SOLA
System Consultant

PHPテンプレート
（Sola PHP Template on IBMi）
ご紹介資料

目次

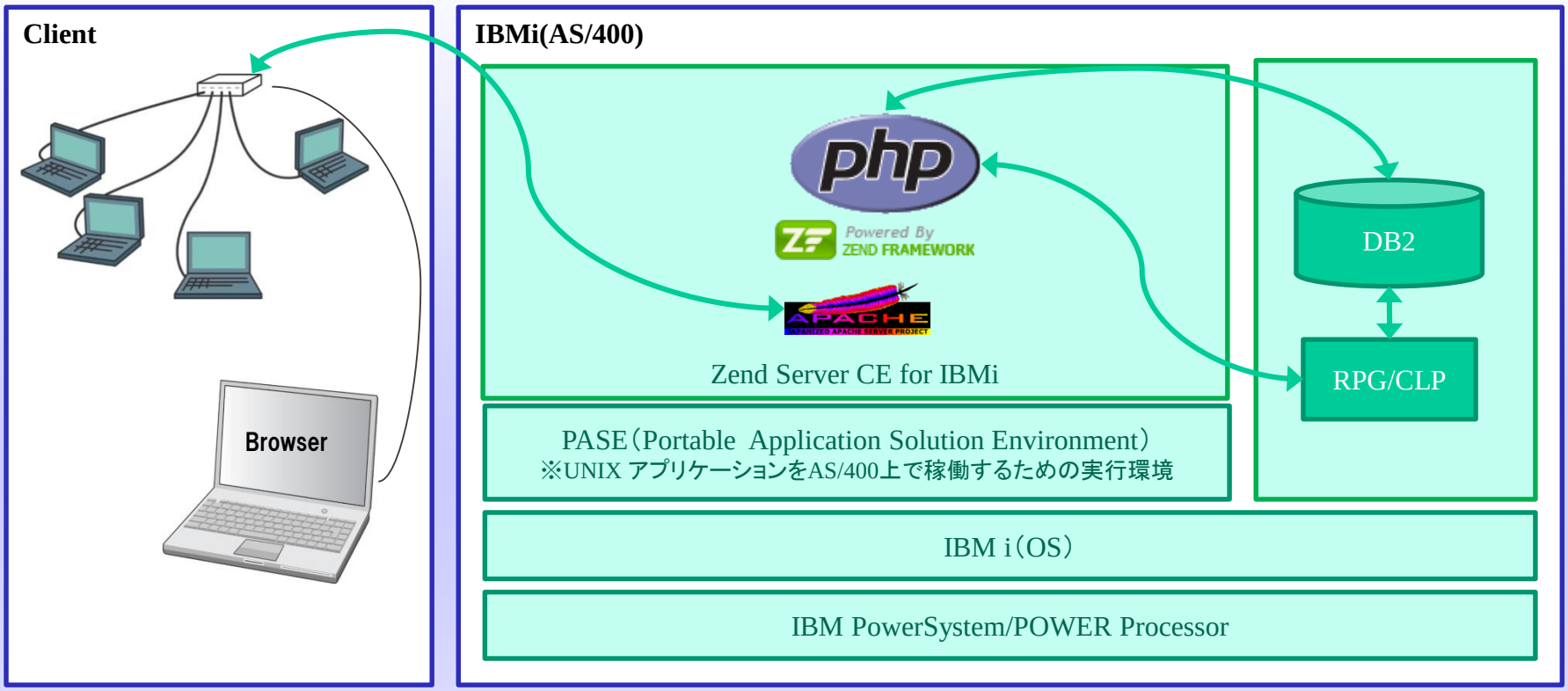
1．はじめに

モバイルや企業ポータルへの対応など、WEBアプリケーションへの取組は、IBMi（AS/400）ユーザーにとって大きな課題となっています

・IBMiのDB（DB/400）に対応したWEBアプリケーションを早期に構築したい
・広く使われている言語を使用して開発したい
・コスト面からオープンソースで開発をおこないたい
・自社でメンテナンスをおこなえるようにしたい

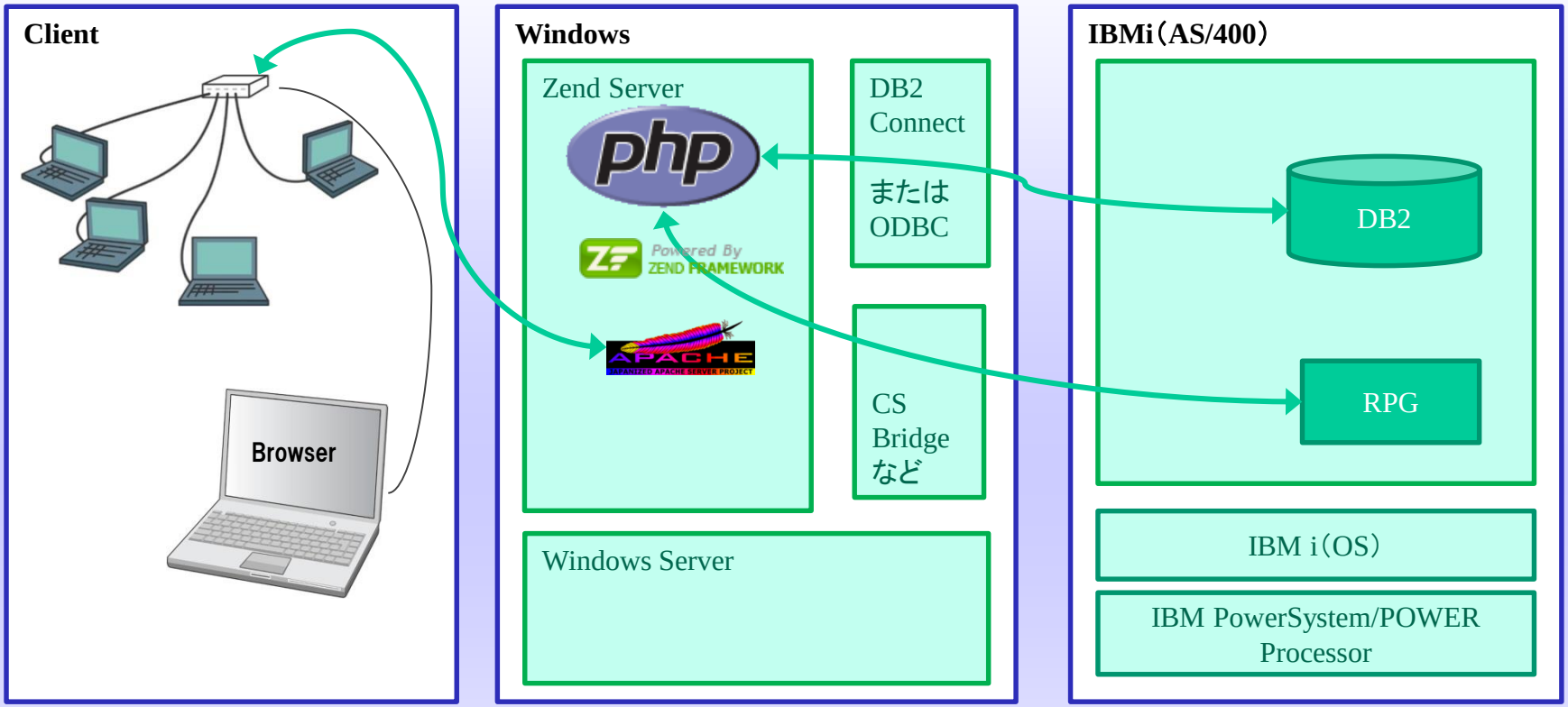
PHPテンプレート（Sola PHP Template on IBMi）をご検討ください

2. PHP稼働環境(1/2)
【IBMiでPHP稼働環境を構築する場合】



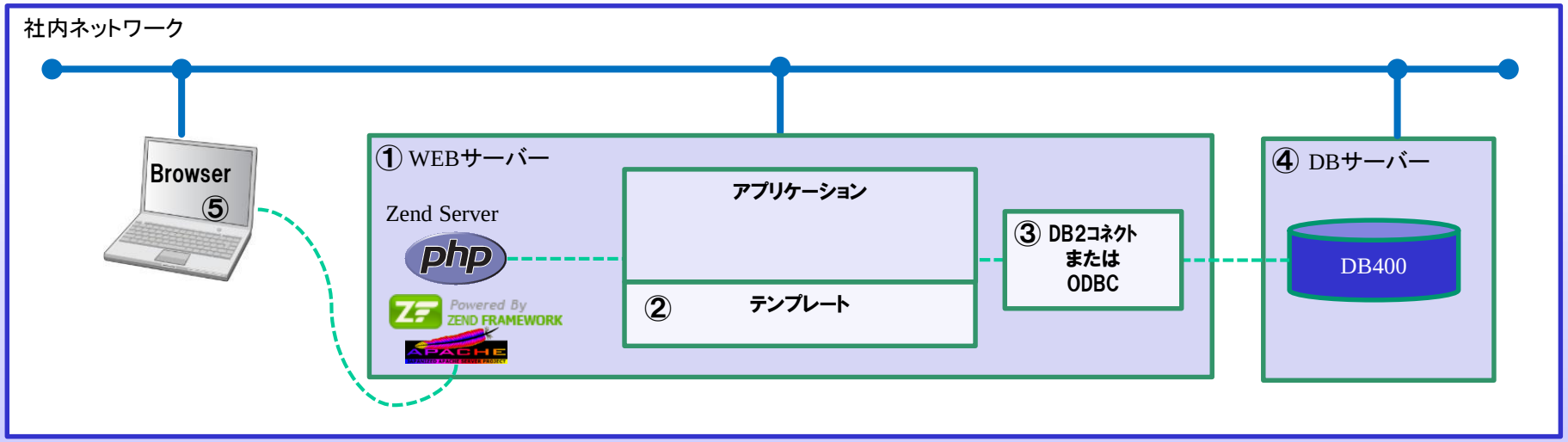
IBMi(AS/400)にZend Server CE for IBMi を導入します。
IBMi V7.1、V6.1、V5.4で無償提供されます。Zend.comからダウンロード可能です。
WEB ServerであるApacheとPHP実行環境およびZend FrameworkがPASEのもとで利用できます。
Zend Server Community Edition for i に含まれる「PHPツールキット」により、PHPアプリケーションから以下の方法でIBM i オブジェクトにアクセス可能となります。
・プログラムCALL(PHPからRPGをコール)
・ネイティブアクセス(DB2ドライバによりDB2/400をアクセス)など

2. PHP稼働環境(2/2)
【Windows でPHP稼働環境を構築する場合】



Zend Serverを導入します。Zendよりダウンロードし入手します。
WEB ServerであるApacheとPHP実行環境およびZend Frameworkが利用できます。
DB2との接続はDB2コネクトあるいはODBCを利用します。
IBMi(AS/400)のRPG起動はCS Bridge(Socket通信による電文の送受信を使用した通信ミドルウェア:日本IBM提供(有償))などを使用すれば可能です。

3. Sola PHP テンプレート対応環境



Sola PHP テンプレートは前項の環境に柔軟に対応しています。

①WEBサーバー
WEBサーバーはIBMiまたはWindowsで構築可能です。

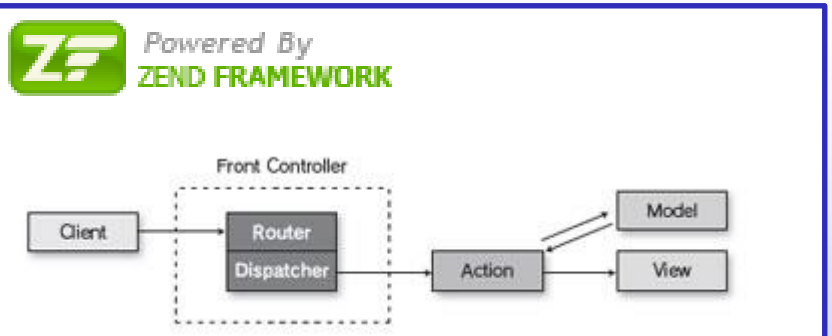
②テンプレート
豊富なテンプレートを利用することによって、早期にアプリケーションシステムを構築することが可能です。

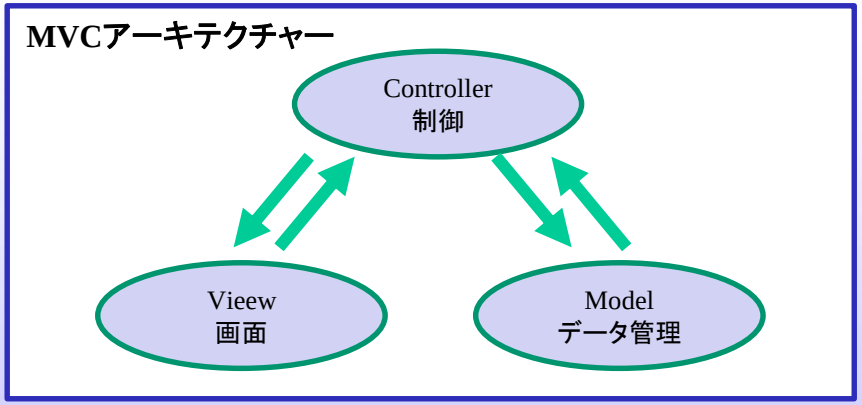
③DBインタフェース
WEBサーバーがWindowsサーバー、DBサーバーがIBMiの場合、DB400とのインタフェースとしてDB2コネクトまたはODBCを使用します。

④DBサーバー
DBサーバーはIBMiのDB400を前提としており、既存のアプリケーションへの連動が容易におこなえます。
（WindowsサーバーのMySQLでDBサーバーを構成することも可能です。）

⑤ブラウザ
WEBアプリケーションで構築されているため、処理はすべてブラウザで行います。

4. Sola PHP テンプレートの特長(1/2)





PHPは非常に柔軟な使い方ができるように作られているため、手軽に作成できてしまいます。そのため全体をしっかりと設計することを重視しないで作り始めてしまう場合があります。このために「PHPは大規模開発に向かない」という評価がされたりします。
これを解消するためにプログラム全体を構造的に作る「枠組み」を提供するのがフレームワークです。

Zend Frameworkは、MVCアーキテクチャを構成し、各種コンポーネントを組み合わせることで効率的なPHPアプリケーションを構築できるフレームワークです。
Zend Frameworkでは、Front ControllerパターンのMVCモデルが採用されています。

この方式では、フロントコントローラーへアクセスを集中させた後、要求されたアドレスなどを解析し、どこに何の要求が送られているのかを調べて、必要な処理を呼出します。

弊社で開発されたテンプレートはZendFrameworkを使用し、MVCアーキテクチャーにもとづいて作成されており、このテンプレートを使用することにより、MVCアーキテクチャーにもとづいたアプリケーションを開発することが可能です。

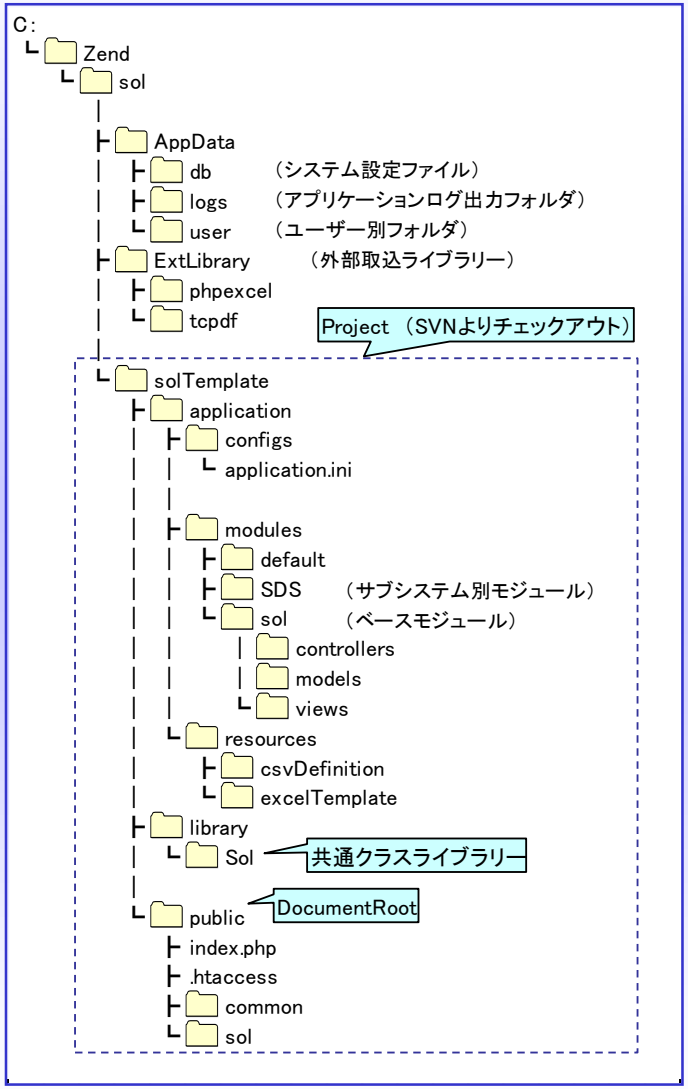
MVCの概念に準拠し、プログラムをController、View、Modelに分割して作成します。フォルダもそれぞれに分割して格納します。

ViewではPHPタグを埋め込んだ画面HTMLを作成します。
ModelではDBアクセスや計算などのビジネスロジックを作成します。
Controllerでは画面とビジネスロジックの実行制御を作成します。

Viewとビジネスロジックを分割することによって画面のデザインをデザイナーにゆだねることが可能になります。

4. Sola PHP テンプレートの特長(2/2)

プログラムディレクトリ例

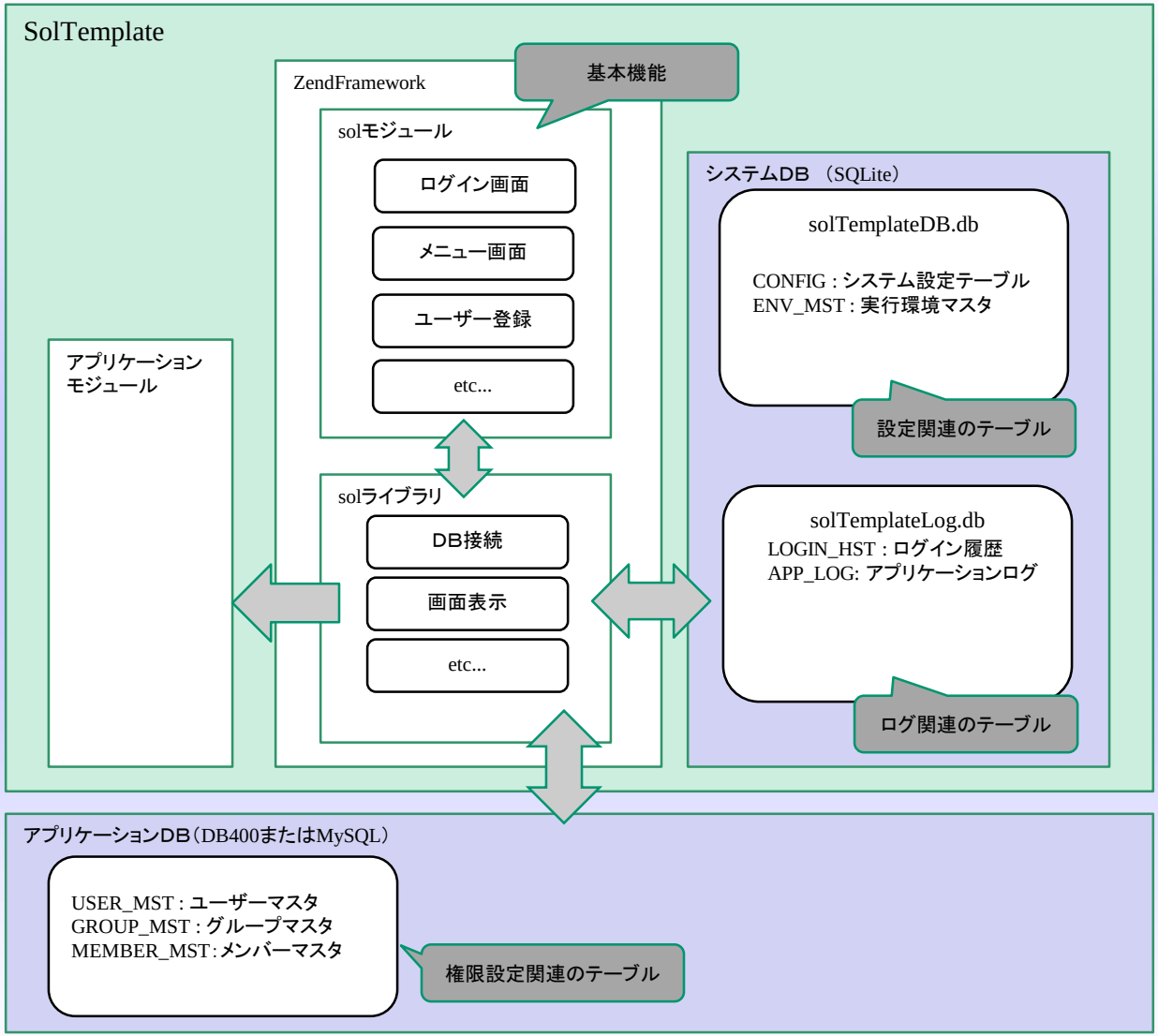


アプリケーション開発においてある特定の機能を持ったプログラムを部品化し、クラスとして継承できるようにしておくことが開発の効率を向上させます。これはPHPでの開発においても同様です。

弊社では下記のようなクラスをライブラリーに蓄積しております。
・ログイン制御
・メニュー制御
・権限制御
・DB接続制御
　－DB2/400-DB2コネクト
　－DB2/400-ODBC
　－MySQL
　－SQLite
・リスト形式の一覧表示、更新
・フラット形式の表示、更新
・ポップアップ検索
・PDF出力
・CSV出力
・EXCEL出力
・メール送信
など

弊社のテンプレート(共通クラス群)を使用して、システムの追加・拡張が容易に実現できます。

5. Sola PHP テンプレートの概要

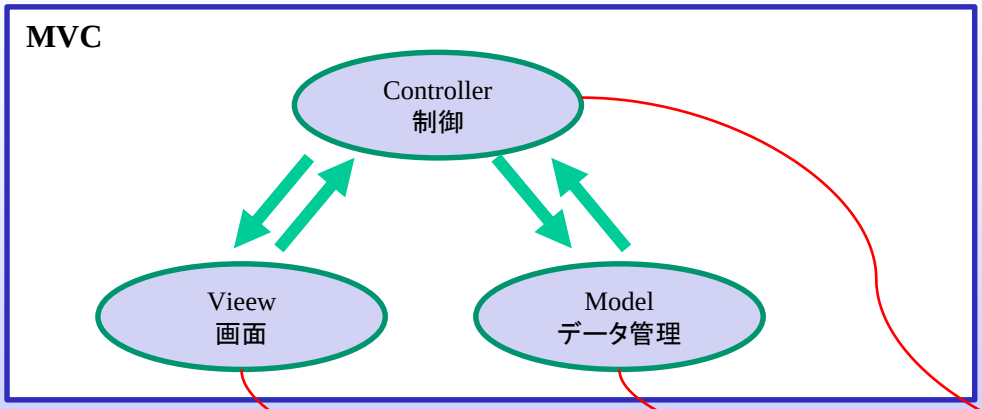


Solモジュール機能（システムDB）	
ログイン	ユーザー、パスワードにより認証を行う
メニュー	ログインユーザーの権限によりメニュータブの利用制限が可能
環境設定	本番環境、テスト環境、開発環境等の実行環境を登録し、それぞれのDB接続設定を行う
ログイン履歴	ログインしたユーザーの情報を照会する
ｱﾌﾟﾘｹｰｼｮﾝﾛｸﾞ照会	アプリケーションログの照会を行う

Solライブラリー機能
DBアクセス機能
一覧照会画面のテンプレート （改ページ機能）
更新画面のテンプレート
テンプレートを元にしたExcel出力機能、
PDF出力機能
メール送信機能
モバイル画面対応

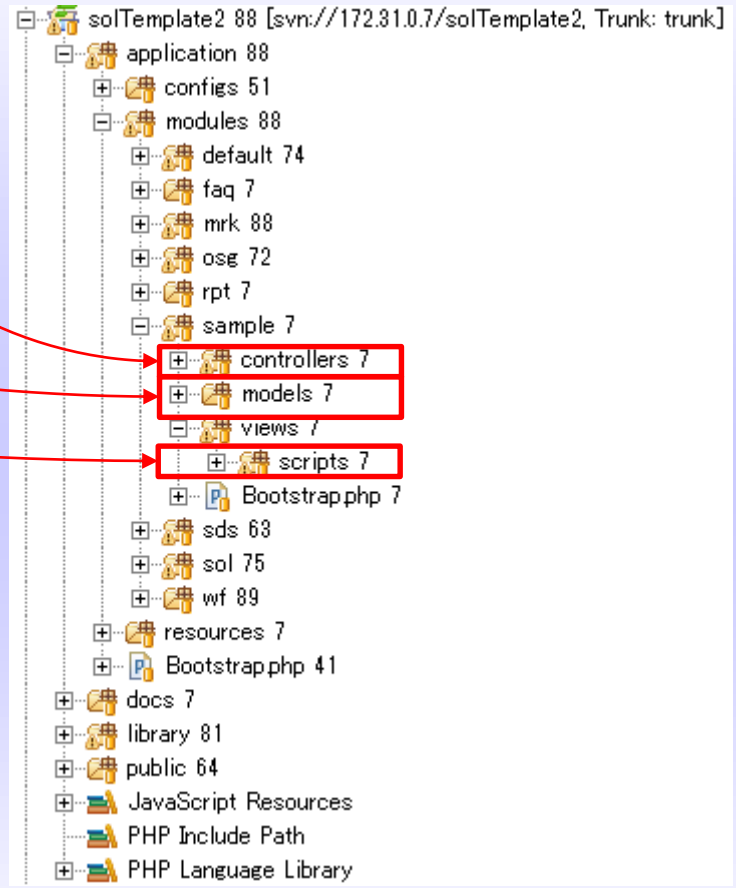
Solモジュール機能（アプリケーションDB）	
ユーザー設定	ユーザーの登録・訂正・削除を行う
グループ設定	ユーザーの属するグループ毎に実行環境、メニュー、プログラムレベルでの権限設定を行う

6. Sola PHP テンプレートの使用例（1/6）



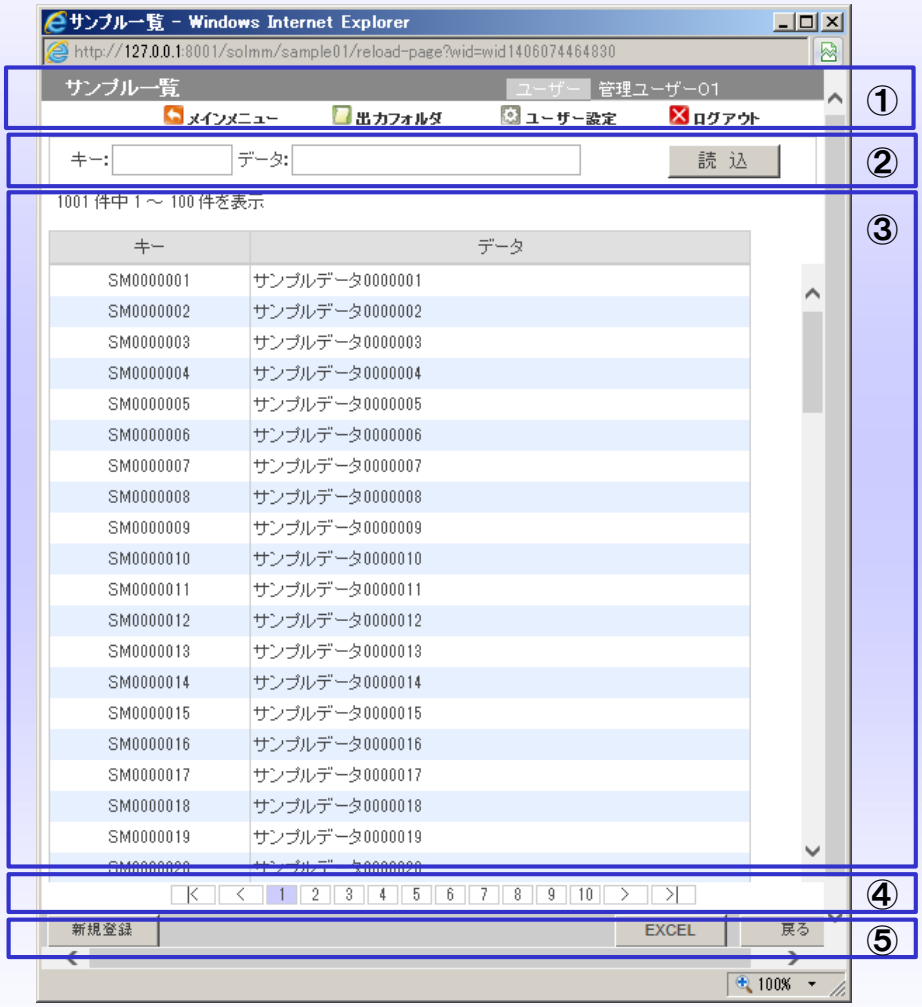
Zend Studioでアプリケーションフォルダを作成したら、そのフォルダ内に下記の3フォルダを準備します。
・controllers
・models
・views→scripts
作成しようとするプログラムに該当するパターンのサンプルプログラムをそれぞれにコピーして修正していきます。
・controllers←xxxxxController.php
・models←xxxxxModel.php
・views←xxxxx.phtml

Zend Studio で表示されるプログラム構成



6. Sola PHP テンプレートの使用例（2/6）

1－1. リスト形式の一覧テンプレートの例



①見出域
すべての画面で共通に設定できます。
メインメニュー：メニュー画面に戻ります。
出力フォルダ：該当ユーザーが出力したEXCEL、CSV、PDFなどが一覧表示され、選択することで起動できます。
ユーザー設定：ユーザーごとの設定する内容（一覧表示の1画面あたりの出力行数、画面サイズ、パスワード）の変更を行います。
ログアウト：ログアウトします。

②検索条件設定域
一覧表示データの検索条件を設定します。

③一覧表示域
検索条件に該当するデータを一覧表示します。
上部に該当データ件数と、現在表示している件数を表示します。

④ページナビゲータ域
ページナビゲータを表示します。
全件を1画面に表示することも可能です。

⑤アクションボタン域
アクションボタンを表示します。
左からF01～F12のファンクションキーが対応しており、ボタンクリックでもファンクションキークリックでも起動します。

一覧形式（伝票形式）の入力プログラムも、このリスト形式のテンプレートを利用して作成します。

6. Sola PHP テンプレートの使用例（3/6）

1－2. リスト形式のコントローラコーディング例　（ユーザーコーディングを行う箇所を　　で示しています。）

```php
<?php
// サンプル一覧
class Solmm_SAMPLE01Controller extends Sol_ListController{
    const PAGE_TITLE    = 'サンプル一覧';       // このページのタイトル
    const ACL_RESOURCE  = Sol_Acl::RS_USER;     // このページの権限レベル

    public function indexAction(){
        $view = new Sol_View();
        // タイトル
        $view->pageTitle = self::getPageTitle();
        // 入力項目セット（Modelで定義されている表示項目をViewにセットする）
        $this->setView($view);
        // 一覧表示部分セット
        $view->list         = $this->Model->List;                   // 一覧データ
        $view->recCounter   = $this->Model->recCounterHtml();       // 現在行表示
        $view->pageNavi     = $this->Model->pageNaviHtml();         // 改ページバー
        // 表示するphtmlファイルを指定
        $view->show('sample01.phtml');
    }

    // Excelダウンロード
    public function excelDownloadAction(){
        $this->excelTemplateName = 'sample01';
        return parent::excelDownloadAction();
    }

    public static function getPageTitle(){return self::PAGE_TITLE;}
    public static function getAclResource(){return self::ACL_RESOURCE;}
}
```

サンプルのリスト用コントローラからコピーしたコントローラで修正する箇所は以下の部分のみです。

・クラス名：フォルダ名とプログラム名を指定してクラスを定義します。リスト形式の場合は、solフォルダのListControllerを継承します。
・タイトル表記：画面見出域のタイトルを指定します。
・HTMLの指定：表示するphtmlファイルを指定します。
・EXCELダウンロードの指定：EXCELダウンロードを指定している場合、該当のEXCELテンプレートを指定します。

一覧形式（伝票形式）の入力プログラムを作成する場合、ユーザーチェックなどのロジックをこのコントローラ内にファンクションとして追加します。

6. Sola PHP テンプレートの使用例（4/6）

1－3. リスト形式のモデルコーディング例　（ユーザーコーディングを行う箇所を　　で示しています。）

```php
<?php
// サンプル一覧

class Solmm_Model_SAMPLE01Model extends Sol_ListModelDB{

    public function __construct($params){
        parent::__construct($params);
        // 検索条件定義
        $this->defCnd('SMPL_KY', 'キー', 10, '', null);
        $this->defCnd('SMPL_NM', 'データ', 50, '', null);
    }

    public function getSqlStmt(){
        $first = true;
        $sql=   'SELECT'
            .       ' S.*'
            .   ' FROM '.$this->getTable('M_SAMPLE').' AS S';
        $sql.=  $this->whereStmt('S.SMPL_KY', 'LIKE', $this->getFldVal('SMPL_KY').'%', null, $first);
        $sql.=  $this->whereStmt('SMPL_NM', 'LIKE', '%'.$this->getFldVal('SMPL_NM').'%', null, $first);
        $sql.=  ' ORDER BY S.SMPL_KY';
        return $sql;
    }
}
```

サンプルのリスト用モデルからコピーしたモデルで修正する箇所は以下の部分です。

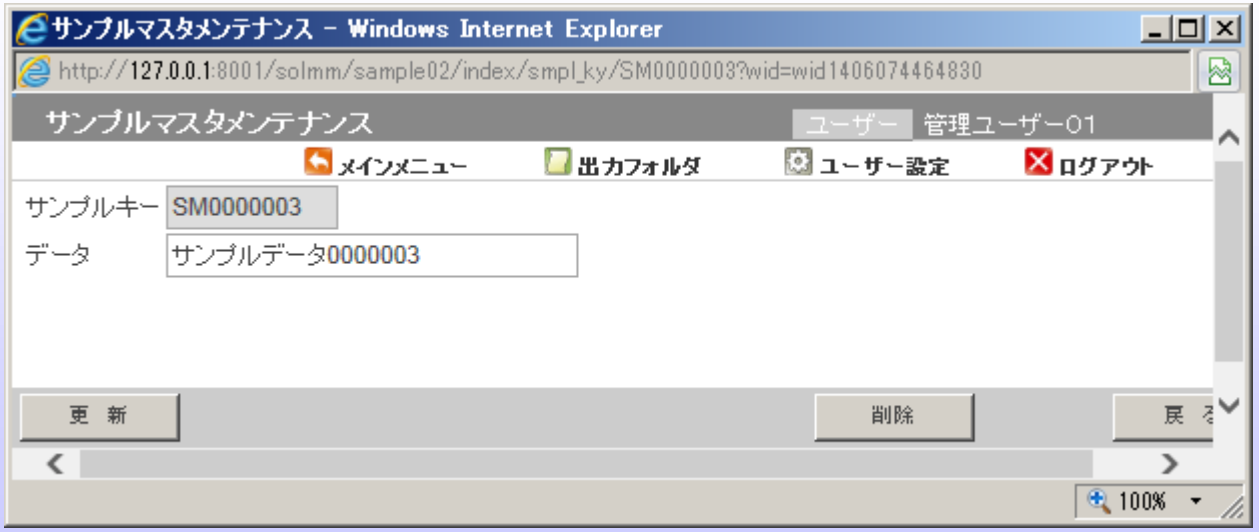
・クラス名：フォルダ名とプログラム名を指定してクラスを定義します。リスト形式の場合は、solフォルダのListModelDBを継承します。
・検索条件：検索条件を指定する項目を定義異します。
・SQL：レコード選択するSQLを定義します。

SQL定義では、WHEREステートメントを簡略表記することができます。

読込行ごとに処理を行いたい場合には、adjustRowというファンクションが用意されていますので、このモデル内にファンクションを追加します。
読込リスト全体で処理を行いたい場合には、adjustListというファンクションが用意されていますので、このモデル内にファンクションを追加します。

6. Sola PHP テンプレートの使用例（5/6）

2－1. フラット形式のテンプレートの例




```php
<?php // サンプルマスター

class Solmm_SAMPLE02Controller extends Sol_PageController{
    const PAGE_TITLE    = 'サンプルマスタメンテナンス';        // このページのタイトル
    const ACL_RESOURCE  = Sol_Acl::RS_USER;                  // このページの権限レベル

    public function indexAction(){
        $view = new Sol_View();
        // タイトル
        $view->pageTitle = self::getPageTitle();
        // 入力項目
        $this->setView($view);
        // 表示するphtmlファイルを指定
        $view->show('sample02.phtml', 'Dialog');
    }

    public static function getAclResource(){return self::ACL_RESOURCE;}
    public static function getPageTitle(){return self::PAGE_TITLE;}
}
```

2－2. フラット形式のモデルコーディング例　（ユーザーコーディングを行う箇所を　　で示しています。）

サンプルのフラットメンテ用コントローラからコピーしたコントローラで修正する箇所はリスト形式の場合とほとんど同様です。
継承するクラスはsolフォルダのPageControllerとなります。

ユーザーチェックを付加する場合は、userCheckというファンクションが用意されていますので、このコントローラ内にファンクションを追加します。

6. Sola PHP テンプレートの使用例（6/6）

2－3. フラット形式のモデルコーディング例　（ユーザーコーディングを行う箇所を　　で示しています。）

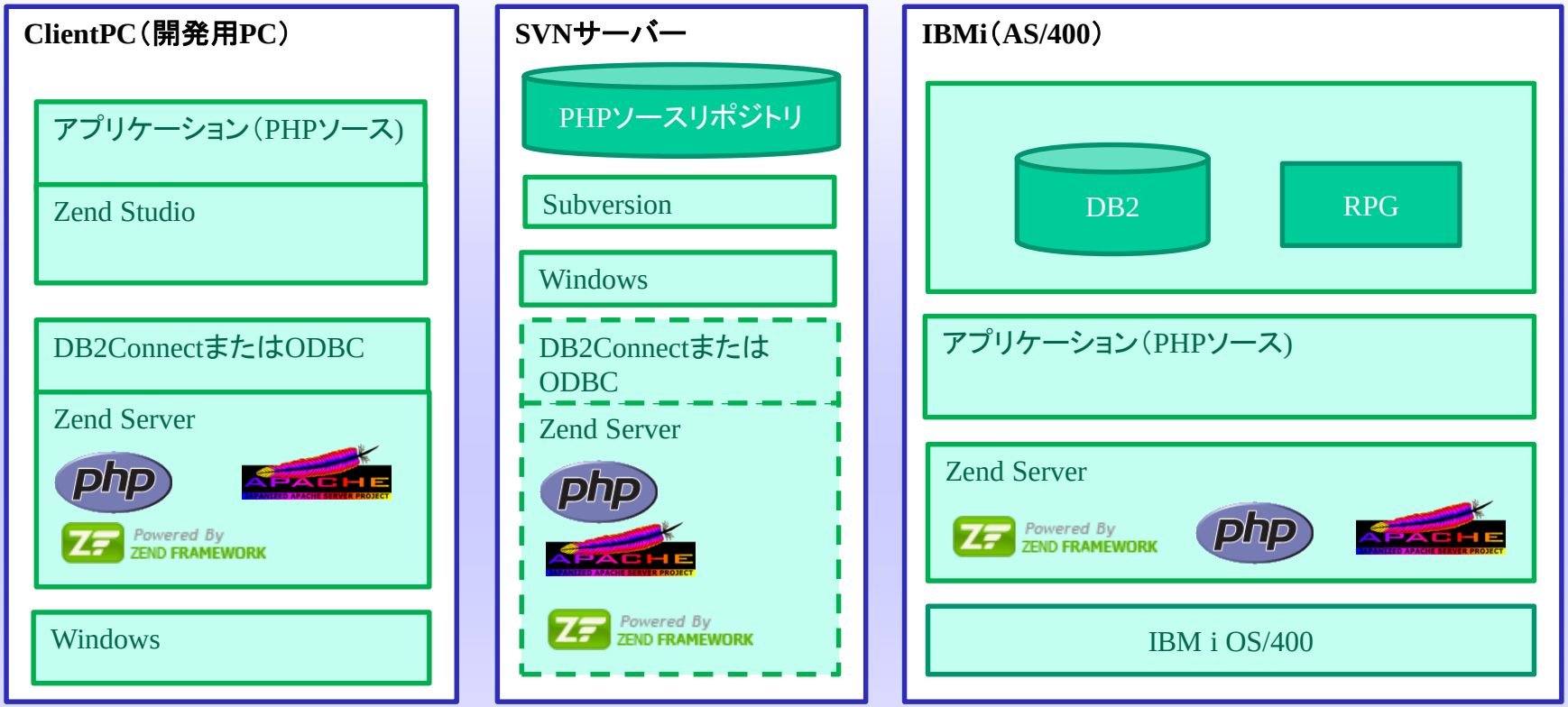
```php
<?php // ユーザーマスタメンテナンス 〔ページ更新画面〕

class Solmm_Model_SAMPLE02Model extends Sol_PageModel{

    public function __construct($params){
        $logger = Zend_Registry::get('logger');

        $this->dbAccess = Sol_AccessDB::getInstance()->setTable('M_SAMPLE');
        $this->keys = array('SMPL_KY'=>''); // キー項目定義

        parent::__construct($params);

        // 入力項目定義
        $this->defInp('SMPL_KY', 'キー', 10, '', array('NN','SB'));
        $this->defInp('SMPL_NM', 'データ', 50, '', array('NN'));

        // データ取得
        if (isset($params['smpl_ky']) and $params['mode']!='add'){
            $this->keys['SMPL_KY'] = $params['smpl_ky'];
            $this->getData();
        }
    }
}
```

サンプルのリスト用モデルからコピーしたモデルで修正する箇所は以下の部分です。

・クラス名：フォルダ名とプログラム名を指定してクラスを定義します。リスト形式の場合は、solフォルダのPageModelを継承します。
・テーブル定義：メンテナンス対象となるテーブル名を定義します。
・キー定義：メンテナンス対象となるテーブルのキーを定義します。
・入力項目定義：入力項目を定義します。
この定義分で基本的な項目チェックを行うことができます。
array () の指定で下記のチェックがなされます。
‘NN’：必須チェック
‘SB’：シングルバイトチェック
‘NM’：ニューメリックチェック
‘DT’：日付チェック
また日付の’ /’ を除去するなどのフィルターの指定もここで行います。
・データ取得：データ取得のためのキーセットを行います。

PageModelではDB更新を自動的に行います。更新処理に機能を追加する場合は、ファンクション「insert」「update」「delete」をそれぞれ定義し、追加処理を記述します。
PageModelには以下の追加ファンクションが用意されていますので、必要な処理があればこのモデル内にファンクションを追加します。
・preGetData：データ取得前に起動する。
・postGetData：データ取得後に起動する。
・

7. 開発環境



開発用のクライアントにZendStudioを導入しPHPプログラムを開発します。
クライアントをWebサーバーとしてIBMiに接続してテストする場合は、クライアントにZendServerとIBMi接続のためのミドルウェア（DB2ConnectまたはODBC）を導入します。
ソース管理のSVNをWindowsに導入します。（PCまたはPCサーバー）
SVNはIBMi用もありますが、古いバージョンとなり、またVisualSVNも存在しません。
SVNサーバーをWebサーバーとして機能させる場合は、SVNサーバーにもZendServerおよびIBMiミドルウェアを導入します。