

軽量MVCコントローラ PokoX

概要とサンプル

2005/10/22
風晶(Kaze Akira)
feng-jing-gsyc-2s@glamenv-septzen.net

お話の流れ

- ・ PokoXの開発目的・ポリシー・ライセンスなど
- ・ PokoXの仕組みのご紹介
- ・ PokoXのコントローラとしての機能紹介
- ・ PokoXのサンプルコードの紹介

PokoXの開発概要

【開発期間】

- ・2005年2月位から。（およそ）

【ライセンス】

- ・LGPL

【CVSリポジトリ】

- ・SourceForge.jp

MVCコントローラPokoXの目的・スタンス

難しくないこと

融通が利くこと

凝らないこと

長所：コントローラとして原始的で自由度が高く、
カスタマイズの余地が残されている。

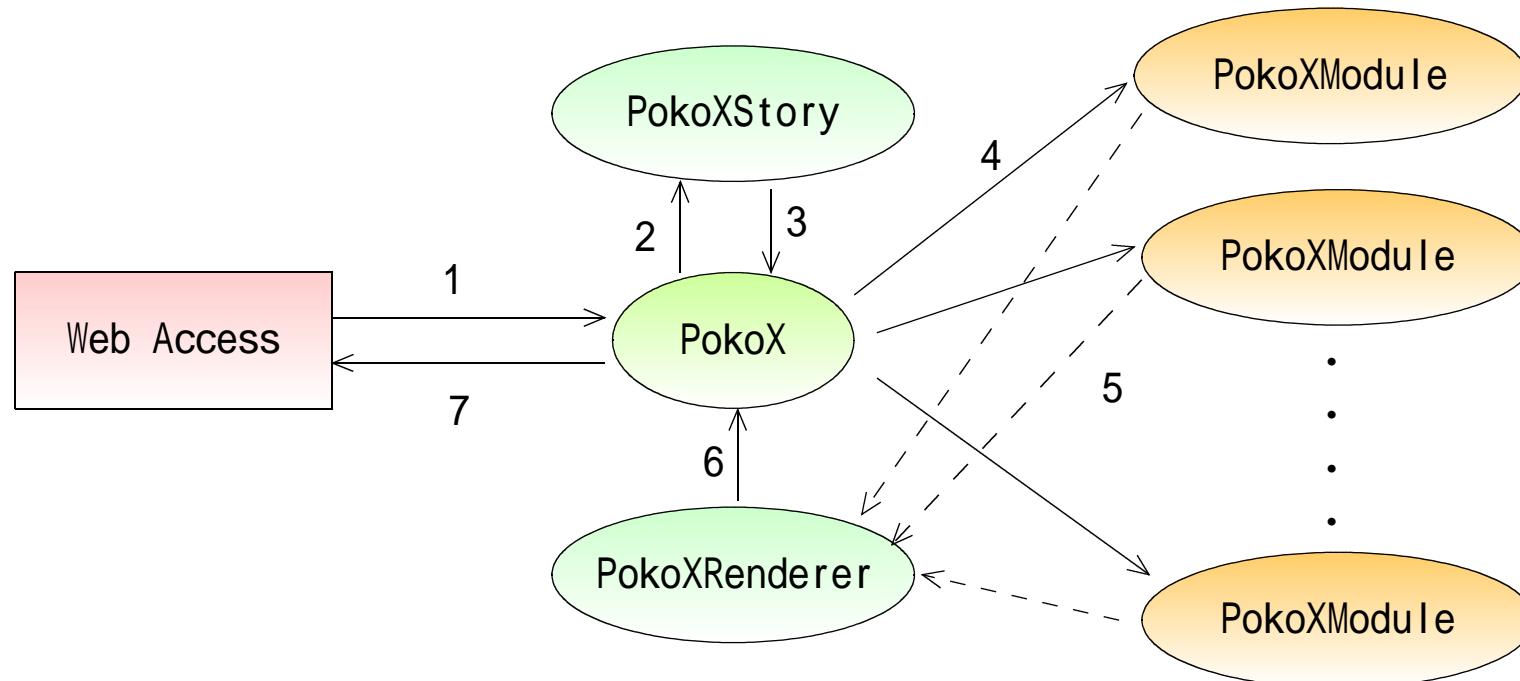
=

短所：Validation, DB-I/O, 二重POST防止, ログ出力など、
開発者がそれぞれに実装する必要がある。

重要：フレームワークではなく、コントローラに徹する。

PokoXの基本

- ・ 基本的なMVCモデルに基づく。
- ・ 「Access Control Identifier(ACI)」：PokoXが内部的に用いるアクセス権限文字列
- ・ 「ストーリー(story)」：モジュールやテンプレートを決定するための\$_GET/\$_POSTパラメータ



1. クライアントからのリクエスト
2. セッション・認証処理要求
3. ACI, STORY の決定
4. モジュールチェインの起動
5. 処理結果をレンダラ(View)にセット
6. レンダラの処理結果(HTML文字列)取得
7. クライアントに処理結果を出力

PokoXを制御する主な変数・定数

【global変数】

- **\$POKOX_MODULE_STORY**
"ACI.STORY"形式の文字列をキーとしてモジュールチェインを定義する連想配列
- **\$POKOX_TEMPLATE_STORY**
"ACI.STORY"形式の文字列をキーとしてレンダラの使用するテンプレートリソースを定義する連想配列
- **\$POKOX_MODULE_AUTOLOAD_PATH**
モジュールクラスの自動ロード対象PATH(配列による複数指定も可能)

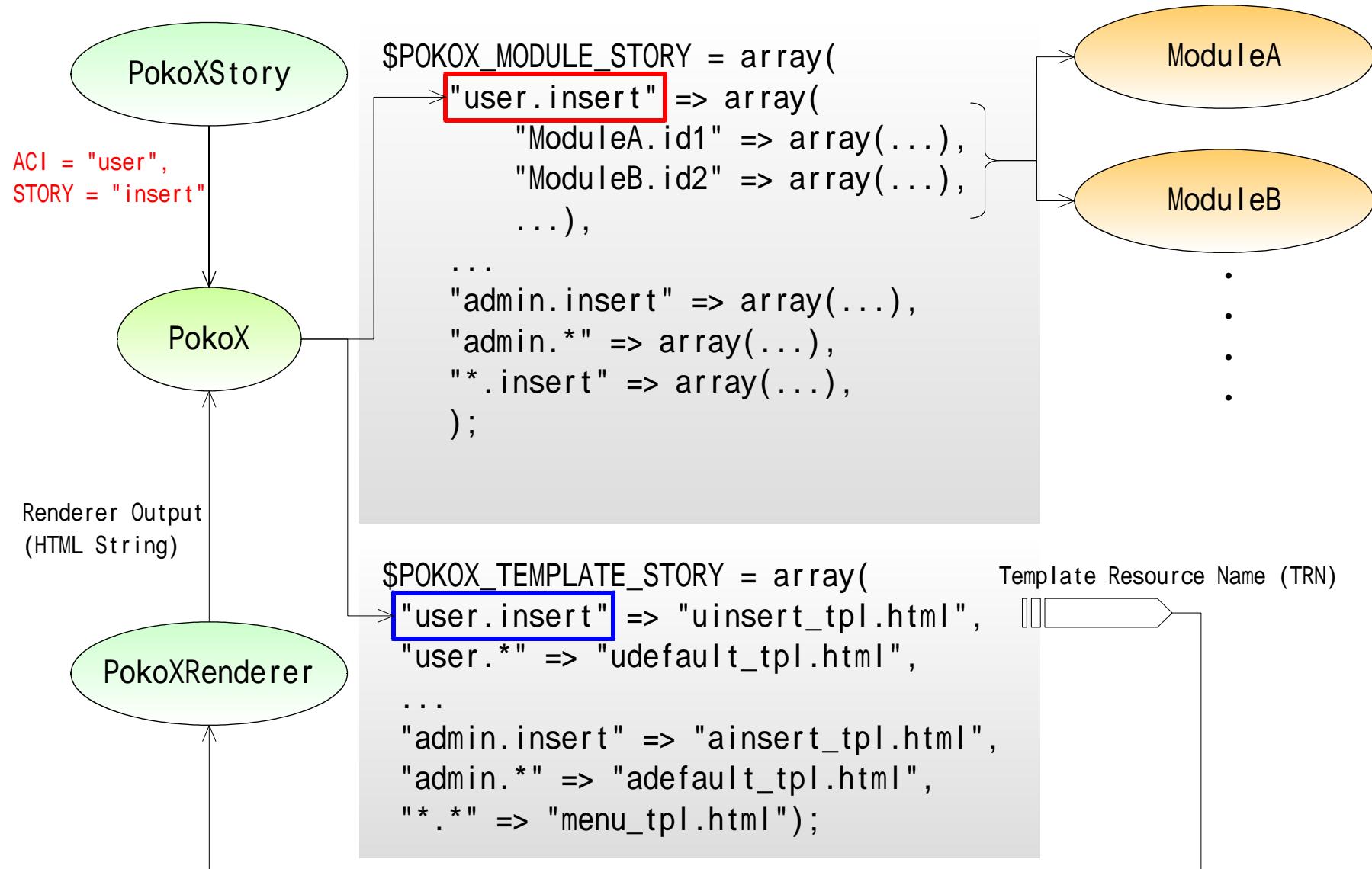
【定数】

- **POKOX_STORY_KEY**
Story取得に使う、\$_GET/\$_POSTのキー名。デフォルトは'pokox_story'なので、
http://....../index.php?pokox_story=XXX
とくれば'XXX'がStoryとして取得できる。
- **POKOX_ACI_KEY**
\$_POKOXからACIを取り出すためのキー名。

【特殊なglobal変数】

- **\$_POKOX**
StoryやACIなど、PokoXが用いる値が格納される連想配列。PokoXStoryクラスのstart()
メソッド内で設定される変数で、セッションを使用するStoryクラスの場合はセッションから
取得される。上記定数のキー名以外は、基本的にアプリ側で自由に利用できる。

ACIとStoryとモジュールチェインの関係



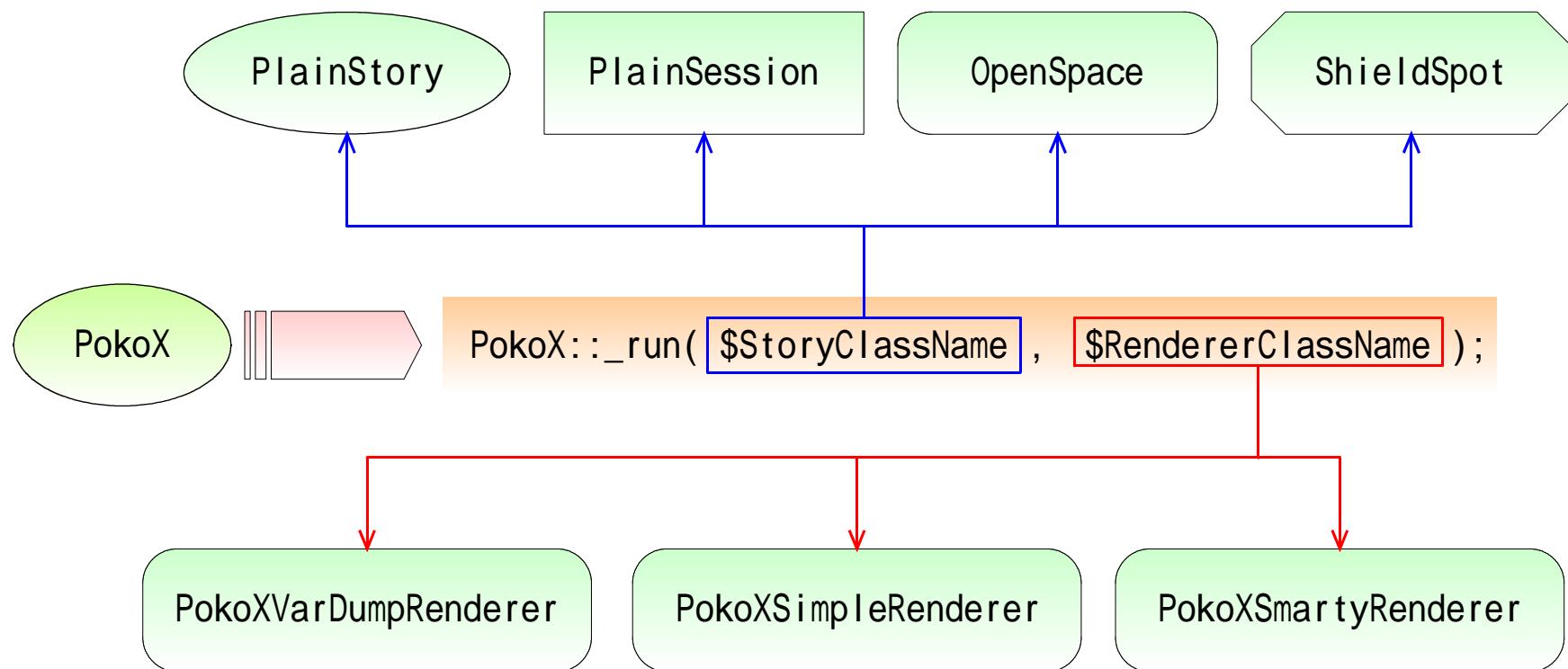
用意済のStoryクラスとRendererクラス

PlainStory : 何もせずに\$_GET/POSTで渡ってきたStoryをそのままスルー。ACIは"**"

PlainSession : セッションを開始し、\$_POKOKをセッション変数に入れる。それ以外はPlainStoryと同じ。

OpenSpace : ユーザー認証用の関数をテンプレートとして持つ。ログインしなくともある程度動き回れるCMS系

ShieldSpot : ユーザー認証用の関数をテンプレートとして持ち、ログインしなければ中に入れない会員制サイト向

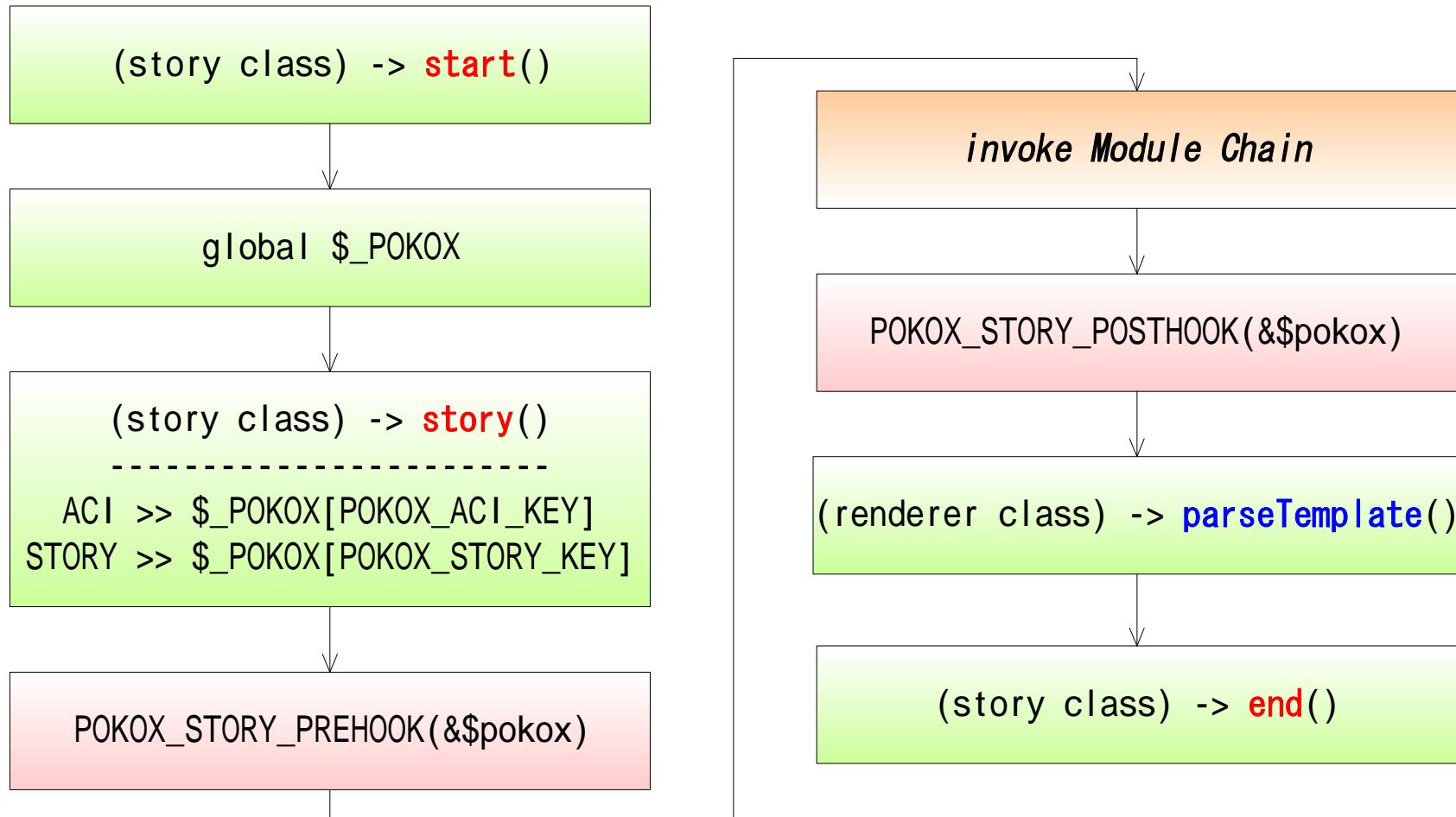


PokoXVarDumpRenderer : セットされた変数をvar_dumpするだけで、HTMLテンプレートは使わない。

PokoXSimpleRenderer : <?php ~ ?>タグが使用されたPHPファイルをテンプレートとしてrequireする。

PokoXSmartyRenderer : Smartyレンダラ。基本的にこれしか使わない。

基本的な処理の流れ(PokoX::(_|) run())



PokoXのコントローラとしての機能群(1.3.0現在)

【モジュールスタック】

- ・前に実行したモジュールのインスタンスを取得できる。

【モジュールチェインescape】

- ・後続のモジュールをスキップし、即座にレンダラ処理を実行させる。

【モジュールチェインdispatch】

- ・後続のモジュール実行をキャンセルし、別のstoryのモジュールチェインを実行する。

【モジュールReturnテンプレートリソース】

- ・\$POKOX_TEMPLATE_STORYではなく、モジュールのexecute()インターフェイスの返した値をテンプレートリソースとして優先する。

【レンダラ無効化】

- ・レンダラ処理をスキップさせる。

【正規表現story】

- ・\$POKOX_MODULE_STORY_EXを用いることで、storyキーの比較に正規表現を使える。

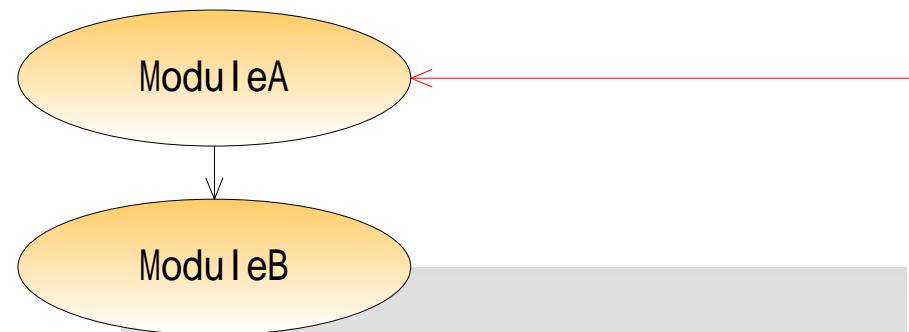
コントローラとして、「こんな機能があったらなあ・・・」を実現していく。

特有の機能：モジュールスタック

```
class ModuleSample {
    function execute(&$pokox,
        $params, $mid = null) {
        ...
        $pokox->renderer->assign(
            "var", $result);
        ...
    }
}
```

`$POKOX_MODULE_STORY = array(
 "user.insert" => array(
 "ModuleA.id1" => array(...),
 "ModuleB" => array(...),
 ...),
 ...
)`

モジュール指定時に、「.id」としてIDを振ることで、後続モジュールからインスタンスを取得できます。



```
class ModuleB {
    function execute(&$pokox, $params,
        $mid = null) {
        ...
        $modA =& $pokox->
            getStackedModule("id1");
        ...
    }
}
```

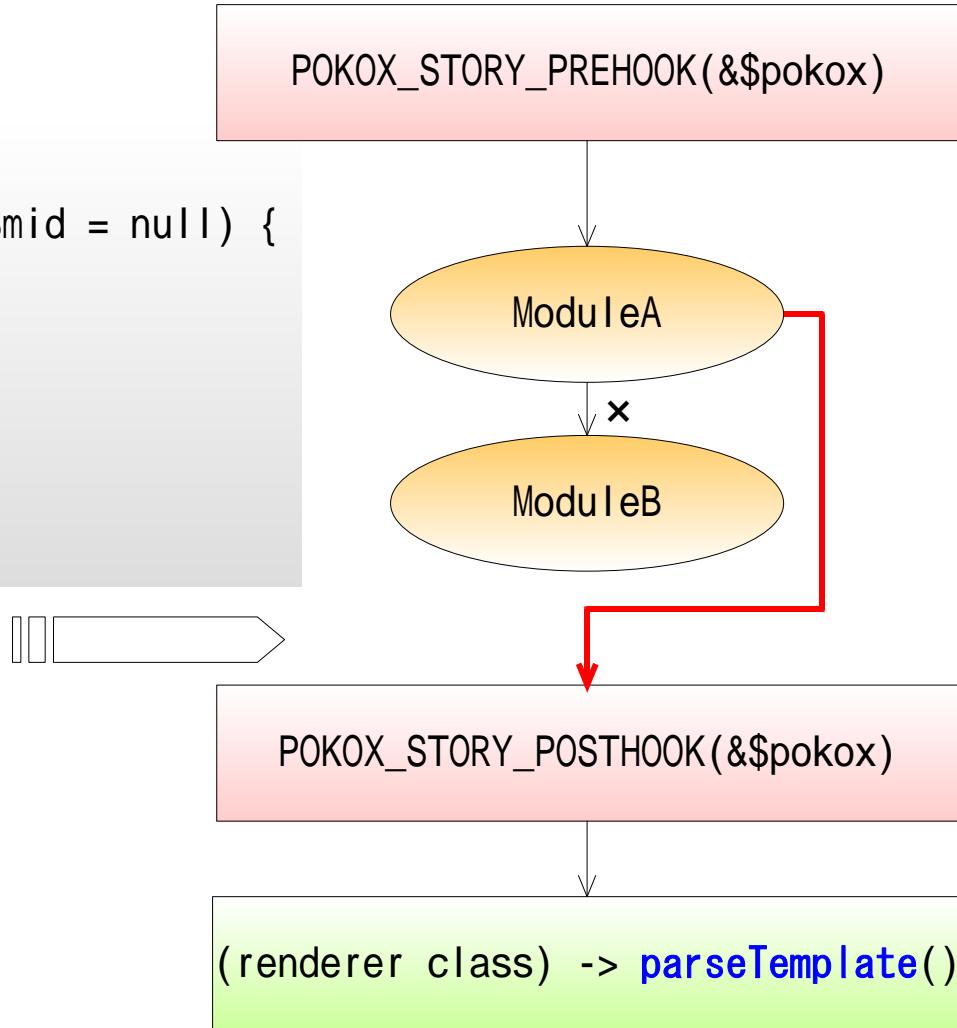
特有の機能：モジュールチェインescape

Locationヘッダーの送出時など、後続モジュール処理をスキップさせたい・しなければならない時に用います。

```
class ModuleA {  
    function execute(&$pokox, $params, $mid = null) {  
        ...  
        header("Location: http://...");  
        $pokox->escape(false);  
        ...  
    }  
}
```

`escape()`の引数を省略するか、`true`を指定すると
`parseTemplate()`自体も省略する。(HTML出力を行わない)

```
$POKOX_MODULE_STORY = array(  
    "user.story1" => array(  
        "ModuleA" => array(...),  
        "ModuleB" => array(...)),  
    ...
```



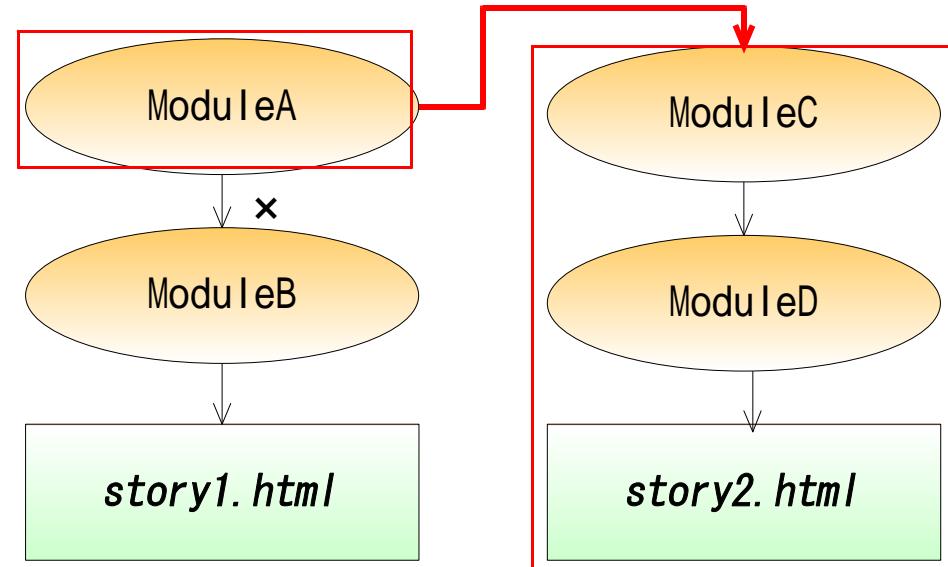
特有の機能：モジュールチェインdispatch

```
class ModuleA {  
    function execute(&$pokox, $params,  
        $mid = null) {  
        ...  
        if(!hoge_exec()) {  
            $pokox->dispatch("story2");  
        } ....  
    }  
}
```

```
$POKOX_MODULE_STORY = array(  
    "user.story1" => array(  
        "ModuleA" => array(...),  
        "ModuleB" => array(...)),  
    "user.story2" => array(  
        "ModuleC" => array(...),  
        "ModuleD" => array(...)),  
    "admin.story1" => array(...),  
    "admin.story2" => array(...),  
    ...
```

```
$POKOX_TEMPLATE_STORY = array(  
    "user.story1" => "story1.html",  
    "user.story2" => "story2.html",  
    ...
```

エラー処理などで、別のstoryに切り替えた
い場合に用います。



特有の機能：モジュールReturnテンプレートリソース

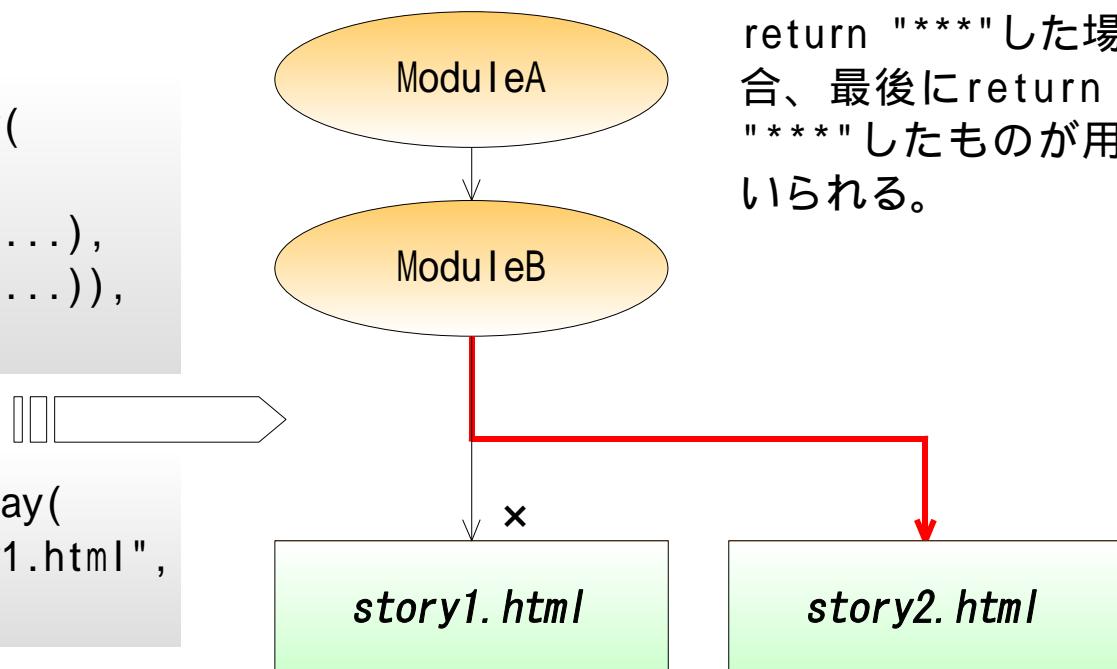
エラー時に専用のページを表示させたい場合に用います。

```
class ModuleA {  
    function execute(&$pokox, $params, $mid = null) {  
        ...  
        return "story2.html";  
    }  
}
```

```
$POKOX_MODULE_STORY = array(  
    "user.story1" => array(  
        "ModuleA" => array(...),  
        "ModuleB" => array(...)),  
    ...  
)
```

```
$POKOX_TEMPLATE_STORY = array(  
    "user.story1" => "story1.html",  
    ...  
)
```

後続モジュールも
return "****"した場
合、最後にreturn
"****"したもののが用
いられる。



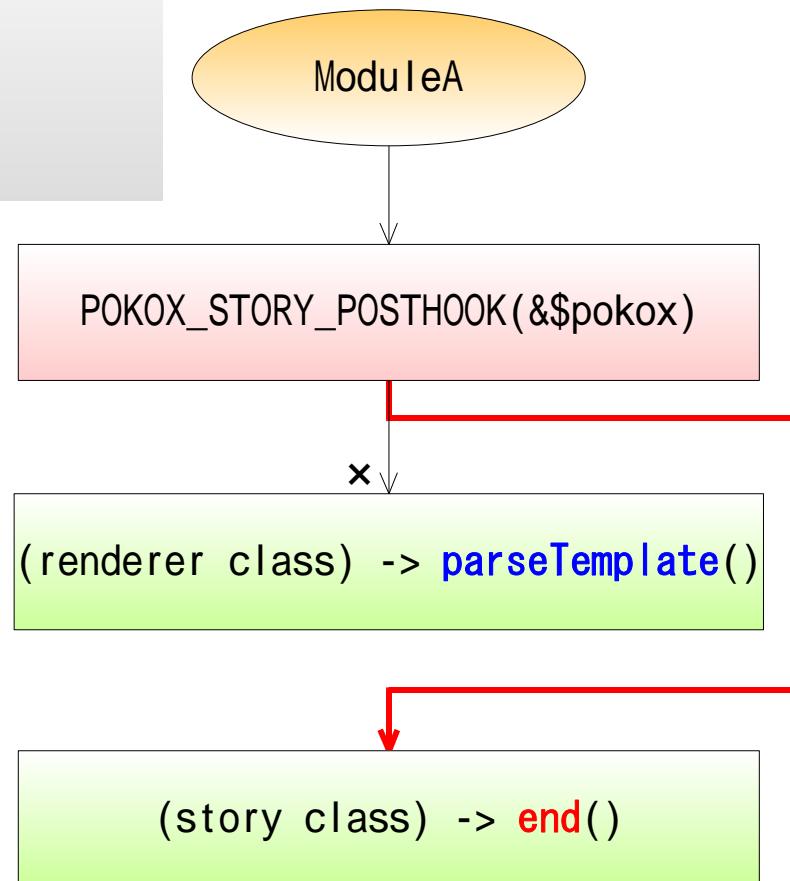
特有の機能：レンダラ無効化

バイナリデータのDL機能など、特殊な出力を行いたいときに用います。

```
class ModuleA {  
    function execute(&$pokox, $params, $mid = null) {  
        ...  
        (no return)  
    }  
}
```

```
$POKOX_MODULE_STORY = array(  
    "user.story1" => array(  
        "ModuleA" => array(...)),  
    ...  
)
```

```
$POKOX_TEMPLATE_STORY = array(  
    "user.story1" => "story1.html",  
    ...  
)
```



特有の機能：正規表現story

```
$POKOKX_MODULE_STORY_EX = array(  
    "/^.*/" => array(...), ←  
    "admin./.*/" => array(...), ←  
    "admin./^insert.*/" => array(...), ←  
    "admin./^insert_Y$/" => array(...), ←  
    "user./.*/" => array(...), ←  
    "user./^update.*/" => array(...), ←  
    "user./^update_Y$/" => array(...), ←  
    "user./^insert.*/" => array(...), ←  
    "*./.*$/ " => array(...), ←  
);
```

フィルタ風に動作させたい時に用います。

全storyで共通の前処理モジュール

"admin"ACIで共通の前処理モジュール

"admin"ACIで、"insert"で始まるstoryの
共通前処理モジュール

"admin"ACIで、"insert_Y"のstoryのみの
モジュール

"user"ACIで共通の前処理モジュール

"user"ACIで、"update"で始まるstoryの
共通前処理モジュール

"user"ACIで、"update_Y"のstoryのみの
モジュール

"user"ACIで、"insert"で始まるstoryの
共通前処理モジュール

全storyで共通の後処理モジュール

サンプル1：セッションを用いたカウントアップ

```
<?php
require_once("pokox.inc.php");

class Counter {
    function execute(&$pokox,
        $params, $mid = null) {
        global $_POKOX;
        $_POKOX['counter']++;
        $pokox->renderer->assign(
            'counter', $_POKOX['counter']);
    }
}

$POKOX_MODULE_STORY = array(
    "*.*" => array("Counter" => array())
);

echo PokoX::__run('PlainSession',
    'PokoXVarDumpRenderer');

?>
```

クラスファイルのrequire

モジュールクラスの実装

PlainSession側で自動的に作ってくれる
\$_POKOXを利用してセッション変数をカウン
トアップし、レンダラにassignする。
(\$_POKOX自体は、PlainSessionでは
\$_SESSION['POKOX_SESSION_VARS']
として取得されます。)

モジュールチェインの定義

実行結果をecho。assingされたのを
var_dump()した文字列が表示される。

サンプル2：独自Storyクラスの作成（1）

【実際にあった話】

携帯サイトのため、文字コードをSJISにして、`session.use_trans_sid`を有効にしたい。ただしSmartyテンプレートファイルはEUC-JP。

【作り方】

- `session_start()`した後に、`ob_start()`を開始させる。
- `PokoXSmartyRenderer`の出力をトラッピングして、`mb_convert_encoding()`したものが出力させる。
`ob_start("mb_output_handler")`という手もある。

```
class OBStory extends PokoXStory {

    function start(&$pokox) {
        session_start();                                // セッションを開始にする。
        ob_start();                                     // 出力バッファリングを開始する。(逆にしない！)

        if(!isset($_SESSION["POKOKX_SESSION_VARS"])) {   // $_POKOKX変数をセッションより取得する。
            $_SESSION["POKOKX_SESSION_VARS"] = array();
        }
        $GLOBALS["_POKOKX"] =& $_SESSION["POKOKX_SESSION_VARS"];
        global $_POKOKX;

        if(!isset($_POKOKX[POKOKX_ACI_KEY]))           // ACIはデフォルトの"*"とする。
            $_POKOKX[POKOKX_ACI_KEY] = "*";
    }
}
```

(次頁へ)

サンプル2：独自Storyクラスの作成（2）

（前頁より）

```
function story(&$pokox) // STORYは基本的に$_GET/$_POSTで渡ってきたものを素通し
{
    if(isset($_GET[POKOX_STORY_KEY])) {
        $story = $_GET[POKOX_STORY_KEY];
    } else if(isset($_POST[POKOX_STORY_KEY])) {
        $story = $_POST[POKOX_STORY_KEY];
    } else {
        $story = "";
    }
    return $story;
}

function end(&$pokox)
{
    echo $pokox->getContents(); // PokoXの内部バッファに格納されているレンダラ出力を、ob内で出力。
    $contents = ob_get_contents(); // ob内に出力されたレンダラ出力(Smarty出力のEUC-JP)を取得
    $contents = mb_convert_encoding($contents, 'SJIS', 'EUC-JP'); // SJISに変換
    ob_end_clean(); // ob内をクリア・ob終了
    $pokox->setContents($contents); // SJISに変換されたコンテンツをPokoXに再セットする。
}
}

// ここまでがStoryクラス
```

サンプル2：独自Storyクラスの作成（3）

前頁までで作成した独自Storyクラスの'OBStory'をPokoX::__run()で指定する。

```
<?php
require_once("Smarty.class.php"); // Smartyクラスの読み込み
require_once("pokox.inc.php"); // PokoXの読み込み
require_once("PokoXSmartySampleRenderer.class.php"); // PokoX添付のサンプルのSmartyRendererを利用する。

class Counter // モジュールクラス自体はサンプル1を流用
{
    ...
}

$POKOX_TEMPLATE_STORY = array( // テンプレートファイルを指定
    "*.*" => "template.html"
);

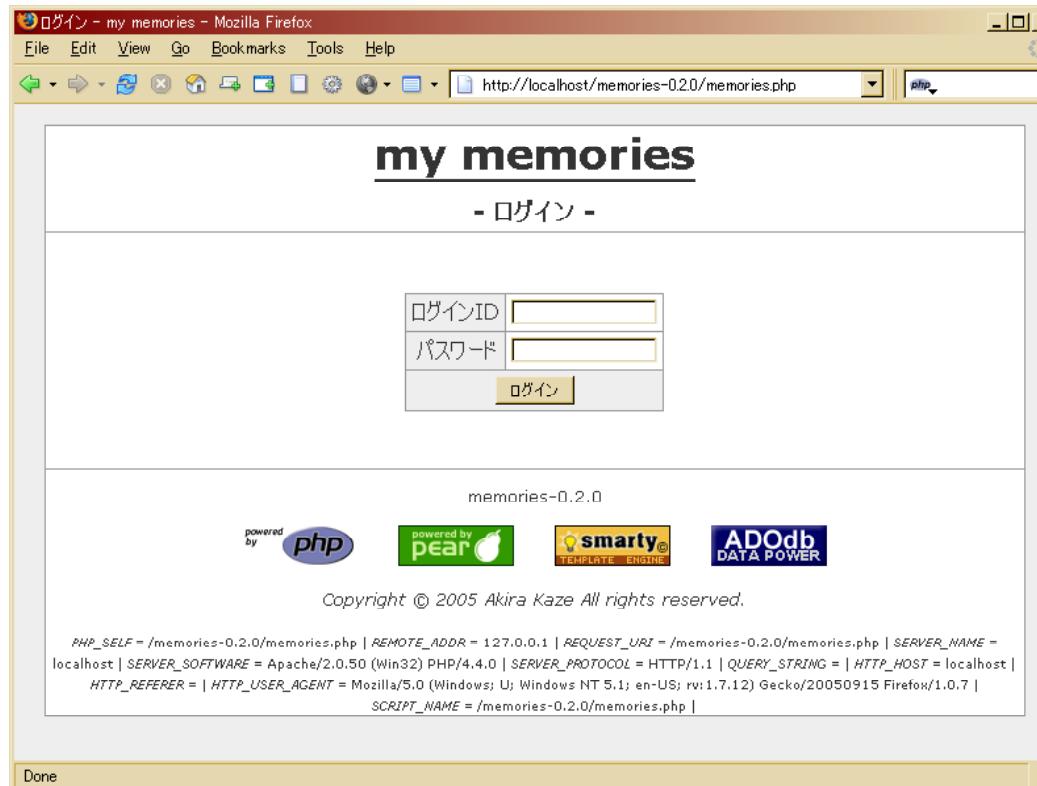
$POKOX_MODULE_STORY = array(
    "*.*" => array("Counter" => array()),
);

echo PokoX::__run('OBStory', 'PokoXSmartySampleRenderer'); // (__run()の第一引数に独自Storyクラス名を渡す。
?>
```

サンプル3：PokoXを利用したサイト構築のリファレンスApp

【memories】

- PokoXを利用したサイト構築のリファレンスコードを目指す。
- Wikiの書式やプラグインが使えるBlogを作つてみたい。
- 記事のマルチカテゴライズ＆カategoriマスク検索がしたい。
- 記事毎にACL(Access Control List)を持たせたい。
　　・・・つくってます。・・・作る意味あるのか？



サンプル3：PokoXを利用したサイト構築のリファレンスApp

ユーザ管理 - my memories - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/memories-0.2.0/modules/users/admin.php

my memories

- ユーザー管理 -

| ユーザー管理 | グループ管理 |

11 件中 1 ~ 10 件を表示 1 2

ユーザー名(ログインID/メールアドレス)	権限	状態	操作
Administrator (admin / feng-jinq-csyc-2s@qlamenv-septzen.net)	System管理者	有効	[編集] - [削除]
ユーザー1 (user1 / user1@hoge.net)	-	有効	[編集] - [削除]
一般ユーザー10 (user10 / user10@hoge.com)	System管理者	有効	[編集] - [削除]
User11 (user11 / user11@hoge.com)	System管理者	有効	[編集] - [削除]
User2 (user2 / user2@hoge.com)	Group管理者	有効	[編集] - [削除]
User3 (user3 / user3@hoge.com)	-	有効	[編集] - [削除]
User4 (user4 / user4@hoge.com)	-	有効	[編集] - [削除]
User5 (user5 / user5@hoge.com)	-	有効	[編集] - [削除]
User6 (user6 / user6@hoge.com)	-	有効	[編集] - [削除]
一般ユーザー8 (user8 / user8@hoge.com)	-	無効	[編集] - [削除]

ログインID で 暗順 に並び替え 10 件ずつ表示

>> ユーザーの新規作成 <<

Administraturでログイン中
[ログアウト](#)

memories-0.2.0

Powered by powered by smarty® TEMPLATE ENGINE

Copyright © 2005 Akira Kaze All rights reserved.

\$_PHP_SELF = /memories-0.2.0/modules/users/admin.php | \$_REMOTE_ADDR = 127.0.0.1 | \$_REQUEST_URI = /memories-0.2.0/modules/users/admin.php | \$_SERVER_NAME = localhost |
\$_SERVER_SOFTWARE = Apache/2.0.50 (Win32) PHP/4.4.0 | \$_SERVER_PROTOCOL = HTTP/1.1 | \$_QUERY_STRING = | \$_HTTP_POST = localhost | \$_HTTP_REFERER =
http://localhost/memories-0.2.0/memories.php | \$_HTTP_USER_AGENT = Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.12) Gecko/20050915 Firefox/1.0.7 | \$_SCRIPT_NAME =
/memories-0.2.0/modules/users/admin.php |

Done

最後まで聴いて頂き、
有り難うございました。

m(_)m