力用配列初期化
        $str='hello Simpley';
        $out['string']=$str;
        return $out;
}
```

return は スキン中で使う各種タグへマップされますので、配列（若しくは連想配列）で値を返す必要があります。

■Simpley　モジュール実装

外部言語ファイルの利用。

最初の説明に書いたように、コンストラクタ実行時に外部言語ファイルを読み込みます。
以下の例の用に利用することにより、異なる文字コードでの再利用性を高めます。
また、モジュール固有の設定等も記述することによりスケーラビリティーの高いコードを実装することが可能です。
(ハードコーディングによる修正範囲の縮小)

例

```
// 外部言語の利用
function method_a( $param ){
        $out=array(); //出力用配列初期化
        $message='';  //固定メッセージ用変数初期化

        if( !isset($_POST['input']) ){
              $_POST['input']='';
              $input='';
        }else{
              $input=$_POST['input'];
        }

        if($input==''){
            $message.=$this->message['noparam'];
        }


        $out['message']=$message;
        return $out;
}
```

例

```
    // 外部言語ファイルの内容

<?php

$this->message['noparam']="入力値が有りません。";



?>
```

■Simpley　モジュール実装

例の様に、ブロックはskinパラメータ以外は自由に実装できます。
ユーザーからのデータを受け取る際は $_GET $_POST等を直接扱います。
しかしながら、外部から来るデータをそのまま使うのはセキュリティー上問題が有る場合が有ります。
例としてSQLインジェクション等が上げられます。
（透過的に外部から来るデータを処理する他のクラスを利用するような場合はモジュールのコンストラクタで実行しておくと便利でしょう）
このような外部から来るデータをサニタイジングやエスケープ処理をしたり、外部内部文字コードの変換を明示的にやりたい場合にSimpleyの標準メソッドが利用できます。（これらは後にまとめて紹介します）

```
例.
//利用しているDBに合致するエスケープ処理をします。DB設定無しの場合はaddslasheです。
//php.ini等の設定でmagic_quotes_gpcがONの場合は何もせずそのまま値を返します。
// intvalはPHPの標準関数です。
$id=intval( $this->simpley->DBquote( $_GET['id'] ) );
```

その他、モジュールには、そのコントローラ特有のユーザー定義関数の代わりに、自由なメソッドを実装することができます。
それらは、通常skinから呼び出す様な使い方はしないようにし、オーギュメントの数の制約等もなく自由に実装できます。

```
例
    // ユーザー定義のメソッド
    function getUsername( $id ){
            //$idからユーザー名を取得する実装
            return $username;//名前を返す
    }
```

4.Skin実装

■Simpley　Skin実装

この章では、skinから実装したモジュールを呼び出し、モジュールが返す値をパースする実装方法を解説します。

基本的なよく使うskinタグを下記に記します。詳細なskinタグについては後のページから解説します。

■モジュール実行タグ
<% block(module,block,$option) %>

インラインスキン
<% skin(module,block) %><% /skin %>
外部ファイルスキン
<% extskin(module,block,"skin_file") %><% /skin %>

■出力タグ
<% val(values) %>

■繰り返しループタグ
<% each() %><% /each %> (<% loop() %><% /loop %>)

■条件指定
<% def(values) %><% /def %>

■条件指定2
<% if() %>(<% else(elseif) %>)<% if%>

■Simpley　Skin実装

ブロック実行タグ
<% block(*module_name,block_name,options*) %>

内部的にはmodule_nameクラスをインクルードし
$object = new modules_name;をする。
その後 block_name を
$output=$object->block_name($option); を実行する。
optionsは配列に置換されます。

optionの書式はGET引数の様に書きます。
例.
```
<%
block(module,block,PAGE=1&OFFSET=0&LIMIT=20
) %>
```
モジュール内でのoptionはこう渡ってきます。

```
function block( $option ) {
  $page=$option['PAGE'];
  $offset=$option['OFFSET'];
  $limit=$option['LIMIT'];
  //の様になります。
}
```

例.
```
<% block(news,index,page=10) %>
```
news.phpクラス
indexメソッド
内部動作的には
```
$object=new news();
$option=array('page'=>'10');
$return = $object->index($option);
return $return;
```
となります。

例２.
```
<% block(old_news,index,page=10&headline=red&topindex=bule) %>
```
old_news.phpクラス
indexメソッド.内部動作的には
```
$object=new old_news();
$option=array('page'=>'10','headline'=>'red','topindex'='bule');
return = $object->index($option);
```
となります。

■Simpley　Skin実装

インラインスキン
<% skin(*module,block*) %><% /skin %>

インラインにて、呼び出したモジュール・ブロックの値を出力する宣言です。
<% block() %>タグにて先にブロックの実行を行ったあと
インラインスキン開始に<% skin() %>
終了に<% /skin %>を宣言します。

例.

```
<html><body>
<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>
<% skin(module,block) %>
skin moduleが実行～～<br>
<% val(string) %>
<!--
モジュールのarray('string'=>'value')がマップされます
-->
<% /skin %>

</body></html>
```

■Simpley　Skin実装

外部ファイルスキン
<% extskin(*module*,*block*,"*skin_file*") %>

外部ファイルをskinとして扱います。
<% block() %>タグにて先にブロックの実行を行ったあ都に宣言します。
*module*には実行したいモジュール名。
*block*にはそのモジュールのメソッド名。
skin_file をフルパス又は相対パスで記述します。
終了の閉じタグはありません。

外部skinには通常のインラインスキンの記述を行います。

例.

```
<html><body>
<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>
<% extskin(module,block,"./ext/skin_file.php") %>
</body></html>

----skin_file.phpの内容-------

<% skin(module,block) %>
skin moduleが実行～～<br />
<% /skin %>
```

■Simpley　Skin実装

出力タグ
<% val(values) %>

値 values を改行コードを
に置き換えて出力します。
値 values は実行したブロックの戻り値の配列のキーの値に相当します。
（　array('values'=>'中身')　）

<% skin() %>タグ中に記述します。

例.

```
<html><body>

<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>

<% skin(module,block) %>
        <% val(string) %>
<% /skin %>

</body></html>
```

■Simpley　Skin実装

出力タグ
<% rval(values) %>

値 values をそのまま出力する。
改行コードやキャリッジリターンコードもそのまま出力されます。

値 values は実行したブロックの戻り値の配列のキーの値に相当します。
（　array('values'=>'中身')　）

<% skin() %>タグ中に記述します。

例.

```
<html><body>

<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>

<% skin(module,block) %>
        <% rval(string) %>
<% /skin %>

</body></html>
```

■Simpley　Skin実装

出力タグ
<% xval(values) %>

値 values を改行コードを
に置き換えて,危険な記号やタグをHTMLエンティティーに変換して出力（xss対策）<>の様なタグ記号を <>等に変換して出力します。

値 values は実行したブロックの戻り値の配列のキーの値に相当します。
（　array('values'=>'中身')　）

<% skin() %>タグ中に記述します。

例.

```
<html><body>

<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>

<% skin(module,block) %>
        <% xval(string) %>
<% /skin %>

</body></html>
```

■Simpley　Skin実装

出力タグ
<% uval(values) %>

値 values をurlエンコード変換して出力します。

値 values は実行したブロックの戻り値の配列のキーの値に相当します。
（　array('values'=>'あいうえお')　）

<% skin() %>タグ中に記述します。

例.

```
<html><body>

<% block(module,block,PAGE=1&OFFSET=0&LIMIT=20) %>

<% skin(module,block) %>
<a href="form.php?japanese=<% uval(values) %>">日本語</a>
<% /skin %>

</body></html>
```

Skinファイル実行後

```
<a href="form.php?japanese=%82%CC%8Fo%97%CD%3Cbr%3E%0A%3Chr%3E">日本語</a>
```

■Simpley　Skin実装

出力タグ
<% kval(key/key2/value) %>

<% kval(key/key2/value) %>
は次のような配列の値を返します。
$out['key']['key2']['value']='うまー';

スラッシュ区切りで配列の深さを指定出来ます。
valやdef等のスラッシュ区切りとは意味合いが異なりますのでご注意ください。

改行コードやhtmlタグコードの変換は行いません.rval()と同様です。

<% skin() %>タグ中に記述します。

メモ.
<% val(key/key2/value) %>
はつぎの様な配列になっています.
$out['key']=array(0,array(0,'key2'=> array(0,'value'=>'うまー')));
$out['key'][0]['key2'][0]['value'][0]='うまー';

```
var_dump($out);
array(1) {
  ["key"]=>
  array(2) {
    [0]=>
    int(0)
    [1]=>
    array(2) {
      [0]=>
      int(0)
      ["key2"]=>
      array(2) {
        [0]=>
        int(0)
        ["value"]=>
        string(6) "うまー"
      }
    }
  }
}
```

■Simpley　Skin実装

繰り返しループタグ
<% each() %><% /each %> (<% loop() %><% /loop %>)

<% each(list) %>から<% /each %>までの間をループ処理する。

値 list が配列１個以上有る場合のみ有効です。
値 list は bolckを実行した配列（か連想配列）の戻り値である

<% skin() %>タグ中に記述します。

例.
$output['list']に次の２次元配列が複数入ってる場合

```
for($i=0;$i<100;$i++){
        $output['list'][$i]=array('tid'=>$tid , 'new'=>$new , 'nicname'=>$nicname , 'email'=>$email );
}
```

--SKINファイル-----

```
<% each(list) %>
<font color="#000000"><% val(list/km) %>km<br>
<% val(list/date) %><a href="main.html?tid=<% val(list/tid) %>">No.<% val(list/tid) %>City<% def(list/new) %></a>
 └<% xval(list/nicname) %><br>
<% def(list/email) %> └<% xval(list/email) %><br><% /def %>
</font>
<% /each %>
```

--実行後----------

```
<font color="#000000">100km<br>
20051022<a href="main.html?tid=10">No.10City新</a>
 └ニックネーム<br>
 └email@address<br>
</font>
```

■Simpley　Skin実装

繰り返しループタグ中で使える連番タグ
<% eachval() %>

<% each() %>タグ中に記述します。

例.

```
<% each(list) %>
  連番= <% eachval(list,1) %>
  ID  = <% val(list/id) %>
  VAL = <% val(list/val) %>
    <select name="select_box">
    <% each(list_s) %>
        <option value="<% val(list_s/id) %>" <% ifs(list_s/id,==,list/id) %>selected="selected"<% /ifs %>><% val(list_s/val)
%></option>
    <% /each %>
    </select>
   <br />
   <br />
<% /each %>

<!--
<% eachval(list,1) %>と書くと、１から順番にリストの終わりまで連番が振られる。
<% eachval(list) %>と書くと0から順番にリストの終わりまで連番が振られる。
-->
```

■Simpley　Skin実装

繰り返しタグ２
<% keach(list) %><% keval(KEY) %><% keval(VAL) %><% /keach %>

１次元配列の変数をループ処理します。
<% keval(KEY) %>は配列のキーを表示します。
<% keval(VAL) %>は配列のキーに格納されている値を表示します。

<% skin() %>タグ中に記述します。

例.
```
$out['list']=array('key1'=>'value','key2'=>'value2','key3'=>'value3' );

<% keach(list) %>
                <% keval(KEY) %> => <% keval(VAL) %><br />
<% /keach %>
```

■Simpley　Skin実装

条件指定
<% def(values) %><% /def %>

<% def(values) %>の条件の時<% /def %>までを処理する。

<% skin() %>タグ中に記述します。

例.
def()での条件とは valuesがarrayの時 要素が１個以上有る場合
valuesがstring 若しくはその他の数字等の場合は 空白スペースを含む１文字以上が有る場合

```
$output['values']=' ';      //条件にマッチする
$output['values']='';       //条件にマッチしない
$output['values']=array( array('toys'=>$toys , 'name'=>$name) );     //条件にマッチする
$output['values']=array();      //条件にマッチしない
```

■その他の条件指定
<% undef(values) %><% /undef %>
<% undef(values) %>の条件がではないとき<% /undef %>までの処理をする。
(<% def() %> タグの逆の動作)

<% anddef(values|values2) %><% /defand %>
<% anddef(values|values2) %> の values と values2の条件(if(values and values2))の時<% /defand %>までを処理する

<% ordef(values|values2) %><% /defor %>
<% ordef(values|values2) %> の values か values2の条件(if(values or values2))の時<% /defor %>までを処理する

<% defxor(values|values2) %><% /defxor %>
<% defxor(values|values2) %> の values と values2の xor 条件(if(values xor values2))の時<% /defxor %>までを処理する

値 bool は blockを実行した返値の変数の中身の有無である。def()の複数版と言ったところ。
中身は string 1文字以上(strlen($str)>0)か array で配列の個数(count($array)>0)が1個以上の場合もdef()と同様。

<% def() %>は条件を1つまでしか設定できないが、
<% defand() %><% defor() %><% defxor() %>は |(パイプ記号)でいくつでも条件を
追加することが出来る。

■Simpley　Skin実装

条件指定２

<% if(ターゲット,比較演算子,任意の文字列又は数字等) %>
<% elseif(ターゲット,比較演算子,任意の文字列又は数字等) %>
<% /else %>
<% /if %>

php のif文と同様である。 elseifは複数個タギングしてもかまわない。
if() 条件の一番最後には必ず <% /if %>をタギングする必要がある。
文字列比較の場合は必ずダブルクオートでくくってください。
数値比較、ブーリアン比較はそのままのタギングでかまいません。

<% skin() %>タグ中に記述します。

例.

```
<% if(value,==,"OFF") %>
                valueはOFFです<br>
<% elseif(value,==,"ON") %>
                valueはONです<br>
<% /else %>
                valueはONでもOFFでもないようです<br>
<% /if %>
```

例2.

```
<% if(value,==,1000) %>
                valueは1000です<br>
<% elseif(value,==,0) %>
                valueは0です<br>
<% elseif(value,>,20000) %>
                valueは20000より大きいようです<br>
<% elseif(value,<,20000) %>
                valueは20000より小さいようです<br>
<% /if %>
```

■Simpley　Skin実装

条件指定３

<% ifs(ターゲット１,比較演算子,ターゲット２) %>
<% elseifs(ターゲット１,比較演算子,ターゲット２) %>
<% /ifs %>

php のif文と同様である。 elseifs() は複数個タギングしてもかまわない。
通常の<% if() %> と違う所はSKIN中で利用される変数同士の比較が出来るところです。
閉じタグは<% /ifs %>で閉じてください。

<% skin() %>タグ中に記述します。

例.
```
//モジュール中
$list_s[]=array('id'=>1,'val'=>'number 1');
$list_s[]=array('id'=>2,'val'=>'number 2');
$list_s[]=array('id'=>3,'val'=>'number 3');
$list_s[]=array('id'=>4,'val'=>'number 4');

$list[]=array('id'=>1,'val'=>'number 1');
$list[]=array('id'=>2,'val'=>'number 2');
$list[]=array('id'=>3,'val'=>'number 3');
$list[]=array('id'=>4,'val'=>'number 4');

$out['list_s']=$list_s;
$out['list']=$list;

return $out;
```

例.
```
<% each(list) %>
  ID  = <% val(list/id) %>  VAL = <% val(list/val) %>
    <select name="select_box">
    <% each(list_s) %>
        <option value="<% val(list_s/id) %>" <% ifs(list/id,==,list_s/id)
%>selected="selected"<% /ifs %>><% val(list_s/val) %></option>
    <% /each %>
    </select>
   <br />
   <br />
<% /each %>
<!--
listのidとlist_sのidが同一の場合セレクトＢＯＸのselected="selected"が表示
される。
-->
```

■Simpley　Skin実装

拡張絵文字タグ
（サイトコンフィグレーションで絵文字変換を有効にする必要があります。）

<% emojival(values) %>
値valuesを 携帯絵文字に変換して出力する他は<% val() %>と同様

<% emojirval(values) %>
値valuesを携帯絵文字に変換して出力する他は値をそのまま出力する

<% emojixval(values) %>
値valuesを携帯絵文字に変換して出力する他は
危険な記号やタグをHTMLエンティティーに変換して出力（xss対策）する他は<% xval() %>と同様

<% emojiuval(values) %>
値valuesを携帯絵文字に変換して出力する他は<% uval() %>と同様

5.Simpleyコア

標準メソッド・関数

■Simpley　コア標準メソッド・関数

Simpleyのコア部分には、良く使われると思われる関数や、メソッドが備わっています。

コア内部にはそのほとんどがskin解析に使われるメソッドが多くそろっていますが、そうでない物も存在します。

ここでは、skin解析系ではないよく使われると思われるのを紹介します。

コアメソッドの利用方法は基本的には

```
$simpley = new Simpley;
```

した後にメソッドアクセスで行います。

■Simpley　コア標準メソッド・関数

Simpley::include_module()

$simpley->include_module(*modulename string*);

指定のSimpleyモジュール *modulename* をインクルード(include_once)する.
Simpleyモジュールはサイト設定で定義した、 /modules/以下になります。

他のモジュールのメソッドやコントローラをskinを介さず使いたいときに使います。

インクルードに失敗したらfalseを返します。成功時にはtrueを返します。

例.

```
if($simpley->include_module('modulename')){
	$object=new modulename;
}else{
	$message=$this->message['error_modules'];
}
```

■Simpley　コア標準メソッド・関数

Simpley::directblock_load()

$simpley->directblock_load(*modulename string* , *blockname string* , *param array*);

指定のSimpleyモジュール *modulename* の *blockname* の実行結果を返します。
スキンタグの blockタグとよく似ています。
違うところは *param* を直接配列で渡すところです。
$param=array('page'=>10 , 'color'=>'red');
の様に渡す必要があります。

他のモジュールのメソッドやコントローラをskinを介さず使いたいときに使います。

例.

```
例.
$param=array('page'=>'nolimit' , 'default_limit'=>30 );
$return = directblock_load('news','list', $param );
print_r($return);
```

■Simpley　コア標準メソッド・関数

Simpley::GetReadFile()

$simpley->GetReadFile(*path string [, mode]*);

指定の path のファイルを読み込んで返します。
mode は fopenの読み取りモードを指定します。デフォルトは rb 読み込みのみのバイナリモードで開きます。

ファイルを読み込むのに失敗したら falseを返します。

例.

```
$body=$simpley->GetReadFile('/path/to/file.txt');

if($body==false){
     $message=$this->message['error_read_file'];
}
```

■Simpley　コア標準メソッド・関数

Simpley::WriteFile()

$simpley->WriteFile(*path string* , body string *[, chmod [, mode]]*);

指定の path にファイルbodyを書き込みます。。
chmodはファイルのパーミッションを指定します。デフォルトは 0777 です。
mode は fopenの読み取りモードを指定します。デフォルトは wb 書込バイナリモードで開きます。

ファイルを書込するのに失敗したら falseを返します。

例.

```
if( !$simpley->WriteFile('/path/to/file.txt',$body) ){
     $message=$this->message['error_write_file'];
}
```

■Simpley　コア標準メソッド・関数

Simpley::ConvartEnc()

$simpley->ConvartEnc(*array array* , *mode int*);

連想配列を指定のモードで文字コードの変換を行います。
arrayは$_GET/$_POST等にも使えます。
Modeは
1 内部エンコードから外部エンコード
2 外部エンコードから内部エンコード
3 DBエンコードから内部エンコード
4 DBエンコードから外部エンコード
5 内部エンコードからDBエンコード
6 外部エンコードからDBエンコード
に変換します。各種コードはサイト設定を行った指定に依存します。指定の文字コードが同一の場合はそのまま値を返します。

例.

```
//PHPで自動文字変換を行わない場合等のために。。
$post = $simpley->ConvertEnc( $_POST , 2 );//外部文字コードから内部文字コードへ変換

$var = SelectRec($sql);    //DBから連想配列の結果セットを取得
$var=$simpley->ConvertEnc( $var , 3 ); //DBの文字コードからPHPの内部コードへ変換
```

■Simpley　コア標準メソッド・関数

Simpley::valconv()

$simpley->valconv(*str string* , *mode int*);

strを指定のモードで文字コードの変換を行います。

Modeは
1 内部エンコードから外部エンコード
2 外部エンコードから内部エンコード
3 DBエンコードから内部エンコード
4 DBエンコードから外部エンコード
5 内部エンコードからDBエンコード
6 外部エンコードからDBエンコード
に変換します。各種コードはサイト設定を行った指定に依存します。指定の文字コードが同一の場合はそのまま値を返します。

例.

```
//PHPで自動文字変換を行わない場合等のために。。
$name = $simpley->valconv( $_POST['name'] , 2 );//外部文字コードから内部文字コードへ変換

$var = SelectRow($sql);    //DBから配列の結果セットを取得
$name=$simpley->ConvertEnc( $var['name'] , 3 );//DBの文字コードからPHPの内部コードへ変換
```

■Simpley　コア標準メソッド・関数

Simpley::DBquote()

$simpley->DBquote(*str string*);

文字列をDBに合うようにエスケープする。

設定したDBのタイプに合わせて適切なエスケープをstrに行います。
MySQLに設定した場合は str は mysql_real_escape_string(str)の結果を返します。
Dbタイプが無い場合はaddslashesされます。
また、magic_quote_gpcがOnの場合は何もされずに文字列を返します。

例.

```
$name = $simpley->DBquote( $_POST['name'] );
$sql='insert into table (name) values(¥''.name.'¥') ';
```

■Simpley　コア標準メソッド・関数

Simpley::GetInternalEnc()

```
$simpley->GetInternalEnc();
```

現在の内部エンコードを取得します。EUC-JP/SJIS等
定数からも取得できますが推奨しません。

Simpley::GetOutputEnc()

```
$simpley->GetOutputEnc();
```

現在の出力エンコードを取得します。EUC-JP/SJIS等
定数からも取得できますが推奨しません。

Simpley::GetEmojiEnc()

```
$simpley->GetEmojiEnc();
```

現在絵文字モジュールが使われてるかどうかを取得します。
使われてる場合は frue 利用無しだと falseが返ります。

Simpley::GetQuoteSet()

```
$simpley->GetQuoteSet();
```

magic_quote_gpcが設定されているかどうかを取得します。
Onの時true Offの時false

■Simpley　コア標準メソッド・関数

Simpley::GetDBtype()

```
$simpley->GetDBtype();
```

現在設定されているデータベースエンジンを取得します。
Mysql / postgres / sqlite等が戻り値として返ります。
定数からも取得できますが推奨しません。

Simpley::GetCacheSet()

```
$simpley->GetOutputEnc();
```

skinをキャッシュする設定かどうかを返します。キャッシュ利用有りの場合は true 無い場合は falseを返します。

Simpley::GetCacheLifeTime()

```
$simpley->GetCacheLifeTime();
```

skinキャッシュの有効時間を取得します。単位は秒です。

Simpley::GetCacheDirSet()

```
$simpley->GetCacheDirSet();
```

キャッシュディレクトリの位置を返す

■Simpley　コア標準メソッド・関数

Simpley::GetMainDir()

```
$simpley->GetMainDir();
```

メインディレクトリのパスを返します。定数からも取得できますが推奨しません。

Simpley::GetDegubSet()

```
$simpley->GetDegubSet();
```

debugの設定の有無を返します。設定が有る場合 true 無い場合は falseを返します。

Simpley::GethttpInputSet()

```
$simpley->GethttpInputSet();
```

PHPによる文字コードの自動エンコード変換の設定を取得
Onの場合は true Offの場合は falseを返します。

■Simpley コア標準メソッド・関数(メール送信モジュール)

Simpleyのコア外ですが、メールを送信するモジュールも有りますので合わせて解説します。

```
$simpley->directblock_load( "mail" ,"mail_send" ,
array('subject'=>'SUBJECT','body'=>'BODY','to_mail'=>'to_mail_address','from_mail'=>'from_mail_address') );
```

directblock_load()メソッドで mail モジュールの mail_send ブロックを利用してメールを送信します。

３番目の引数に以下の配列でパラメータを渡すことによりメールを送信できます。

array('subject'=>サブジェクト,
'body'=>メール本文,
'to_mail'=>宛先メールアドレス,
'from_mail'=>差出人メールアドレス)

上記のようにパラメータを渡します。
文字コード等はメールモジュール内で自動判定し、JISコードとしてメールを送信します。

to_mail 及び from_mail を省略した場合はメールモジュールのデフォルト設定のアドレスが利用されます。

渡すパラメータの文字コードは内分文字コードで有る必要が有ります。

6.DBアクセス

■DBアクセス

この章ではデータベースアクセス関数について解説します。

Dbアクセスは、現在スタティックな関数により実装されています。

その理由は、PHPの標準関数の用に扱い、PHPとの一体感を出すためです。

また、オブジェクト操作となる実装では、コントローラ側の実装を増やさなければいけないためです。

DBを使う所で毎回オブジェクトを生成する new とか一切無しで利用するためです。
あくまで、手軽にDBへアクセス出来るようにしています。

現在の実装では１つのDBへのアクセスに限られていますが、今後のVerUPにより、複数のDB（DBエンジンや、DBインスタンス）を同時に操作出来るようにする予定です。

■DBアクセス

SelectRec(*sql*)

sql　セレクトを実行します。
結果が正しく返ってきたときは２次元配列で結果が返ってきます。
（以下の例は数字添え字を利用した場合ですが、SELECTで指定したカラム名がキーになった配列も同時に取得できています。）
sql実行エラーの場合は false が返ってきます。

例.

```
$output=array();
$sql = "SELECT ID,SUBJECT,BODY,DATE,LAST_UPDATE FROM NEWS ORDER BY DATE DESC LIMIT 0,10 ";
$value = SelectRec( $sql );
      //比較演算子は===でなければ結果セットが０件の場合もfalseと判定されてしまいます。
if($vales===false){
    $message="エラー¥n";
}else{
    foreach( $value as $line ){
        $output['list'][]=array(                          //SQLの取得したレコード順に
                                'ID'=>$line[0],           //ID
                                'SUBJECT'=>$line[1],      //SUBJECT
                                'BODY'=>$line[2],         //BODY
                                'DATE'=>$line[3],         //DATE
                                'LAST_UPDATE'=>$line[4]   //LAST_UPDATEが入ります。
                                );
    }
}

//カラム名をそのままスキンにパースしたい場合は
$output['list']=$value;
//の用に返せます。
//skinでは <% val(list/ID) %>の様に書けます
//リスト表示の場合は<% each() %>を利用してください。

$output['message']=$message;
return $output;
```

■DBアクセス

SelectRow(*sql*)

sql　セレクトを実行します。
結果が正しく返ってきたときは1次元配列で結果が返ってきます。
(以下の例はカラム名をキーに利用した場合ですが、SELECTで指定したカラム順に数字添え字がキーになった配列も同時に取得できています。)
sql実行エラーの場合は false が返ってきます。

例.

```
$output=array();
$sql = "SELECT ID,SUBJECT,BODY,DATE,LAST_UPDATE FROM NEWS ORDER BY DATE DESC LIMIT 0,1 ";
$value = SelectRec( $sql );
       //比較演算子は===でなければ結果セットが０件の場合もfalseと判定されてしまいます。
if($vales===false){
    $message="エラー¥n";
}else{
        $output['info']=array(                                  //SQLの取得したレコード順に
                              'ID'=>$vales['ID'],               //ID
                              'SUBJECT'=>$vales['SUBJECT'],     //SUBJECT
                              'BODY'=>$vales['BODY'],           //BODY
                              'DATE'=>$vales['DATE'],           //DATE
                              'LAST_UPDATE'=>$vales['LAST_UPDATE']    //LAST_UPDATEが入ります。
                              );
}

return $output;
```

■DBアクセス

RunSql(*sql*)

bool RunSql($sql)
sql アップデート・インサート・デリート（他テーブル操作等）を行います。
主に更新系SQL文を実行する場合にこの関数を利用します。
sql実行エラーの場合は false が返ってきます。

例.

```
$output=array();
$SUBJECT='php news';
$ID=12;
$sql="UPDATE NEWS SET SUBJECT='$SUBJECT' , LAST_UPDATE=NOW() WHERE ID=$ID";
if( RunSql($sql)==false ){
    $message="エラーってます。¥n";
}else{
    $message="更新しました。¥n";
}

$output['message']=$message;
return $output;
```

7.DBセッション

■DBセッション

この章では、DBを利用したセッションについて説明します。

通常PHPのセッション($_SESSION)は、特に何も設定していない場合は、ファイルにセッション内容が保存されますが、Simpleyの
DBセッションを利用すればファイルではなく、DBテーブルにセッション毎の情報を記録することができます。

利用する方法は至って簡単で、各種ＤＢに対応するＤＢスキーマを作成し、サイト設定ファイルにＤＢセッションを利用する様に
するだけです。

なぜ、ＤＢセッションなのか？
理由は色々ありますが、もっとも簡単に説明する方法として、webサーバのロードバランシングを行った場合にファイルベースの
セッションを利用すると、ロードバランサによって利用するwebサーバが自動的に振り分けられ、セッション情報を失うからです
（そのサーバのセッション情報をまたげないため）。NFS等の共有ディスクにセッションファイルを設置した場合はその限りでは
ありません。
NFS等の共有ディスクとセッションDBをどちらを使うかは、そのコストにより変わってくるでしょう。

■DBセッション

Simpleyの標準モジュールにsessionモジュールが入っていますので、それを使った例が下の通りです。

モジュール例.

```
/*
    テスト14メソッド
*/
function test14( $param ){

    if( !isset($_SESSION['count']) ){
        $_SESSION['count']=0;
    }

    $_SESSION['count']++;
    $out['count']=$_SESSION['count'];

    return $out;
}
```

例.Skin

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>Simpley</title>
</head>
<body>
Simpley.
<br>
<a href="index15.php">NextPage</a><br>
<a href="index13.php">PrevPage</a><br>
<br>
<br>
<a href="index14.phps">SKINソース</a><br>
<br>
セッション利用してみる<br>
<br>
<% block(session,session_start,) %>
<% block(example,test14,) %>
<% extskin(example,test14,'./ext_skin/header.php' %>
<br>
<a href="index14.php">カウントアップ</a><br>
<br>
<% skin(example,test14) %>
<% rval(count) %><br>
<% /skin %>

</body>
</html>
```

8.携帯電話絵文字変換

■携帯電話絵文字変換

この章では、携帯電話の絵文字変換について説明します。

Simpleyには、携帯電話の絵文字変換の機能が標準で備わっています。

機能として以下があります。
•内部文字コードやDB文字コードがSJISでなくても良い
•携帯電話３（ドコモ・ａｕ・Vodafone）キャリアに加えPHS（WILLCOM）系４キャリアの絵文字が扱える
•４キャリア相互に類似する絵文字に変換することが出来る。
•キャリアを意識せず絵文字をDB等に保持できる。
以上の機能を持っています。

これらを利用するには、環境を整える必要があります。
まず、magic_quote_gpsがOffの必要があります（絵文字がエスケープされる為）。
内部エンコードは任意。
外部エンコードはSJIS。
PHPによる自動エンコード変換はOff。
SKINはEUCでも良いですが、絵文字を埋め込む場合はSJISが必須。

それ以外には、モジュールの実装にも気をつけなければならない点があります。

それはユーザーが入力した絵文字を内部に取り込む際、専用関数（ emoji_input_conv('string') ）を利用して、外部文字コードを内部文字コードに変換する必要があります。
内部コードと外部コードが同一の場合も専用関数（ emoji_input_conv('string') ）を必ず利用してください。

通常のmb_convert系関数を利用すると、変換時絵文字が壊れてしまうからです。

またskinには通常の val() 等ではなく emoji_val等に置き換える必要があります。

上記特記事項以外は今まで解説してきた通りで利用できます。

9.その他参考資料

■その他参考資料

目次

- コントローラでskinを選択しない理由
- コマンドラインからSimpleyモジュールを利用する
- Apache以外のwebサーバでSimpleyを利用する
- 関連サイト
- 商標

■その他参考資料

コントローラーでskinを選択しない理由

PHPのテンプレートエンジンには様々な物が存在していますが、それらは通常コントローラでテンプレートを選択している物がほとんどです。

しかし、SimpleyではSkin内で実行したいブロックを選択したり条件で外部skinを使う様にしています。

これらの理由は、コントローラでskinを選択してしまった場合、プログラマではないデザイナのみで選択するskinを入れ替えるのが困難であったり、修正するのが難しくなったりするためです。

skin側で実行したいブロックを選択するようにしておけば、ほとんどの場合は、決まったデータの入出力がすでに判明している場合がほとんどですので、プログラマ以外の人が自由にskinを書いたり、遷移を変更することが出来ます。

また、プログラマもブロック毎に規則性のある入出力を決めておけば、遷移変更等でコントローラを修正することなく、skin変更等ができるので、バグの誘発を未然に防げるのではないかと思います。

この様に、プログラマとデザイナが、それぞれの作業を完全分業する事により、より高い効率を生み出せるはずです。

■その他参考資料

コマンドラインからSimpleyモジュールを利用する

コマンドラインからモジュールを利用するには、skinに該当するPHPファイルに block呼び出しの他各種出力タグを用いるのは、通常のskin実装と同じです。

注意しなければならないのは、.htaccess等にauto_prepend_file等を設定している場合、CLI版は無視しますので、skinの先頭行に
<?php include_once("/path/to/siteconf.php"); ?>
の様にサイト設定ファイルをインクルードする必要があります。

■その他参考資料

Apache以外のwebサーバでSimpleyを利用する

Apache以外のwebサーバには .htaccess等がありませんので、各種設定をphp.iniにする必要があります。

また、auto_prepend_file等の設定をphp.iniにした場合、全てのPHPでその設定が有効になってしまうので、全てskinの先頭行に
<?php include_once("/path/to/siteconf.php"); ?>
の様にサイト設定ファイルをインクルードする必要があります。

webサーバの設定につきましてはご利用のwebサーバのセットアップマニュアルをご参考の上設定頂ければと思います。

■その他参考資料

商標

本資料に登場する会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

■その他参考資料

関連サイト

PHP本家	http://www.php.net
PHPマニュアル	http://www.php.net/manual/ja/
PHP日本ユーザー会	http://www.php.gr.jp/
Pearとか	http://pear.php.net/
pearマニュアル	http://pear.php.net/manual/ja/
Apache	http://httpd.apache.org/
MySQL本家	http://www.mysql.com/
MySQL日本ユーザー会	http://www.mysql.gr.jp/
PostgreSQL	http://www.postgresql.org/
日本PostgreSQLユーザー会	http://www.postgresql.jp/
SQLite	http://www.sqlite.org/
OpenOffice.org日本ユーザー会	http://ja.openoffice.org/
Debian GUN/Linux	http://www.debian.org/
VineLinux	http://www.vinelinux.org/

Simpley 公式サイト http://simpley.87op.com
Simpley利用サイト http://simpley.87op.com/link.html

■MEMO