開発者マニュアル

著作権

本書は、情報提供のみを目的としています。NHN(株)は、本書に記載された情報の完全性と正確性を検証するために細心の注意をもって作成しましたが、発生し得る内容上の誤りや記載漏れに対しては責任を負いません。したがって、本書の使用や使用の結果による責任は一切利用者のものとし、NHN(株)はこれについて明示的、黙示的を問わず何ら保証を行うものではありません。
URL 情報を含め、本書で言及されている特定のソフトウェア商品、製品は該当所有者の著作権法に従うものとし、該当の著作権法の遵守は、利用者の責任で行うものとします。
NHN(株)は、本書の内容を将来予告なしに変更することがあります。

オープンソースライセンス使用告知

XE は、様々なオープンソースライセンスのうち、LGPL(GNU Lesser General Public License) v2 を採用しています。LGPL v2 と v3 の間には、多少の違いがあるため、バージョン v2 を採用している点に注意が必要です。LGPL は、基本的には GPL と同様ですが、適用範囲がより制限的であるという点で異なります。LGPL も、GPL と同様に、該当ライセンスを保有するソフトウェアを含むソフトウェアに同じライセンスを強制する効力があります。しかし、GPL が GPL ライセンスを保有したソフトウェアを含むすべてのソフトウェアに無条件にソース公開を強制するのに対し、LGPL ライセンスを保有するプログラムは、特定の条件内で使用する場合であれば、ソース公開の義務はありません。したがって、LGPL ライセンスを保有するソフトウェアは、独占的ソフトウェアの開発にも利用することができます。詳細については、下記のサイトを参照してください。

LGPL ライセンス : http://www.gnu.org/copyleft/lesser.html
GPL ライセンス : http://www.gnu.org/licenses/gpl.html

はじめに

本書について

本書では、XE モジュールとアドオン、ウィジェットなど XE の追加機能を開発する方法について説明します。なお、本書の内容は、XE core バージョン 1.5 をもとにされています。

読者

本書の対象読者は、XE の追加機能を開発する開発者です。本書では、Web サーバーや PHP の技術については詳細に取り扱いません。Web サーバーと PHP の技術については、関連書籍を参考にしてください。

連絡先

本書の内容に誤りや不明な点がありましたら、以下の連絡先までお問い合わせください。

電子メール： developers@xpressengine.com

改版履歴

版数	日付	履歴
1.0	2011.07.29	1.0 配布
1.1	2011.12.15	XE core 1.5 アップデートに伴い改訂
1.1	2012.09.06	Modified sub-query.

表記規則

参考表記

> **参考**
> 読者が参考にすべき内容を記述します。

注意表記

> **注意**
> 読者が知っておくべき内容、システムエラーを引き起こす恐れのある内容、実行しない場合は財産上の被害を被りかねない内容を記述します。

画面名/メニュー名/選択値/ユーザー入力値の表記

本書では、画面名、メニュー名、選択値、ユーザー入力値を以下のように表記します。

- 画面名：**画面名**画面
- メニュー名：**メニュー > 下位メニュー**
- 選択値： **NBoard 1.0** を選択します。
- ユーザー入力値：*home page* を入力します。

ソースコードの表記

本書では、ソースコードを、以下のように灰地に黒文字で表記します。

```
COPYDATASTRUCT st;
st.dwData = PURPLE_OUTBOUND_ENDING;
st.cbData = sizeof(pp);
st.lpData = &pp;
::SendMes(GetTargetHwnd(), WM_COPYDATA, (WPARAM)this->m_hWnd, (LPARAM)&st);
```

目次

図表一覧

表一覧

図一覧

1. XE core について

この章では、XE core の基本情報について説明し、XE のリクエストライフサイクルとフォルダ構造について説明します。

1.1 概要

XE core は、開発者が Web アプリケーションを作成する際の基盤となるフレームワークです。XE core は、会員管理、ドキュメント/コメント管理だけでなく、DBMS の種類に依存せずにデータを管理する機能を提供します。XE 自体が MVC(Model-View-Controller)構造になっているため、完璧な SoC(Separation of Concerns)を実装することができます。

XE アプリケーションへ流れ込むすべてのリクエストは、index.php で処理します。このページは、リクエストコンテキストを初期化し、適切なモジュールを探し、クライアント(ブラウザ)へ応答を送る役割をします。

XE のほとんどの機能は、モジュールで構成されています。XE は、リクエストがあったら、モジュール名とアクション名(なければ基本値を使用)を基準にどのモジュールを使用するかを決定します。たとえば、管理者ページを表示するための URL は<root_url>?module=admin&act=dispBoardAdminContent になります。

この章では、XE 基盤の追加機能であるモジュール、アドオン、ウィジェットを開発するために知っておくべき XE の基本構造とリクエストライフサイクルについて説明します。

1.2 XE のリクエストライフサイクル

XE のリクエストライフサイクルとは、URL に接続した瞬間からクライアントへ応答を送るまでの、XE の一連の過程のことをいいます。下記図にて、XE リクエストライフサイクルを示します。

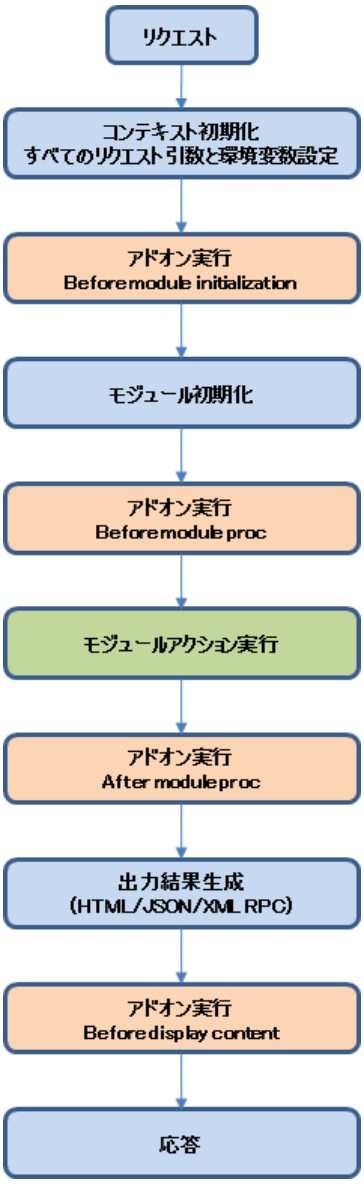


図 1-1 XE リクエストライフサイクル

XE リクエストライフサイクルの主要プロセスは、下記のとおりです。

1. コンテキスト初期化
2. モジュール初期化
3. 要求されたモジュールアクションの実行
4. 応答結果生成

開発者は、アドオンを使用することで、このライフサイクル内の特定のタイミングにカスタムコードを実行することができます。アドオンは、XE 追加機能の一種で、PHP include メカニズムで動作します。コアメソッドに直接、コードが含ま

れているため、ライフサイクルを操作する機能を実装できます。アドオンについての詳細は、「2.2 アドオン」を参照してください。

1.2.1 コンテキスト初期化

コンテキスト初期化は、Context クラスで処理します。このクラスは、XE アクションの実行環境をカプセル化します。主な役割は、下記のとおりです。

- $GLOBALS のコンテキスト変数設定(display handler で使用)
- 言語タイプにしたがって言語ファイルをインクルードする
- コンテキストとセッションの認証(authentication)情報設定
- サーバーで rewrite モードを使用するか確認
- JavaScript 使用のための位置設定

Context クラスの位置は、./classes/context/Context.class.php です。

1.2.2 モジュール初期化

モジュール初期化は、ModuleHandler クラスの init()メソッドで処理します。init()メソッドの役割は、下記のとおりです。

- モジュールを初期化する前にアドオン実行(before_module_init hook)
- リクエスト引数で変数設定
- XSS を防止するための変数認証
- module_srl、mid、document_srl を基準にリクエストモジュール検索
- 現在のモジュール情報をコンテキストに設定

ModuleHandler クラスの位置は、./classes/module/ModuleHandler.class.php です。

1.2.3 リクエストモジュールのアクション実行

すべてのモジュールは、ModuleHandler クラスの procModule()メソッドを通じて実行されます。このメソッドの役割は、下記のとおりです。

- モジュールを実行する前にフックされたアドオンを実行(before_module_proc hook)
- 現在モジュールのアクションを実行

1.2.4 応答結果の生成

DisplayHandler クラスは、結果生成を担当します。リクエストの種類によって HTML や XML/JSON コンテンツを出力できます。HTML の場合、このクラスは、指定されたテンプレートファイルを検索して解析し、完成された HTML 形式を作成します。XML/JSON の場合、ModuleObject 属性は、別途のフォーマッティングが必要なく、XML/JSON にシリアライズ(serialize)されます。

1.3 XE フォルダ構造

XE をインストールすると、ルートに下記のファイルとフォルダが生成されます。

表 1-1 XE のファイルとフォルダ

フォルダ/ファイル	説明
addons	すべての XE アドオンが含まれています。
classes	XE core クラスが含まれています。
common	すべての XE モジュールに共通的に使用される静的ファイルとテンプレートが含まれています。グローバル言語ファイルもこのフォルダに含まれています。
config	基本設定と共通機能が含まれています。
files	このフォルダは、インストール中に生成され、アップロードされたファイルと内部キャッシュファイル、DB/環境設定ファイルを保存します。
layouts	基本、カスタム XE レイアウトをすべて含んでいます。
libs	XE core で使用するすべてのライブラリが含まれています(例: ftp, tar)。
m.layouts	モバイルレイアウト
modules	すべてのモジュール(XE core モジュール、カスタムモジュール)が含まれています。
themes	テーマ(レイアウトと各種モジュールのスキンをすべて含む)が含まれています。
widgets	すべての XE ウィジェットが含まれています。
widgetstyles	ウィジェットを飾るために必要なすべてのウィジェットスタイルが含まれています。
index.php	XE のすべての入出力のためのゲートウェイ機能が含まれています。
.htaccess	Apache Web Serve の rewrite mod を使用するための設定ファイル
LICENSE	XE のライセンスが含まれています。

1.3.1 addons フォルダ

アドオンは、単純に有効/無効に設定可能であり、追加設定が必要な場合にはモジュールと連動します。addons フォルダ構造は、下記表のとおりです。

表 1-2 addons フォルダ構造

フォルダ/ファイル	説明
addons	アドオンの最上位フォルダ
addon_name	アドオン名になっているフォルダ
conf	アドオンの設定ファイルフォルダ
info.xml	アドオンの説明と製作者、バージョン、および作成日情報を作成します。
addon_name.addon.php	アドオン実行コード。アドオン実行時点に挿入されます。
queries	アドオンに使用するクエリの集まり。
queryID.xml	クエリファイル。クエリスキーマは、モジュールで使用するものと同じです。

詳細な内容については、「2.2 アドオン」を参照してください。

1.3.2 classes フォルダ

classes フォルダには、XE モジュールとアドオン、ウィジェットなどのコンポーネントが共通的に使用するライブラリクラスが含まれています。フォルダ別に下記のクラスを提供します。

表 1-3 classes フォルダ別クラス

フォルダ	説明
cache	XE core で使用できるすべてのキャッシュクラス(CacheAPC, CacheMemcache, CacheHandler)が含まれています。基本クラスは、CacheHandler。
context	リクエスト引数や環境変数のようなコンテキストを管理するコンテキストクラスが含まれています。
db	CUBRID、MySQL、Firebird、MySQL Innodb、MySQLi、PostgreSQL、SQLite2、SQLite3 with PDO など XE core でサポートするすべての DB が含まれています。
display	実行結果出力を担当するクラスが含まれています(リクエストタイプによって異なる： HTML、JSON、または MLRPC)。
editor	エディターハンドラークラスが含まれています。
extravar	記事、会員などに使用する拡張変数を処理するクラスが含まれています。
file	ファイルとフォルダ処理に使用するクラスが含まれています。
handler	(*)Handler の抽象クラスが含まれています。
httprequest	外部サーバーへ HTTP リクエストを送り、応答を収集するために使用するクラスが含まれています。
mail	メーリング関連クラスが含まれています。
mobile	モバイル最適化のためのクラスが含まれています。
module	モジュールハンドラーとモジュールオブジェクトクラスが含まれています。
object	XE モジュール間のオブジェクトインスタンスを渡す基本クラスが含まれています。
page	ページナビゲーション(navigation)を処理する基本クラスが含まれています。
template	正規表現式を利用してテンプレートファイルを PHP コードに変換し、その後も使用するために XE キャッシュで作成するクラスが含まれています。
widget	ウィジェット実行のためのハンドラークラスが含まれています。
xml	XML をパースして生成するクラスが含まれています。

1.3.3 common フォルダ

common フォルダには、XE で必ず必要なリソースが含まれています。

表 1-4 common フォルダ構造

フォルダ/ファイル	説明
common	XE で使用する共通の JS、CSS ファイル集
css	XE で使用する共通の CSS ファイル集
default.css	基本スタイルと XE に特化されたスタイルの定義
button.css	XE で使用する基本ボタンスタイルの定義
js	XE で使用する共通 JS ファイル集

フォルダ/ファイル	説明
common.js	XE で使用する様々なタイプの JavaScript 関数定義
jquery.js	XE で使用する JavaScriptフレームワークである jQuery(http://jquery.com)ファイル
js_app.js	XE で使用する JavaScript アプリケーションフレームワークである JAF ファイル
x.js	クロスブラウジングのための JavaScript ライブラリファイル。今後、削除予定のため、このファイルはなるべく使用しないでください。
xml_js_filter.js	XE で使用する XML JS フィルターファイル
lang	XE でサポートする言語ファイルを含むフォルダ
tpl	XE の共通レイアウト/テンプレートファイル集
common_layout.html	XE で使用する共通レイアウト
default_layout.html	使用中のレイアウトスキンがないとき、コンテンツだけを出力する空のレイアウト
mobile_layout.html	XE モバイル環境で使用するレイアウト
popup_layout.html	XE でポップアップ画面を開くときに使用するレイアウト
redirect.html	別ページへ移動しなければならないときに使用するテンプレートファイル
refresh.html	リフレッシュするときに使用するテンプレートファイル

1.3.4 config フォルダ

設定フォルダには、基本設定内容と、よく使う関数の集まりを保存しているファイルが含まれています。

表 1-5 config フォルダ構造

フォルダ/ファイル	説明
config.inc.php	開発者のための XE バージョンとデバッグ設定ファイル
config.user.inc.php	開発者のためのデバッグ設定保存ファイル(開発者が直接作成して使用)
func.inc.php	XE でよく使う関数が含まれているファイル

1.3.5 files フォルダ

キャッシュファイル、アップロードされたファイル、その他モジュールで必要なファイルが含まれています。

表 1-6 files フォルダ構造

フォルダ/ファイル	説明
_debug_message.php	XE ログファイル。 ./config/config.inc.php の_DEBUG_OUTPUT_常数に設定された値によって PHP エラーメッセージと DB エラーなどを表示します。このファイルは、基本的には存在せず、デバッグログが発生したときに生成されます。
attach	ファイル添付(アップロードファイル)のためのフォルダ
binaries	gif、jpg、jpeg、png、swf、mpeg 以外の拡張子の添付ファイルを保存します(悪意的な攻撃を避けるために fpassthru()関数を使用してファイルを実行(execute)せずにコンテンツをクライアントに渡す)。

フォルダ/ファイル	説明
images	ブラウザで直接アクセスできる静止画と動画像ファイルを保存します。下位フォルダ名は、./$module_srl/$document_srl/$file_name の形式で作成します。
cache	キャッシュフォルダ
addon	アドオン関連のキャッシュファイルの集まり
mobileactivated_addons.cache.php	有効アドオンを実行する PHP コードが含まれています(モバイル環境)
pcactivated_addons.cache.php	有効アドオンを実行する PHP コードが含まれています(PC 環境)
document_category	文書カテゴリ用 XML、PHP キャッシュファイル
editor	エディターコンポーネントの情報キャッシュファイル
js_filter_compiled	XE の XML JS フィルターのキャッシュファイル
lang_defined	ユーザー定義言語コードのキャッシュファイル
layout	XE のレイアウトキャッシュファイル。レイアウト編集で修正されたレイアウトコンテンツは、ここで保存されます。
menu	XE のメニューモジュールで生成されたメニュー情報のための XML と PHP キャッシュファイル
module_info	各 XE モジュールの情報キャッシュファイルを保存します。
opage	XE の外部ページモジュール用のキャッシュファイル
optimized	CSS と JS ファイルを統合してトラフィックを削減し、ページの読み込み速度を向上させるための最適化キャッシュファイル
page	XE のページモジュール用キャッシュファイル
queries	XE の XML Query コンパイル用キャッシュファイル
template_compiled	XE のテンプレートキャッシュファイル
thumbnails	XE の文書サムネイル画像
widget	XE のウィジェット情報用キャッシュファイル
widget_cache	生成されたウィジェット情報を保存し、活用するキャッシュファイル。キャッシュ時間がウィジェット内に指定された場合、キャッシュファイルを保存します。
triggers	XE のトリガー関数用キャッシュファイル
widgetstyles	ウィジェットスタイルの情報を保存し、活用するキャッシュファイル
newest_news.language.cache.php	管理者ページにある最新ニュースの一時保存ファイル
config	DB、FTP などサイトのトップマネージャの設定情報の集まり
db.config.php	DB 設定ファイル
ftp.config.php	XE のインストールされているサーバーのファイルを保存する FTP 情報
lang_selected.info	管理者の作業する特定サイトの言語リストを保存します。
member_extra_info	会員情報の拡張変数に使用されたファイル集
image_mark	会員の名前の前につけるマークの画像ファイル

フォルダ/ファイル	説明
image_name	会員画像の名前ファイル
profile_image	会員が登録したプロフィール画像ファイル
signature	会員の署名
point	各会員のポイント
new_message_flags	特定会員に新しいメッセージが届いたかどうかを示すテンポラリファイルの位置
agreement.txt	会員管理モジュールに設定した利用規約を保存するファイル
ruleset	動的ルールセットファイルを保存します。
theme	現在のテーマ情報を保存します。

1.3.6 layouts フォルダ

レイアウトは、コンテンツ(モジュール)を囲む「殻」といえます。レイアウトは、モジュールインスタンス別に設定して適用したり、製作者が設定したメニューインスタンスと連動してメニューインスタンスに含まれているすべてのモジュールインスタンスに一括で適用したりできます。また、レイアウト管理メニューでウィジェットやレイアウト編集機能を通じてレイアウトテンプレートファイルを修正することができます。

表 1-7 layouts フォルダ構造

フォルダ/ファイル	説明
Layout Name	レイアウトルートフォルダ
conf	レイアウト情報が含まれている設定ファイルが含まれています。
info.xml	レイアウト製作者と説明、拡張変数、連動メニューの数と名前を定義します。
layout.html	レイアウトのテンプレートファイル

1.3.7 modules フォルダ

modules フォルダ構造は、下記表のとおりです。

表 1-8 modules フォルダ構造

フォルダ/ファイル	説明
module_name	モジュール名になっているフォルダ
conf	モジュール説明、アクションと権限(permission)設定が含まれています。
info.xml	モジュール製作者情報と説明
module.xml	モジュール動作関連の情報を含むアクションモジュール定義
lang	言語パックファイル
en.lang.php	英語言語パック
schemas	モジュールインストールに使用する DB テーブルスキーマ。 現在のモジュールが新しい DB を使用するときだけ使用するオプションフォルダ。
table.xml	テーブルスキーマ(テーブル名でファイルを生成)

フォルダ/ファイル	説明
queries	insert、select、update に使用されるクエリを定義する XML 文法ファイル
ruleset	モジュールで使用するルールセットの XML ファイル
tpl	モジュールの管理者ビュー(administrator view)のために使用するテンプレートファイル
css	スタイルシート
images	テンプレート画像保存
js	テンプレート JS ファイル保存
filter	処理ファイルに転送されるフォームでノードとパラメータを宣言します。
template_files.html	スキンが使用されていない画面の、XE テンプレート文法で製作したスキンファイル(モジュールの管理者画面など)
skins	モジュールのフロントエンドに出力されるスキンファイル
Skin Name	スキンの名前
css	スタイルシート
images	スキンイメージ保存
js	スキン JS ファイル保存
skin.xml	スキン製作者情報とスキンの拡張変数宣言が含まれています。
template_files.html	XE テンプレート文法で製作したスキンファイル
module_name.class.php	インストール、アップデート、削除関数を含むモジュールの基本クラス
module_name.view.php	モジュールのフロントエンドを出力するビュー関数
module_name.model.php	モジュールモデルクラスと関数定義
module_name.controller.php	ユーザーインターフェースのためのコントローラ
module_name.admin.view.php	モジュールのバックエンドを出力するために使用するビュークラスと関数
module_name.admin.model.php	管理者用モデルクラスと関数を宣言します。
module_name.admin.controller.php	管理者関数のコントローラアクション
module_name.api.php	ビュー機能のように、画面出力のためのデータを用意します。より正確には、出力結果で内部データを削除し、Web のみでなく、アイフォンアプリのように他の種類のアプリケーションを生成するためのインスタンスに使用される JSON や XML を返します。
module_name.wap.php	出力結果の異なる WAP 携帯のためのクラス
module_name.smartphone.php	アイフォンなど、スマートフォンのための特殊クラス

モジュールについての詳細は、「2.1 モジュール」を参照してください。

1.3.8 themes フォルダ

テーマは、レイアウトとモジュールスキンの管理を便利にする機能です。サイトデザインの統一性を高めるために使用します。

表 1-9 themes フォルダ構造

フォルダ/ファイル	説明
Theme Name	テーマルートフォルダ
conf	テーマ情報のある設定ファイルを含みます。
info.xml	テーマ製作者と説明、テーマに含まれているスキンを定義します。
layouts	レイアウトスキンのルートフォルダ
Layout Name	レイアウトのフォルダ
conf	レイアウト情報が含まれている設定ファイルを含みます。
info.xml	レイアウト製作者と説明、拡張変数、連動メニューの個数と名前を定義します。
layout.html	レイアウトのテンプレートファイル
modules	モジュールスキン集のルートフォルダ
Module Name	スキンを適用するモジュール名
css	スタイルシート
images	スキンイメージを保存します。
js	スキン JS ファイルを保存します。
skin.xml	スキン製作者情報とスキンの拡張変数宣言を含みます。
template_files.html	XE テンプレート文法で製作したスキンファイル

1.3.9 widgets フォルダ

ウィジェットは、画面に表示される小さなプログラムです。ウィジェットのうちの一部は、最新記事や会員情報(ログインフォーム)と連動し、他の一部は、外部オープン API と連動したりします。

ウィジェットフォルダの名前は、該当ウィジェットと同名でなければなりません。フォルダ構造は、下記のとおりです。

表 1-10 widgets フォルダ構造

フォルダ/ファイル	説明
Widget Name	ウィジェットルートフォルダ
widget_name.class.php	ウィジェットのクラスファイル。データを処理しテンプレートファイルを指定します。
conf	設定フォルダ
info.xml	ウィジェット情報(名前、説明)とウィジェットクラスに使用できる変数を定義します。
skins	スキンフォルダ
Skin Name	ウィジェットスキン用ファイルが含まれています。このフォルダの名前は、スキン名と同じ名前でなければなりません。
skin.xml	スキンの名前、説明、製作者、カラーセット情報を含む設定ファイル

1.3.10 widgetstyles フォルダ

このフォルダは、ウィジェットスタイルが含まれています。ウィジェットスタイルは、ウィジェットコンテナを飾るために使用します。ユーザーは、ウィジェットスタイルを利用してウィジェットの背景や枠、タイトルなどウィジェットのかたちを変更することができます。

ウィジェットスタイルのフォルダ構造は、下記表のとおりです。

表 1-11 widgetstyles フォルダ構造

フォルダ/ファイル	説明
widgetstyles	ウィジェットスタイルフォルダ
Widgetstyle names	ウィジェットスタイル名
widgetstyle.html	ウィジェットスタイル用テンプレートファイル
skin.xml	ウィジェットスタイルのタイトル、説明、製作者、拡張変数などを設定するファイル
preview.gif	ウィジェットスタイルのプレビュー

2. XE 追加機能

この章では、XE 追加機能であるモジュール、アドオン、ウィジェットを開発する方法について説明します。

2.1 モジュール

XE は、様々な追加機能を使ってアップグレードできる CMS(Contents Management System)です。追加機能のうち、最も重要な機能は、モジュールです。モジュールは、プラットフォームに新しい機能を追加するファイルの集まりです。

モジュールを生成するには、次の 3 つのルールに従わなければなりません。

- モジュールは、modules フォルダ配下のフォルダに保存しなければなりません。フォルダ名は、モジュール名と同じ名前にします。生成したモジュールを配布するには、他の開発者が作成したモジュールと名前が衝突しないよう、独創的な名前にします。
- info.xml ファイルにモジュール製作者とモジュール説明、バージョン、製作日のような一般的な情報を入力します。
- module.xml ファイルに設定パラメータとアクション定義などを保存します。

2.1.1 config/info.xml 作成

info.xml は、下記のようになっています。

```
<?xml version="1.0" encoding="UTF-8"?>
<module version="0.2">
    <title xml:lang="en">Module name</title>
    <description xml:lang="en">Module description </description>
    <version>1</version>
    <date>2011-05-01</date>
    <category>service</category>
    <author email_address="author@authorland.com" link="http://www.authoria.com/">
    <name xml:lang="en">Author name</name>
    </author>
</module>
```

<category>要素は、管理者メニューにてモジュールの分類をあらわします。入力可能なオプションは、下記のとおりです。

- service: サービス管理
- member: 会員管理
- content: 情報管理
- construction: サイト設定
- utility: 機能設定
- accessory: 付加機能設定
- system: システム管理/設定
- package: cafeXE, textyle などのパッケージモジュール

2.1.2 アクション作成

XE においてすべての入出力は、index.php を通じて処理されます。アクションリクエスト引数は、Module Handler で決定し、通常$act 変数を使用します。モジュールのアクションは、conf/module.xml ファイルで宣言されます。

```
<?xml version="1.0" encoding="utf-8"?>
<module>
    <grants>
        <grant name="post" default="guest">
            <title xml:lang="en">Post</title>
    </grants>
    <permissions>
        <permission action="dispForumAdminInsertForum" target="manager" />
        <permission action="dispForumAdminForumInfo" target="manager" />
```

```
        <permission action="procForumAdminInsertForum" target="manager" />
        <permission action="procForumAdminInsertListConfig" target="manager" />
    </permissions>
    <actions>
        <action name="dispForumIndex" type="view"     />
        <action name="dispForumContent" type="view" index="true"/>
        <action name="dispForumNoticeList" type="view" />
        <action name="dispForumContentList" type="view" />
        <action name="dispForumContentView" type="view" />
        <action name="dispForumCatogoryList" type="view" />
        <action name="dispForumContentCommentList" type="view" />
        <action name="dispForumContentFileList" type="view" />

        <action name="procForumInsertDocument" type="controller" />
        <action name="procForumDeleteDocument" type="controller" />

        <action name="dispForumAdminContent" type="view" standalone="true" admin_index="true" menu_name="forum"
menu_index="true" />
        <action name="dispForumAdminForumInfo" type="view" standalone="true" menu_name="forum" />
        <action name="dispForumAdminExtraVars" type="view" standalone="true" menu_name="forum" />
        <action name="dispForumAdminForumAdditionSetup" type="view" standalone="true" menu_name="forum" />
        <action name="procForumAdminDeleteForum" type="controller" standalone="true" menu_name="forum"
ruleset="deleteForum" />
        <action name="procForumAdminInsertListConfig" type="controller" standalone="true" menu_name="forum"
ruleset="insertListConfig" />

        <action name="dispForumCategory" type="mobile" />
        <action name="getForumCommentPage" type="mobile" />
    </actions>
    <menus>
        <menu name="forum">
            <title xml:lang="en">Forum</title>
            <title xml:lang="ko">フォーラム</title>
        </menu>
    </menus>
</module>

<action>
```

conf/module.xml で使用する属性は、下記のとおりです。

表 2-1 アクション作成に使用する属性

属性	説明
name	モジュール名も含むアクション名。管理者権限の必要なアクション名には"Admin"を含みます。
type	アクションがどんなタイプであるか、どのファイル(ビュー、モデル、コントローラ)に保存されるべきかを定義します。名前に"Admin"が含まれている場合、管理者ビュー、モデル、またはコントローラ PHP ファイルになくてはなりません。
standalone	この属性が"true"に設定された場合、現在のアクションは、他のモジュールに依存しません。この属性が"false"に設定された場合は、リクエストは受信されず、モジュールが実行されるときにエラーを出力します。 現在、使われていない属性で、deprecated になる予定です。
index	モジュールの基本アクション設定。ひとつのアクションだけに適用します。
admin_index	モジュールバックエンドの基本アクション。ひとつのアクションだけに適用します。
setup_index	モジュール設定ページで使用し、管理者アクションだけに設定可能です。
menu_name	該当アクションの属するメニューの名前

属性	説明
menu_index	この属性が"true"の場合、このアクションは現在メニューの初期アクションであることを意味します。
ruleset	該当アクションに適用するルールセットの名前
action	権限(permission)の宣言されたアクションの名前
target	サポートする権限は、下記のとおり。 • member: 会員 • manager: 管理者

2.1.3 Action Forward の使用

一般的にアクションは、XE モジュールに属しますが、ひとつのアクションがいろんなモジュールで使用される場合もあります。これを Action Forward といいます。

最も典型的な例として、RSS モジュールがあります。RSS アクションは、掲示板モジュールで定義したアクションではありませんが、Action Forward 機能で呼び出し、実行します。

```
?mid=board&act=rss
```

Action Forward を使用して独立された機能と共にモジュールを処理できます。

上記のリクエストで XE は、"board"という mid を探します。この mid が、rss アクションを含んでいなければ、XE は DB で Action Forward テーブルを通じて登録された rss を探します。rss アクションは、rss モジュールのビュータイプで DB に登録されているため、XE は掲示板のためのすべての mid 情報を設定し、rss モジュールのビューオブジェクトを生成して rss メソッドを実行します。この Action Forward は、XE がレイアウトや現在要求されたモジュールの情報を維持しながら他のメソッドを必要とする際に必要です。

別の例題として、友達リストを閲覧するための Communication モジュールの dispCommunicationFriend アクションがあります。このアクションは、現在モジュールのレイアウトを維持しながら、そのコンテンツを友達リストに交替させます。

すなわち、コンテンツ領域の出力は、指定されたアクションによって変更されることがあり、要求されたモジュールの情報によって別の結果になることがあります。

Action Forward 登録

一般的に Action Forward は、module.class.php で moduleInstall()を処理するときに保存されます。登録方法は、下記のとおりです。

```
$oModuleController = &getController('module');
$oModuleController->insertActionForward('module', 'type(Ex:controller)', 'action_name');
```

Action Forward 検証

以下のように Action Forward の登録を確認することができます。通常、module.class.php の checkUpdate()メソッドで使用されます。

```
$oModuleModel = &getModel('module');
if(!$oModuleModel->getActionForward('action_name')) ...
```

Action Forward 削除

Action Forward が必要なくなったら、以下のように削除します。

```
$oModuleModel = &getModel('module');
$oModuleModel = &getController('module');
if($oModuleModel->getActionForward('Action Name'))
```

```
$oModuleController->deleteActionForward('Module Name','Type','Action Name');
```

アクション名が(disp|proc|get)+ModuleName+ActionName になっていたら、Action Forward を登録しなくても良いです。

2.1.4 トリガー使用

あるモジュールが、他のモジュールの特定のアクションにある動作を実行させたいとき、トリガーを使用します。ただし、そのモジュールでトリガーを提供しなければなりません。たとえば、document モジュールの triggerDisplayDocumentAdditionSetup に、すでにある管理者用ビューをフォーラムモジュールで使用する場合、このときにトリガーを使用します。

トリガーを使用する方法は、下記のとおりです。

DB にトリガーを挿入する

```
$oModuleController->insertTrigger('forum.dispForumCommentSetup', 'comment', 'view', 'triggerDispCommentAdditionSetup', 'before');
```

トリガーを取得する

```
if(!$oModuleModel->getTrigger('forum.dispForumAdditionSetup', 'document', 'view', 'triggerDispDocumentAdditionSetup', 'before'))
return true;
```

トリガーを呼び出す

```
ModuleHandler:: triggerCall ('Trigger Name', 'call time (Called Position)', the trigger will be used as a parameter of the object);
```

トリガーを削除する

```
$ OModuleController-> deleteTrigger ('Trigger Name', 'module name', 'call the method belongs to the type of instance', 'call the method (Called Method)' + ',' call time (Called Position) ');
```

2.1.5 ルールセットを使用する

ルールセットは、HTML フォームの情報を、PHP にある処理メソッドに渡す際にクライアント側はもちろんのこと、サーバー側でも情報の有効性を検証するために使用します。ルールセットは、各モジュールフォルダの ruleset フォルダにある XML ファイルに保存されます。下記にて、ルールセットの例題を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<ruleset version="1.5.0">
    <customrules>
    </customrules>
    <fields>
        <field name="user_id" required="true" length="3:20" />
        <field name="user_name" required="true" length="2:40" />
        <field name="nick_name" required="true" length="2:40" />
        <field name="email_address" required="true" length="1:200" rule="email" />
    </fields>
</ruleset>
```

ルールセットで使用する要素と属性は、下記のとおりです。

表 2-2 ルールセット作成に使用する要素と属性

要素	属性	説明
customrules		ユーザー定義規則を定義できます。
rule		ユーザー定義規則
	name	ユーザー定義規則の名前
	type	ユーザー定義規則のタイプ。"regex"、"enum"、"expression"のうち、いず

要素	属性	説明
		れかひとつを使用できます。 • ″regex″: 正規表現式を作成するとき • ″enum″: 与えられた値のうち、ひとつだけ選択可能なとき • ″expression″: 修飾が必要なとき
	test	ユーザー定義規則のテストコード
fields		有効性をチェックするフィールドの集まり
field		有効性をチェックするフィールド
	name	フォーム要素名
	rule	適用する規則
	required=″true″	必ず入力しなければなりません。
	length	サイズ制限。″最小:最大″のように作成できます。
	default	デフォルト値
	equalto	equalto に入れた要素の値が、現在の要素の値と一致しなければならないことを意味します。(パスワード、パスワードの確認など)。
	modifier	規則を使用する前に入力値を変更したり、検査を終えた後、結果を変更できる機能。

より詳細な内容については、「4 フォームの使用」を参照してください。

2.1.6 フォームフィルターを使用する

フィルターは、HTML フォームにて PHP の処理メソッドで情報を渡す JavaScript コールバック関数を指定するために使用します。 フィルターは、tpl/filter フォルダ内にある XML ファイルに保存されます。以下は、フォームフィルターの例題です。XE 1.5 以降からは、フォームフィルターよりルールセットを使用することを推奨します。

```
<filter name="insert_contest" module="contest" act="procContestAdminInsertContest" confirm_msg_code="confirm_submit">
    <form>
        <node target="mid" required="true" maxlength="40" filter="alpha_number" />
        <node target="browser_title" required="true" maxlength="250" />
    </form>
    <parameter>
        <param name="contest_name" target="mid" />
        <param name="module_srl" target="module_srl" />
        <param name="module_category_srl" target="module_category_srl" />
        <param name="layout_srl" target="layout_srl" />
        <param name="skin" target="skin" />
        <param name="browser_title" target="browser_title" />
        <param name="header_text" target="header_text" />
        <param name="footer_text" target="footer_text" />
    </parameter>
    <response callback_func="completeInsertContest">
        <tag name="error" />
        <tag name="message" />
        <tag name="module" />
        <tag name="act" />
        <tag name="page" />
        <tag name="module_srl" />
    </response>
</filter>
```

フォームフィルターに使用される要素と属性は、下記表のとおりです。

表 2-3 フォームフィルターで使用される属性

要素	属性	説明
form		入力値が有効であるか確認する最上位要素
node		HTML フォーム確認
	required	入力要素が必ず必要な値であるか設定します。required 属性値が true に設定された場合、その要素に値が入力されていない場合、alert が発生します。
	filter = "filter type"	フィルターで使用できるタイプは、email(email_address), userid(user_id)、url(homepage), korean, korean_number, alpha, number, alpha_number です。
	equalto = "target person"	equalto に入れた要素の値が、現在の要素の値と一致しなければならないことをあらわします(パスワード、パスワード確認など)。
	maxlength	最大の長さ
	minlength	最小の長さ
parameter		サーバーへ転送する際にフォーム要素名を変更したり、フォーム要素のうち開発者が parameter で作成した値のみをサーバーに転送したいときに使用します。基本的には、parameter を使用しない限り、すべてのフォーム要素をサーバーへ転送します。
param		再定義したり、サーバーへ転送するフォーム要素情報を作成します。
	name	フォーム要素名
	target	再定義する要素名
response		
	callback_func	JavaScriptコールバック関数。実際に実装しなければなりません。
tag		コールバック関数で渡される変数を定義します。
	name	コールバック関数に渡す変数名。この変数は、controller でアクションを実行した後、コールバック関数に渡す値を$this->add('変数名', '値')で実装します。

より詳細な内容は、「4 フォームの使用」を参照してください。

2.1.7 DB クエリ定義

XE は、カスタムクエリ言語を使用してクエリを定義します。XML コードは、./classes/xml フォルダにある XmlQueryParser.class.php にてパースします。使用例は、下記のとおりです。

```
<query id="getCounterStatus" action="select">
<tables>
<table name="counter_status" />
</tables>
<columns>
<column name="sum(unique_visitor)" alias="unique_visitor" />
<column name="sum(pageview)" alias="pageview" />
</columns>
<conditions>
<condition operation="more" column="regdate" var="start_date" notnull="notnull" pipe="and" />
<condition operation="less" column="regdate" var="end_date" notnull="notnull" pipe="and" />
</conditions>
</query>
```

2.2 アドオン

XE でのアドオンは、フック(hooking)を行います。フックとは、他の正常なアクションを取得する行為のことをいいます。フックは、PHP のようなインタープリター基盤言語で使用する「include」を使用します。XE では、アドオンを XE コンテキストにネイティブコードで挿入できるよう、関数やクラス形式では作成しません。このため XE のアドオンは呼び出された瞬間から強力な効果を発揮します。しかし、アドオンは、XE 全体の運営に負荷がかかる恐れがあるため、気をつけて生成しなければなりません。

アドオンを生成するには、次の規則に従わなければなりません。

- アドオンは、addons フォルダ配下の addon_name フォルダに保存しなければなりません。
- アドオン実行ファイルの名前は、addon_name.addon.php でなければなりません。
- info.xml ファイルに製作者情報、アドオン説明、管理者(必要な場合)から受け取ったアドオン変数を保存しなければなりません。

2.2.1 アドオン呼び出し時点

アドオンを呼び出す時点は、下記のとおりです。

- before_module_init: モジュールオブジェクト生成前: ユーザーが要求したモジュールを探した後、該当モジュールのオブジェクトを生成する前
- before_module_proc: モジュール実行前: モジュールのオブジェクトを初期化した後、該当モジュールを実行する前
- after_module_proc: モジュール実行後: 生成されたモジュールオブジェクトを実行して結果を得た直後
- before_display_content: 結果出力前: レイアウトの提供されたモジュールの結果を出力する直前

各フックがどのような役割をし、なぜ XE コントロールパス(control path)にて一部のアドオンが特定の時点だけを使用するのかを理解するために、下記にていくつか例を挙げます。

タグリスト – After module proc

すべての文書のタグリストを1ページに出力するアドオンがあったとします。このようなタグリストを生成するには、先に現在ページの module_srl が含まれた文書を取得しなければなりませんが、その前に module_srl を知っておかなければなりません。このため、after_module_proc というポイントを選択しなければなりません。モジュール情報を処理後に、以前に定義されたすべての作業を実行できます。

メタタグ – Before module proc

このアドオンは、メタ説明とキーワード、製作者などのメタタグをすべてのページに挿入する役割をします。 コンテンツがモジュール処理作業で生成される前にメタタグを挿入しなければならないため、before_module_proc ポイントをフックに使用します。

ポイントレベルアイコン – before display content

特定の会員が貯めたポイントレベルによって、各ユーザーにアイコンを表示するアドオンが必要です。このアドオンは、すでに処理された一部のパラメータによってコンテンツ内の HTML コードを修正しなければならないため、before_display_content ポイントをフックとして使用します。

カウンター– Before Module Init

このアドオンは、XE を使用して構築した Web サイトの訪問統計を確認するために開発されました。このアドオンは、カウンターモジュールを使用します。カウンターアドオンは、$is_logged 変数の情報を使用して Web サイト訪問回数を

計算します。モジュール処理作業からもう情報を得る必要がないため、このモジュールは、時間的に一番目のフックである before_module_init を使用します。

2.2.2 アドオン呼び出し時に渡される変数

前述した 4 番目の呼び出し時点で XE core は、次の共通変数をアドオンに渡します。

- $called_position: 呼び出し時間情報が含まれています。値は、before_module_init, before_module_proc, after_module_proc, before_display_content の4つのうち、ひとつです。
- $addon_path: 呼び出されたアドオンのパスを含みます。
- $addon_info: XE のアドオンは、独立して設定でき、該当アドオンが動作する対象モジュールを指定できます。$addon_info 変数は、アドオンで宣言された extra_vars(info.xml 内)の情報を含み、これはアドオンごとに異なります。

2.2.3 アドオンファイル作成

addons フォルダにいろんな名前のファイルを保存でき、フォルダ内でクラスを使用することもできます。しかし、関数宣言は、ネイティブコードで動作する include 構造を使用するため、許容されません。

config/Info.xml

info.xml ファイルは、以下のように作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<addon version="0.2">
 <title xml:lang="en">Addon title</title>
 <description xml:lang="en">Addon description</description>
 <version>Addon version</version>
 <date>Year-Month-Date</date>
 <author email_address="The email address of an author" link="The homepage address of an author">
 <name xml:lang="en">Author name</name>
 </author>
 <extra_vars>
 <var name="Variable name" type="textarea">
              <title xml:lang="en">Variable name (for output)</title>
              <description xml:lang="en">Variable description</description>
 </var>
 </extra_vars>
</addon>
```

必要であれば、extra_vars を生成します。詳細内容がなければ、"<extra_vars />"コマンドを使用して省略します。上記のように作成したファイルを info.xml という名前で conf フォルダ配下に保存します。

addon_name.addon.php

アドオンが、あるアクションを行うことになっている場合、PHP 形式でアドオンファイルを作成します。ただし、アドオンは、大概、クラスオブジェクトのメソッド内で呼び出されるため、関数は宣言できません。アドオン内でクラスを定義し、使用できます。

アドオンファイルの開始部は、下記のようにしなければなりません。

```
<?php

 /**
              * @file addon name.addon.php
              * @author author name (email address)
              * @brief description
              **/
              if(!defined('_ZBXE_')) exit();
```

XE のすべての機能は、index.php を通じて行われ、index.php は、_ZBXE_常数が true に設定されると開始されます。したがって、index.php に宣言されている_ZBXE_常数が true に設定されているか、機能を実行する前に確認します。アドオン実行時点は、called_position で確認でき、この作業は必ず該当アドオンで手動的に行わなければなりません。

例えば、ページ下にすべての文書のタグリストを出力するアドオンががあったとします。まず、アドオンを呼び出すには、どんなフックが適切であるか確認しなければなりません。文書のリストを獲得するには、以下のコードのようにページモジュールを処理して 'after_module_proc' を使用しなければなりません。

```
<?php
if(!defined("_ZBXE_")) exit();
/**
* @file tag_list.addon.php
* @author Author (author@authorland.com)
* @brief Description of the addon
**/

if($called_position != 'after_module_proc' || Context::getResponseMethod()!=='HTML') return;

$obj->module_srl=Context::get('module_srl');
$document_list=executeQueryArray('addons.tag_list.getModuleDocumentTags',$obj);
$tags='';
foreach ($document_list->data as $val) {
  $tags=$tags.','.$val->tags;
}
$tags=explode(',', $tags);
for($i=1;$i<count($tags);$i++) {
  $tags[$i]='<a href="'.getUrl('act','TS','is_keyword',$tags[$i]).'">'.$tags[$i].'</a>';
}
$tags=implode(' ', $tags);
$tags='<div class="tags" align="center">'.$tags.'</div>';
$content=Context::get('page_content');
$content=$content.$tags;
Context::set('page_content',$content);
?>
```

上記コードでは、現在見える HTML ページにタグリスト HTML コードを挿入しています。

2.2.4 XE XML クエリの使用方法

XE アドオンで、他のモジュールが生成した DB にあるデータは、XML クエリを通じて使用できます。この場合、addon フォルダ下に queries というフォルダをひとつ生成して XML クエリ文を定義した XML ファイルを保存します。クエリを実行する方法は、下記のとおりです。

```
$document_list=executeQueryArray('addons.tag_list.getModuleDocumentTags',$obj);
```

2.2.5 アドオン生成時の考慮事項

アドオン生成時の考慮事項は、下記のとおりです。

- XE のアドオンは、すべてのモジュールのいろんな部分に挿入されるため、<?php ... ?>前後に空白があってはいけません。空白が含まれると、before_display_content が呼び出されても正常に動作しません。
- XE core は、アドオンをプログラミングするときに発生し得る例外を別途処理しません。したがって現在の呼び出し状況を確認するルーチンをしっかり実装しておき、例外が発生しないようにしなければなりません。
- アドオンコードエラーによって Web サイトに深刻なエラーが発生したら、files/cache/activated_addons.cache.php ファイルを修正して再度アップロードします。

XE アドオンは、強力なアクションを行うことができます。しかし、コードを適切に作成しなかった場合、予期しない結果が発生したり、XE が中断されることもあります。したがって、アドオンを生成するときは、基本アドオンを参照することを推奨します。

2.3 ウィジェット

ウィジェットは、画面でデータを出力するために使用するコンポーネントです。ウィジェットは、最新記事と会員情報のような既存モジュールや外部 API から抽出したデータとともに連動できます。ウィジェットは、すべての種類のページに追加でき、レイアウトに直接追加することもできます。ウィジェットを通じて出力されるコンテンツをかんたんにカスタマイズできます。

ウィジェットは、管理者が手動でページモジュールに入力し、<img/>要素に保存します。出力する Web ページを呼び出すときに widgetController::triggerWidgetCompile()トリガーが widgetproc()を使用して<img/>要素のコードを実行し、正しい HTML コードに変換します。

2.3.1 config/info.xml 作成

info.xml ファイルは、ウィジェット製作者とバージョン、その他設定変数に関する情報を保存します。以下のように作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<widget version="0.2">
    <title xml:lang="en">Widget title</title>
    <description xml:lang="en">Widget description</description>
    <version>Widget version</version>
    <date>Widget creation date</date>
    <author email_address="..." link="...">
        <name xml:lang="en">Author name</name>
    </author>
    <extra_vars>
        <var id="extensionVariableName">
            <name xml:lang="en">Extension variable name</name>
            <type>Type of extension variable: text | textarea | select | select-multi-order | mid | mid-list | menu </type>
        </var>
    </extra_vars>
</widget>
```

2.3.2 ウィジェットクラスの開発

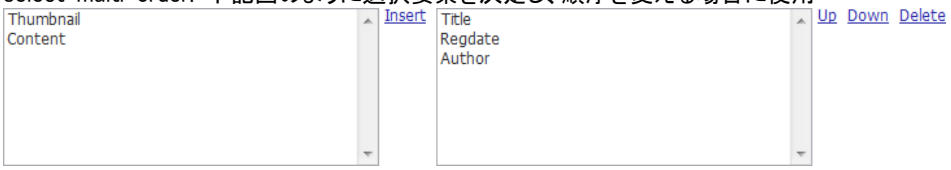
ウィジェットがどのような機能をするか、widgetName.class.php というクラスファイルに実装します。ウィジェットを実装するすべてのクラスは、WidgetHandler を継承して proc()メソッドを実装しなければなりません。

```
<?php
class myWidget extends WidgetHandler {
    function proc($args) {
                // .. Widget implementation ..

                // Template, specify the path of the skin (skin, colorset according to the value)
                $tpl_path = sprintf('%sskins/%s', $this->widget_path, $args-> skin);
                Context::set ('colorset', $args->colorset);

                // Template file name
                $tpl_file = 'HTML template file except the extension ';

                // Template compilation
                $oTemplate = &TemplateHandler::getInstance();
                return $oTemplate->compile($tpl_path, $tpl_file);
    }
}
?>
```

2.3.3 拡張変数の使用

拡張変数は、ウィジェットをページに挿入する直前にウィジェットの管理部からデータを取得するために使用します。ページで自動的に生成される各変数の値を得るために、変数別に入力タイプを設定できます。ウィジェットの拡張変数は、下記のとおりです。

- text: 一般文字列タイプ
- textarea: 段落を含む文字列タイプ
- select: 複数の内容のうち、ひとつを選択
- select-multi-order: 下記図のように選択要素を決定し、順序を変える場合に使用



- mid: モジュールをひとつだけ選択
- mid_list: モジュールを複数選択
- menu: サイトメニューのうち、ひとつを選択

3. DB 連動

この章では、XE と DB の連動方法について説明します。

3.1 概要

XE には、データベース-アグノスティック(database-agnostic)DB 抽象レイヤがあります。XE を様々な DBMS と共に使用でき、DBMS 間での切り替えも簡単に行えるという意味です。XE 1.5 は、MySQL、MS SQL、 CUBRID、PostgreSQL、SQLite3、Firebird をサポートします。

このために XE の XML スキーマ言語(XML Schema Language)と XML クエリ言語(XML Query Language)を使用して全体 DB スキーマとクエリを XML で作成します。

以下は、XML スキーマファイルの例題です。

```
# Excerpt from ./modules/member/schemas/member.xml
<table name="member">
    <column name="member_srl" type="number" size="11" notnull="notnull" primary_key="primary_key" />
    <column name="user_id" type="varchar" size="80" notnull="notnull" unique="unique_user_id" />
    <column name="find_account_question" type="number" size="11" />
    <column name="allow_mailing" type="char" size="1" default="Y" notnull="notnull" index="idx_allow_mailing" />
    <column name="limit_date" type="date" />
    <column name="regdate" type="date" index="idx_regdate" />
    <column name="description" type="text" />
    <column name="list_order" type="number" size="11" notnull="notnull" index="idx_list_order" />
</table>
```

XE の初回インストールの際に含まれているモジュールのうち、テーブル.xml があれば、自動的にテーブルが生成されます。XE をインストールした後、追加モジュールをインストールしたときにテーブル.xml があれば、管理者画面に**モジュールインストール**ボタンが表示されます。XML ファイルを通じてこのテーブルに対するクエリを生成できます。

```
#./modules/member/queries/getMemberInfo.xml
<query id="getMemberInfo" action="select">
    <tables>
        <table name="member" />
    </tables>
    <columns>
        <column name="*" />
    </columns>
    <conditions>
        <condition operation="equal" column="user_id" var="user_id" notnull="notnull" />
    </conditions>
</query>
```

このクエリを PHP で呼び出すコードは、下記のとおり、とてもかんたんです。

```
$args->user_id = $user_id;
$output = executeQuery('member.getMemberInfo', $args);
```

3.2 XML スキーマ言語リファレンス

XE の DB テーブルスキーマは、XML ファイルで定義されます。DB テーブルスキーマは、各モジュールの schemas フォルダに保存されます。

XML スキーマファイルは、ひとつのルート<table>要素とひとつ以上の子<column>要素で構成されます。各要素の属性は、下記のとおりです。

表 3-1 <table>要素の属性

属性	説明
name	生成されるテーブル名。接頭文字 xe_が、自動追加され、別途指定する必要がありません。 XML ファイル名と同名でなければなりません。

表 3-2 <column> 要素の属性

属性	説明
Name	カラム(column)名
Type	列が保存するデータタイプ。値は、次のうちのひとつ。 • number • bignumber • varchar • char • text • bigtext • date • float パーサーが、このデータタイプを各 DB のデータタイプに自動的にマッピングします。例えば、bignumber は、MySQL の bigint に該当します。 各データタイプを DB 別データタイプにマッピングする方法についての詳細は、「表 3-4 XE-DBMS 間データタイプマッピング」を参照してください。
size	列のサイズ。数字や文字タイプに使用されます。 • 数字タイプ: 正確度をあらわします。 • 文字タイプ: 該当文字列が含む文字の数をあらわします。
Default	列の基本値を指定
notnull	列が null 値を許容するかを指定。許容する場合は、この属性を省略し、許容しない場合は次のようにこの属性を追加します。 例題) notnull = "notnull"
primary_key	テーブルの primary key として使用する列(column)を指定。各列にて primary_key="primary_key"属性を指定すると、ふたつの属性を束ねて、primary_key になります。
index	列のインデックス生成。この属性の値は、生成されるインデックスの名前をあらわします。 インデックス名をひとつ以上の列に重複して使用すると、結合インデックスが生成されます。 例題) index="idx_list_order"
unique	列に対する固有のインデックスを生成します。この属性の値は、生成されるインデックスの名前

属性	説明
	をあらわします。
auto_increment	列の値が自動インクリメントするかを指定します。 例題) auto_increment=″auto_increment″

3.3 XML クエリ言語

XE は、様々な DB に対応するために SQL クエリをそのまま使用せず、XML で作成します。

3.3.1 使用方法

XML クエリは、モジュールとアドオン、ウィジェットなどで次のように使用できます。

```
$args->name = "zero";
$output = executeQuery("member.getMemberInfo", $args);
```

executeQuery()関数は、./classes/db/DB.classs.php にある DB::executeQuery()関数の別称(alias)です。この関数は、実際に DB データを操作し使用した DB によって XML クエリがネイティブ SQL でパースされた後に結果を受信します。

```
function executeQuery($xml_query_name, $args = null);
```

- 最初のパラメータは、実行される XML クエリの名前です。値は、「モジュール名.クエリ ID」です。
- 2 番目のパラメータは、stdClass のタイプで、データをクエリに渡す際に使用されます。このパラメータは、null になることがあります。
- 結果値は、Object クラスのオブジェクトで返されます。
  - $output->toBool()が FALSE であれば、クエリの失敗を意味し、$output->toBool()が TRUE であれば、クエリが正常に実行されたことを意味します。
  - select 文の結果は、$output->data 変数に入れ、返されます。

3.3.2 XML 要素

```
<query id="query_id" action="select|update|delete|insert">
 <tables>
            <table name="tableName" alias="alias" />
 </tables>
 <columns>
            <column name="columnName" alias="alias" />
 </columns>
 <conditions>
            <condition operation="doSomething" column="column1" var="variable" filter="filterType" default="default"
notnull="notnull" minlength="minimumLength" maxlength="maximumLength" pipe="TheConcatenationOperator " />
            <group pipe="pipe">
                        <condition operation="anotherOperation" column="column" var="variable" filter="filterType"
default="default" notnull="notnull" minlength ="minimumLength" maxlength="maximumLength"
pipe="TheConcatenationOperator"/>
            </group>
 </conditions>
 <navigation>
            <index var="var" default="default" order="desc|asc" />
            <list_count var="var" default="default" />
            <page_count var="var" default="default" />
            <page var="var" default="default" />
 </navigation>
 <groups>
            <group column="GroupBy daesang" />
 </groups>
 </query>
```

XML クエリに使用される XML 要素と属性は、下記表のとおりです。

表 3-3 XML クエリに使用される XML 要素と属性

要素	属性	説明
<query>		クエリ XML の最上位要素
	id	クエリ検索のための ID。module.query_id を使用してクエリ XML ファイルを検索して使用します。
	action	アクションは、select、update、delete、insert の 4 つのタイプです。
	alias	サブクエリを使用するときのクエリ文の alias 名です。
<tables>		クエリに使用されるテーブルの集まり
<table>		テーブル要素
	name	元のテーブル名(XE の接頭文字は無視)
	alias	列指定や検索などで使用するテーブルの別称
<columns>		クエリに使用される列の集まり
<column>		列要素
	name	列名
	alias	列の元の名前を変更するときに使用
<conditions>		条件文を作るときに使用。<group>要素を使用して条件文をいろんなグループで区別できます。
<group> ... </group>		条件文がグループで使用されるときに pipe="and|or"を使用してグループ間の条件を指定できます。
<condition>		条件文
	operation	次のような演算子で処理できます。 • equal : column = (var|default) • more : column >= (var|default) • excess : column > (var|default) • less : column <= (var|default) • below : column < (var|default) • notequal : column != (var|default) • notnull : column is not null • null : column is null • like_prefix : column like '%var|default' • like_tail : column like 'var|default%' • like : column like '%var|default%' • in : column in (var|default) • notin : column not in (var|default)
	column	列名を指定します。
	var	executeQuery(Array)()関数のふたつめののパラメータである、stdClass のキー値を指定します。
	filter	var 値の条件フィルタリング。対応するフィルターは、下記のとおりです。 • email, email_address: メール形式

要素	属性	説明
		• homepage: http|https:// 同じ Web サイトアドレス形式 • userid, user_id: XE のユーザーidj 形式(最初の 2 文字は、アルファベットでなければなりません。3 番目の文字から number+alphabet+ _形式にします) • number: 数字許容 • alpha: アルファベット許容 • alpha_number: 数字と文字すべて許容
	default	Var 値が null であれば基本値に。基本値は、一般文字列、数字をすべて使用可能であり、下記の関数を使用することもあります。 • ipaddress(): IP アドレス • unixtime(): ユニックス時間(php 内 time()関数) • curdate(): YYYYMMDDHHIISS • plus(int count): column = column + count • minus(int count): column = column − count • multiply(int arg): column = column * arg • sequence(): XE の getNextSequence()を実行
	notnull	null であるかどうか確認。指定した場合、必ず var 値がなくてはなりません。
	minlength	最小の長さを確認
	maxlength	最大の長さを確認
	pipe	and|or 同じ条件を指定
<navigation>		整列順やページングをサポート
<index>		整列される列と整列方式指定
	var	列名が値の変数名
	default	var 値が指定されていない場合に使用する基本整列されている列名
	order	整列方式。 asc|desc でない変数名を使用すると、変数の値によって整列されます。ただし、変数値は asc|desc で渡さなければなりません(昇順 "asc"、降順 "desc")
<list_count>		ページング結果を受信できるようにします。
	var	行の数が値の変数名
	default	var 値が指定されていない場合に使用する行数の基本値
<page_count>		ページングを計算するときのナビゲーション数指定
	var	ページングナビゲーションの数が値の変数名
	default	var 値が指定されていない場合に使用する基本ページングナビゲーションの数
<page>		現在ページ番号指定
	var	現在ページ番号を値として持つ変数名
	default	var 値が指定されていない場合に使用する基本ページ番号
<groups>		条件文によってグループを使用できるようにします。 group by 節の使用時に作成
	column	group by 基準列の名前

3.3.3 XML サブクエリの使用例題

XE 1.5 バージョンからサブクエリを作成できます。下記にて、サブクエリタイプ別の作成例を示します。

Select 節使用

SQL 使用の例

```
select *,
(select count(*) as "count"
            from "xe_documents" as "documents"
              where "documents"."user_id" = "member"."user_id"
        ) as "totaldocumentcount"
  from "xe_member" as "member"
  where "user_id" = 7
```

XML サブクエリ作成の例

```
<query id="getStatistics" action="select">
    <tables>
        <table name="member" alias="member" />
    </tables>
    <columns>
        <column name="*" />
        <query id="getMemberDocumentCount" alias="totalDocumentCount">
            <tables>
                    <table name="documents" alias="documents" />
            </tables>
            <columns>
                    <column name="count(*)" alias="count" />
            </columns>
                <conditions>
                    <condition operation="equal" column="documents.user_id" default="member.user_id" />
                </conditions>
        </query>
    </columns>
    <conditions>
        <condition operation="equal" column="user_id" var="user_id" notnull="notnull" />
    </conditions>
</query>
```

Where 節使用

SQL 使用の例

```
SELECT *
FROM xe_member as member
WHERE regdate = (SELECT MAX(regdate) as regdate
                                    FROM xe_documents as documents
                                    WHERE documents.user_id = member.user_id)
```

XML サブクエリ作成の例

```
<query id="getMemberInfo" action="select">
    <tables>
        <table name="member" alias="member" />
    </tables>
    <columns>
        <column name="*" />
    </columns>
    <conditions>
        <query operation="equal" column="regdate" notnull="notnull" alias="documentMaxRegdate">
            <tables>
                    <table name="documents" alias="documents" />
            </tables>
            <columns>
```

```
                    <column name="max(regdate)" alias="maxregdate" />
            </columns>
            <conditions>
                    <condition operation="equal" column="documents.user_id" var="member.user_id" notnull="notnull" />
            </conditions>
        </query>
    </conditions>
</query>
```

From 節使用

SQL 使用の例

```
SELECT m.member_srl, m.nickname, m.regdate, a.count
FROM (
 SELECT documents.member_srl as member_srl, count(*) as count
 FROM xe_documents as documents
 GROUP BY documents.member_srl) a
            INNER JOIN xe_members m on m.member_srl = a.member_srl
```

XML サブクエリ作成の例

```
<query id="getMemberInfo" action="select">
    <tables>
        <table query="true" alias="a">
            <table>
                    <table name="documents" alias="documents" />
            </table>
            <columns>
                    <column name="member_srl" alias="member_srl" />
                    <column name="count(*)" alias="count" />
            </columns>
                <groups>
                    <group column="member_srl" />
                </groups>
        </table>
        <table name="member" alias="m" type="inner join">
            <conditions>
                <condition operation="equal" column="m.member" default="a.member_srl" />
            </conditions>
        </table>
    </tables>
    <columns>
        <column name="m.member_srl" />
        <column name="m.nickname" />
        <column name="m.regdate" />
        <column name="a.count" />
    </columns>
</query>
```

3.4 データタイプマッピング

XE と各 DBMS のデータタイプは、下記のようにマッピングされます。

表 3-4 XE-DBMS 間データタイプマッピング

XE	MySQL	CUBRID	MS SQL
number	Bigint	Integer	int
bignumber	Bigint	numeric(20)	bigint
varchar	varchar	character varying	varchar
char	char	Character	char
text	text	character varying(1073741823)	text
bigtext	longtext	character varying(1073741823)	text
date	varchar(14)	character varying(14)	varchar(14)
float	float	Float	float
tinytext		character varying(256)	

3.5 XML Query Parser

XML Query Parser クラスは、XML クエリファイルの入力を受けパースした後、SQL クエリ(select, update, insert, delete のようなクエリタイプ、使用されている表現式、連結/フィルタリング条件、条件文によるグループ/順序)を生成するために必要なすべての情報を関連クラスオブジェクト形式で含める PHP ファイルを生成します。この PHP ファイルは、各 DB クラスの入力値で使用され、各 DB クラスは DBMS 別に適切なエスケープ文字とカスタム言語構造を使用して SQL を生成します。

例えば、次のような XML クエリがあったとします。

```
# ./modules/document/queries/getCategory.xml
<query id="getCategory" action="select">
    <tables>
        <table name="document_categories" />
    </tables>
    <conditions>
        <condition operation="equal" column="category_srl" var="category_srl" filter="number" notnull="notnull" />
    </conditions>
</query>
```

executeQuery 関数を使用してこのクエリを呼び出すと、XE はパースした結果を含む PHP 形式のキャッシュファイルが生成されているか確認します。XML Query Parser クラスを呼び出していなかったら、PHP ファイルを 生成して./files/cache/queries に保存します。

```
# ./files/cache/queries/document.getCategory.1.5.0.8.cache.php
<?php if(!defined('_ZBXE_')) exit();
$query = new Query();
$query->setQueryId("getCategory");
$query->setAction("select");
$query->setPriority("");

$category_srl1_argument = new ConditionArgument('category_srl', $args->category_srl, 'equal');
$category_srl1_argument->checkFilter('number');
$category_srl1_argument->checkNotNull();
$category_srl1_argument->createConditionValue();
if(!$category_srl1_argument->isValid()) return $category_srl1_argument->getErrorMessage();
if($category_srl1_argument !== null) $category_srl1_argument->setColumnType('number');

$query->setColumns(array(
new StarExpression()
));
$query->setTables(array(
new Table('`testtesttest_document_categories`', '`document_categories`')
));
$query->setConditions(array(
new ConditionGroup(array(
new ConditionWithArgument('`category_srl`',$category_srl1_argument,"equal")))
));
$query->setGroups(array());
$query->setOrder(array());
$query->setLimit();
return $query; ?>
```

次に、DB 別の executeQuery メソッドが呼び出され、上記ファイルの出力値がこのメソッドの入力に使用されます。DB クラスは、SQL クエリを生成して実行します。たとえば、上記のクエリは、下記の SQL クエリになります。

```
select * from "xe_document_categories" as "document_categories" where ("category_srl" = 15)
```

cache.php ファイルもまた、列(column)タイプに関する情報を含めています。この情報は、テーブルスキーマファイルで抽出します。XE は、先に./modules/<module_name>/schemas/<table_name>内で該当スキーマファイルを探します。探すファイルがなければ、<table_name>というファイルを探すまで各モジュールを検索します。

3.6 XE DB クラス

XE は、サポートするすべての DBMS に対し、カスタムクラスを提供します。カスタムクラスは、DBMS 別にカスタム SQL 文法を生成します。

例えば、XE core と共に使用されるクラスは、下記のとおりです。

```
DB.class.php
DBMysql.class.php
DBCubrid.class.php
DBMssql.class.php
….
```

すべてのカスタムクラスは、./classes/db に保存されます。

すべてのカスタム DB クラスは、共通 DB クラスを継承します。コードを作成するときに一般 DB クラスを使用すると、XE がどの DB クラス実装体を使用するかを、実行時間に決定します。

4. フォームの使用

この章では、フォームを使用する方法について説明します。

4.1 概要

フォームは、ユーザー入力値をサーバーに転送する際に使用されます。XE ではフォームを転送するときに入力値の有効性をチェックするためのルールセット機能を提供します。XE のルールセット機能を使用すると、入力値の有効性確認スクリプトを別途製作する必要がありません。

フォーム別に次の項目が必要です。

- フォームのマークアップと設計
- フォーム転送時に呼び出されるサーバー側メソッド

このようなフォーム転送のために連動する XE の構成要素は、下記のとおりです。

- フォームテンプレートファイル: フォームのレイアウトとフィールドの定義
- フォーム転送を処理する controller メソッド(controller ファイル内)
- フォームの有効性チェックのためのルールセット XML ファイル

4.2 XE フォームの作成

ユーザー名を問い、入力を受けた後、ウェルカムメッセージを表示するページを表示する、というとてもかんたんなフォームを作ってみましょう。モジュールには、ひとつのビューしかありません。このビューは、ユーザーが名前を入力した後、hello メッセージを表示するか、または名前を入力するフォームを再度出力します。

この例題モジュールの完成バージョンは、hello.zip からダウンロードできます。ここで説明するチュートリアルに沿って、モジュールを完成するには、開始ファイル(hello-tutorial.zip)のみダウンロードしてください。

4.2.1 フォームビュー生成

先に、入力ボックスと登録ボタンのみのフォームを作成してみます。ファイル名(name.html)を決め、./modules/hello/tpl/に保存します。

```
<h1>Enter your name:</h1>
<form id="name_form" action="./" method="post" ruleset="say_hello">
        <input type="hidden" name="module" value="hello" />
        <input type="hidden" name="act" value="procHelloGreet" />
        <input type="text" name="name" id="name" value="" />
        <br />

        <input type="submit" value="OK" />
</form>
```

XE でフォームを転送するには、基本的にどのモジュールのどのアクションでデータを転送するかを追加し、データ有効性チェックのためにルールセットを指定しなければなりません。上記の例題では、hello モジュールの procHelloGreet アクションにデータを転送し、say_hello というルールセットファイルで有効性をチェックするように設定しました。

> **参考**
>
> formタグのruleset属性は、適用するルールセットファイルのファイル名です。該当属性値の前に@をつけると、XEで動的に生成されるルールセットを参照せよという意味になります。例えば、XE 1.5では、ログインアカウントとしてuser_id、またはemail_addressを選択するようになっていますが、このとき、設定値によってログインデータの有効性チェック方法が異なるため、動的ルールセットを適用します。動的ルールセットファイルは、files/rulsetに保存されます。

テンプレートファイルを出力するビューメソッドを生成します。./modules/hello/hello.view.php に次のメソッドを追加します。

```
        /**
         * @brief Display form for entering a name
         **/
        function dispHelloName() {
            $this->setTemplateFile('name');
        }
```

./modules/hello/conf/module.xml に次のように追加してエンドユーザーが使用できるようにします。

```
<?xml version="1.0" encoding="utf-8"?>
<module>
    <grants />
    <permissions />
    <actions>
        <action name="dispHelloName" type="view" standalone="true" index="true" />
    </actions>
</module>
```

これで、/?module=hello にアクセスすると、下記のフォームを確認することができます。

図 4-1 名前入力フォーム

4.2.2 XML ルールセットファイルとコントローラアクションの追加

先ほど作ったフォームは、まだなんの機能もしない状態ですが、このフォームに、ユーザー名を照会し、hello メッセージを出力するメソッドを追加してみます。

./modules/hello/hello.controller.php に以下のようにメソッドを追加します。ルールセットファイルを使用するときには、アクション実行後に移動するアクションを明示しなければなりません。procHelloGreet 関数の実行が完了したら、dispHelloName 画面へ移動するよう、下記のように setRedirectUrl を設定します。

```
/**
 * Action for handling the name input form submission
 * Retrieves the name given by the user and passes it on for displaying the greeting screen
 */
function procHelloGreet(){
        $name = Context::get('name');
        $this->setRedirectUrl(getNotEncodedUrl('', 'module', 'hello', 'act', 'dispHelloName', 'name', $name));
}
```

./modules/hello/conf/module.xml の<actions>要素に次のように追加します。

```
<action name="procHelloGreet" type="controller" standalone="true" />
```

フォームの内容の有効性をチェックするためには、XML ルールセットファイルを追加しなければなりません。ファイル名を say_hello.xml にし、./modules/hello/ruleset/配下に保存します。

```
<?xml version="1.0" encoding="utf-8"?>
<ruleset version="1.5.0">
    <fields>
        <field name="name" required="true" />
    </fields>
</ruleset>
```

ルールセットファイルの要素と属性についての詳細は、「2.1.5 ルールセットを使用する」を参照してください。

4.2.3 ウェルカムメッセージの出力

./modules/hello/hello.view.php にある dispHelloName メソッドを次のように修正します。

```
    /**
     * @brief Display form for entering a name
     **/
    function dispHelloName() {
            $name = Context::get('name');
            if(isset($name)){
                    $hello_message = "Hello " . $name;
```

```
                                    Context::set('hello_message', $hello_message);
                            }
        $this->setTemplateFile('name');
         }
```

テンプレートファイル(./modules/hello/tpl/name.html)も次のように修正します。

```
<h1 cond="isset($hello_message)">{$hello_message}</h1>
<block cond="!isset($hello_message)">
    <h1>Enter your name:</h1>
    <form id="name_form" action="./" method="post" ruleset="say_hello">
        <input type="hidden" name="module" value="hello" />
        <input type="hidden" name="act" value="procHelloGreet" />
        <input type="text" name="name" id="name" value="" />
        <br />

        <input type="submit" value="OK" />
    </form>
</block>
```

ブラウザで該当ページを再度読み込むと、下記図のようにウェルカムメッセージが出力されます。

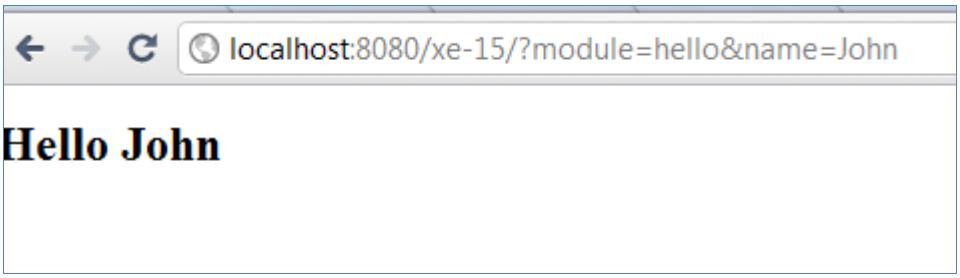


図 4-2 ウェルカムメッセージの出力

これで、フォームの完成です。

5. document モジュールの使用

この章では、XE で基本提供する document モジュールを使用する方法について説明します。

5.1 概要

XE は、モジュール式の構造になっていて、すでに作成されているモジュールを使用して XE core の機能をかんたんに拡張することができます。コンテンツ関連の追加機能を実装するときに最も重要な要素は、document モジュールです。document モジュールで提供する機能は、下記のとおりです。

- コンテンツを生成して照会する機能
- コメント数と閲覧数、その他有用な統計に関する情報
- 修正履歴
- カテゴリやタグを通じてコンテンツをかんたんに構成する機能
- バッチ(batch)編集
- XE の別のモジュールとかんたんに統合

このモジュールを活用する方法についてもっと知りたい場合は、コンテンツを保存する際に document モジュールを使用するフォーラム、ウィキ、textyle、Issue トラッカーなどのモジュールを参考にしてください。

5.2 document モジュールの作成

5.2.1 文書生成

文書生成メソッドは、documentController – ./modules/document/document.controller.php に定義されています。例題は、下記のとおりです。

```
$obj->title = "My sample document";
$obj->content = "Hello World!";
$obj->tags = "demo, hello";
$document_srl = getNextSequence();
$obj->document_srl = $document_srl;
$obj->module_srl = $this->module_srl;
$obj->allow_comment = 'Y';
$obj->allow_trackback = 'Y';
$oDocumentController = &getController('document');
$output = $oDocumentController->insertDocument($obj);
```

すべての文書は、[DB フラグ]_documents テーブルに保存されます。基本フィールド以外にも、extra_var 機能を使用して必要なカスタムフィールドをかんたんに追加することができます。extra_vars は、モジュールインスタンスを基準に生成されます。したがって、このモジュールインスタンスに含まれているすべての文書は、extra_vars で定義されているフィールドを使用できます。

カスタムフィールドの名前とタイプ情報は、[DB フラグ]_document_extra_keys テーブルに保存されます。documentController にある insertDocumentExtraKey メソッドを使用して新しいキーを追加できます。新しいキーの値は、[DB フラグ]_document_extra_vars テーブルに保存されます。documentController クラスの insertDocumentExtraVar メソッドを使用して新しいキーの値を追加できます。

5.2.2 文書属性

使用する文書属性は、下記のとおりです。

表 5-1 文書属性

属性	説明
document_srl	文書固有の ID
module_srl	文書のつながるモジュールインスタンス
category_srl	文書カテゴリの ID。文書カテゴリは、[DB フラグ]_document_categories テーブルに保存されます。
lang_code	文書の言語コード。同じ文書を他の言語の様々なバージョンで製作し、使用します。
is_notice	文書に重要表示をするときに使用する属性。例えば、文書リストの一番上にお知らせを表示するときにこの属性を使用します。
title	文書タイトル
content	文書内容
readed_count	文書を閲覧した回数
voted_count	文書の推薦回数。この属性は、ポイントモジュールと統合して実装します。
blamed_count	文書の申告回数
comment_count	文書につながっているコメントの数
trackback_count	文書のトラックバック数

属性	説明
uploaded_count	文書の添付回数
password	秘密文書に使用。非会員が記事を作成するときにパスワードを保存し、記事を修正したり、削除するときのみ使用します。秘密記事は、記事閲覧で使用されます。
user_id, user_name, nick_name, member_srl	文書所有者に関する情報
tags	文書タグ。値をコンマ(,)で区切って保存します。
regdate	文書の生成された日付
last_updated	文書が最後に修正された日付
ipaddress	文書を生成したユーザーの IP アドレス
comment_status	文書へのコメント許可(ALLOW: 許可する、DENY: 制限する)
status	文書の状態値(PRIVATE: 非公開、PUBLIC: 公開、SECRET: 秘密、 TEMP: 一時保存)

この属性は、すべて[DB フラグ]_documents テーブルのフィールドをあらわします。文書項目のモデルクラスは、document.item.php です。

5.2.3 文書 URL

文書は、様々な方法でアクセスできます。

まず、次の構造で不変アドレス(permalink)を表示します。

```
http://<xe_name>/<document_srl>
```

XE のすべての文書は、次のようにユーザーフレンドリーな名前でアクセスすることもできます。

```
http://<xe_name>/entry/<document_title>
```

文書タイトルが長すぎたり、空白を含めている場合、管理者コントロールパネルの **情報管理** > **文書**で文書の別称を定義することもできます。また、ひとつの文書にひとつ以上の別称を付けることができます。別称でドキュメントにアクセスする際の URL 構造は、下記のとおりです。

```
http://<xe_name>/entry/<alias>
```

上記のように内蔵されている文書アクセス方法以外にも、カスタムモジュールで自分だけのビューメソッドを定義することもできます。

> **参考**
> 上記の例題は、XE をインストールする際に mod_rewrite ができるように設定していないと使用できません。

5.2.4 文書カテゴリ

各文書は、カテゴリに含むことができます。カテゴリは、[DB フラグ]_document_categories テーブルに保存され、階層構造で作成できますが、基本的には、非階層構造になっています。

カテゴリは、documentController と documentModel クラスを使用して管理します。documentController クラスは、カテゴリ管理と関連した次のメソッドを含みます。

- insertCategory

- deleteCategory
- moveCategoryUp
- moveCategoryDown
- procDocumentMoveCategory
- updateCategory
- updateCategoryCount

documentModel クラスは、カテゴリ管理と関連した次のメソッドを含みます。

- getCategory
- getCategoryChildCount
- getCategoryDocumentCount
- getCategoryHTML
- getCategoryList
- getDocumentCategories
- getCategoryTplInfo

5.2.5 文書改定履歴

document モジュールは、文書の改定履歴を維持するメカニズムを持っています。文書が修正されるたびに documentController クラスの updateDocument メソッドがログエントリーを自動的に追加します。

改定履歴は、基本的に無効になっています。改定履歴を有効にするためには、文書部分設定ページで**ヒストリー使用**オプションを選択しなければなりません。

改定履歴は、[DB フラグ]_document_histories テーブルに保存されます。documentModel クラスにある次のメソッドを使用して文書のログを照会できます。

- getHistories
- getHistory

5.2.6 文書照会

文書を照会するときに使用されるメソッドは、documentModel の getDocumentList です。このメソッドを使用して次の基準で文書をフィルタリングできます。

- モジュール srl
- カテゴリ
- 文書を作成した会員
- タイトル
- 内容
- タグ
- タイプ – お知らせ、パスワード
- 閲覧数、推薦数など
- 作成日、修正日

6. API リファレンス

この章では、XE のグローバル関数と各クラス別関数について説明します。

6.1 XE のグローバル関数

XE のグローバル関数は、XE_ROOT/config/func.inc.php ファイルに定義されています。

debugPrint(mixed OBJECT)

デバッグ関数。

_DEBUG_の値は、[XE_ROOT]/config/config.inc.php ファイルに、1 以上で定義されなければなりません。
_DEBUG_OUTPUT_値によって結果値を獲得する方法を選択できます。

- 0: 出力するファイル/_debug_message.php につなげる
- 1: HTML フッターにコメントで出力(応答タイプが HTML の場合)
- 2: Firebug コンソールに出力(PHP >= 5.2.0. Firebug/FirePHP プラグイン必要)

instance getController(string MODULE_NAME)

モジュールの Controller インスタンスを取得します。

```
// If you want to get the document.controller.class instance
$oDocumentController = &getController('document');
```

instance getAdminController(string MODULE_NAME)

モジュールの Admin Controller インスタンスを取得します。

```
// If you want to get the documentAdminController instance
$oDocumentAdminController = &getAdminController('document');
```

instance getView(string MODULE_NAME)

モジュールの View インスタンスを取得します。

```
// If you want to get the rssView instance
$oRssView = &getView('rss');
```

instance getAdminView(string MODULE_NAME)

Admin View インスタンス関数を取得します。

```
// If you want to get the adminAdminView instance
$oAdminAdminView = &getAdminView('admin');
```

instance getModel(string MODULE_NAME)

モジュールの Model インスタンスを取得します。

```
// If you want to get the documentModel instance
$oDocumentModel = &getModel('document');
```

instance getAdminModel(string MODULE_NAME)

モジュールの Admin Model インスタンスを取得します。

```
// If you want to get the documentAdminModel instance
$oDocumentAdminModel = &getAdminModel('document');
```

instance getAPI(string MODULE_NAME)

モジュールの API インスタンスを取得します。

```
// If you want to get the boardAPI instance
$oBoardAPI = &getAPI('board');
```

instance getWAP(string MODULE_NAME)

モジュールの WAP インスタンスを取得します。

```
// If you want to get the boardWAP instance
$oBoardWAP = &getWAP('board');
```

instance getClass(string MODULE_NAME)

モジュールのクラスインスタンスを取得します。

```
// If you want to get the documentClass instance
$oDocumentClass = &getClass('document');
```

Object executeQuery(string QUERY_ID, stdClass PARAM)

XML クエリを実行します。結果データは、Object クラスのインスタンスで返されます。Object::toBool()が FALSE の場合は、クエリに失敗したことをあらわし、TRUE の場合は、クエリが正常に実行されたことをあらわします。

select 文の結果データは、Object::data 変数に入れ、オブジェクトに返されます。

Object executeQueryArray(string QUERY_ID, stdClass PARAM)

executeQuery()のような機能をしますが、Object:data 変数の結果が一行であっても配列で返します。

int getNextSequence()

次のシーケンス番号を取得します。

XE は、内部的にひとつのシーケンスを使用し、member_srl、module_srl、document_srl のようなすべての primary_key は、この関数を使用して設定します。すなわち、[DB フラグ]_documents テーブルで document_srl を 1 ずつインクリメント(auto increment)せずに、このシーケンス番号を使用します。

string getUrl(['',] string KEY, string VALUE [,string KEY, string VALUE ...])

URL を生成します。

XE は、現在の要求 URL で与えられたパラメータの値を変更した後、新しい URL を返します。もし最初のパラメータが ''の場合、XE は与えられたパラメータ値だけを使用して新しい URL を生成します。

```
// domain : www.example.com
// xe install path : /xe
// request url : www.example.com/xe/index.php?module=sample&act=dispSampleAct

$reset_url = getUrl('', 'module', 'reset');
print_r($reset_url);
// result : /xe/index.php?module=reset

$update_url = getUrl('module', 'update');
print_r($update_url);
// result : /xe/index.php?module=update&amp;act=dispSampleAct
```

string getFullUrl(['',] string KEY, string VALUE [,string KEY, string VALUE ...])

http://で始まる URL を生成します。

```
// domain : www.example.com
// xe install path : /xe
// request url : www.example.com/xe/index.php?module=sample&act=dispSampleAct

$reset_url = getFullUrl('', 'module', 'reset', 'mid', 'samplemid');
print_r($reset_url);
// result : http://www.example.com/xe/index.php?module=reset&amp;mid=samplemid
```

string getNotEncodedFullUrl(['',] string KEY, string VALUE [,string KEY, string VALUE ...])

URL エンコードされていない URL を生成します。getFullUrl()のような機能をします。

```
// domain : www.example.com
// xe install path : /xe
// request url : www.example.com/xe/index.php?module=sample&act=dispSampleAct

$reset_url = getNotEncodedFullUrl('', 'module', 'reset', 'mid', 'samplemid');
print_r($reset_url);
// result : http://www.example.com/xe/index.php?module=reset&mid=samplemid
```

string getAutoEncodedUrl(['' ,], string KEY, string VALUE [,string KEY, string VALUE …])

すでにエンコードされている場合、重複してエンコードされないように URL を生成します。

```
// domain : www.example.com
// xe install path : /xe
// request url : www.example.com/xe/index.php?module=sample&act=dispSampleAct

$reset_url = getAutoEncodedUrl('', 'name', '<script>', 'title', '&lt;title');
print_r($reset_url);
// result : http://www.example.com/xe/index.php?name=&lt;script&rt;&amp;title=&lt;title
```

string getSiteUrl(string DOMAIN, ['',] string KEY, string VALUE [,string KEY, string VALUE ...])

仮想サイト URL を生成します。最初の媒介変数は、ドメイン、もしくは vid を使用します。

```
// domain : www.example.com
// xe install path : /xe
// request url : www.example.com/xe/index.php?module=sample&act=dispSampleAct

$reset_url = getSiteUrl('site_id', '', 'module', 'reset');
print_r($reset_url);
// result : http://www.example.com/xe/index.php?module=reset&vid=site_id
```

string getNotEncodedSiteUrl(string DOMAIN, ['',] string KEY, string VALUE[,string KEY, string VALUE...])

エンコードされていない URL を生成します。getSiteUrl()のような機能をします。

string getFullSiteUrl(string DOMAIN, ['',] string KEY, string VALUE [,string KEY, string VALUE ...])

仮想サイトに対し http://で始まる URL を生成します。

int ztime(string STR)

YYYYMMDDHHIISS 形式の時間値をユニックス時間に変更します。

string getTimeGap(string DATE, string FORMAT)

YYYYMMDDHHIISS 形式の時間値を現在時間との差異(分/時)で表示します。時間差が一日以上であれば、FORMAT に設定した形式で表示します。

string getMonthName(int MONTH, bool SHORT)

月名を表示します。

```
print_r(getMonthName(3, true));
// result : Mar

print_r(getMonthName(10, false));
// result : October
```

string zdate(string STR, string FORMAT, bool CONVERSION)

YYYYMMDDHHIISS 形式の時間値を希望する時間形式に変更します。

```
print_r(zdate('19830310123644', 'Y-m-d H:i:s'));
// result : 1983-03-10 12:36:44
```

string cut_str(string STRING, int CUT_SIZE, string TAIL)

文字列を特定のサイズに切り取り、文字列の後ろに末尾文字(tail)を追加します。

```
print_r(cut_str('All roads lead to XE', 3, '…'));
// result : All…
```

string removeHackTag(string CONTENT)

ハック疑いのあるコードを削除します。

bool isCrawler(string AGENT)

ログインユーザーエージェントと IP を検査してクローラであるか確認します。

6.2 Context クラス

Context は、GET/POST の値を受信して変数と様々な情報をテンプレートに渡します。また、要求が XMLRPC、JSON、GET/POST のうち、どれにあたるかを識別します。

Context::set(string KEY, mixed VALUE)

テンプレートに渡される変数を設定します。

```
Context::set('user_id','user');
```

テンプレートでは、{$user_id}で渡された値を出力できます。

mixed Context::get(string KEY)

リクエスト(Request)に渡される変数や設定結果値を照会します。

```
$user_id = Context::get('user_id');
```

stdClass Context::gets(string KEY1 [, string KEY2 ...])

複数の値を一度に照会し、stdClass に返します。

stdClass Context::getRequestVars()

リクエストから渡された変数を stdClass で返します。

Context::addJsFile(string FILE_PATH, bool OPTIMIZED ,string TARGETIE, int INDEX)

JS ファイルをテンプレートに追加します。拡張子が js のファイルのみ追加されます。

Context::addCSSFile(string FILE_PATH, bool OPTIMIZED ,string TARGETIE, int INDEX)

CSS ファイルをテンプレートに追加します。

Context::addJsFliter(string FILTER_NAME)

XML で作成されたフィルターをテンプレートに読み込みます。

Context::setBrowserTitle(string TITLE)

HTML のタイトルを指定します。

Context::loadJavascriptPlugin(string PLUGIN_NAME)

JavaScript プラグインをテンプレートに読み込みます。

Context::addHtmlHeader(string HEAD)

HTML の<head>と</head>の間に文字列を追加します。

6.3 Extravar クラス

Extravar クラスは、通常、拡張変数と掲示板などのモジュールに使用されます。

ExtraItem::setValue(string VALUE)

拡張変数の値を指定します。

ExtraItem::getValueHTML()

拡張変数のタイプによって HTML に適したマークアップされた状態で拡張変数の値を返します。

ExtraItem::getFormHTML()

拡張変数のタイプによって HTML 結果ファイルの入力フォームを出力します。

6.4 Mail クラス

Mail クラスは、XE において、メール転送を担当します。XE では、サーバーがメールを転送できるように設定されているときのみメールを転送することができます。

Mail::setSender(string NAME, string EMAIL)

メールの発信者を指定します。

Mail::getSender()

Mail::setSender()関数で指定した発信者を返します。

- 発信者は、base64 でエンコードし、発信者名があれば返します。
- 発信者名がなければ空の文字列('')を返します。

Mail::setReceiptor(string NAME, string EMAIL)

メールの受信者を指定します。

Mail::getReceiptor()

Mail::setReceiptor()関数で指定した受信者を返します。

- 受信者は、base64 でエンコードし、受信者名があれば、返します。
- 受信者名がなければ、空の文字列('')を返します。

Mail::setTitle(string TITLE)

メールのタイトルを指定します。

Mail::getTitle()

base64 でエンコードされているメールのタイトルを返します。

Mail::setContent(string CONTENT)

メールの本文を指定します。

Mail::replaceResourceRealPath(mixed MATCHES)

本文に含まれている画像のアドレスを絶対パスに変換します。

Mail::getPlainContent()

メール本文をテキストで返します。

Mail::getHTMLContent()

メール本文を HTML 形式で返します。

Mail::setContentType(string MODE)

メール本文の形式を指定します。基本値は、HTML 形式です。

Mail::send()

メールを転送します。メールを転送する前に Mail::setSender()、Mail::setReceiptor()、Mail::setContent()関数を使用して発信者、受信者、メール本文を指定しなければなりません。

Mail::checkMailMX(string EMAIL_ADDRESS)

メールアドレスが有効であるか検査します。メールアドレスが正しくなければ、false を返します。

Mail::isVaildMailAddress(string EMAIL_ADDRESS)

メールアドレスが有効であるか正規表現式で素早く確認します。メールアドレスが有効であれば、渡された変数を変更せずにそのまま返します。

6.5 Object クラス

Object クラスは、モジュール間のデータを受け渡しする際に使用します。モジュールは、Object クラスを継承し error と message、variables 変数を利用して値と状態を交換します。

Object::Object([int ERROR, string MESSAGE])

Object 生成子。

- ERROR: エラーコード(この値が、0 であれば、エラーではない)
- MESSAGE: エラーメッセージ(この値が、success であれば、エラーではない)

bool Object::toBool()

Object がエラーであるか確認します。返し値が、true であれば該当オブジェクトはエラーではありません。

```
$output = executeQuery('document.insertDocument', $obj);
if(!$output->toBool()) {
        $oDB->rollback();
        return $output;
}
```

Object::add(string KEY, mixed VALUE)

KEY をキーにして変数を Object に追加します。

Object::adds(stdClass OBJECT)

渡された stdClass に属するすべての変数を Object に追加します。

```
$oObj = new Object();
$params->key1 = "value1";
$params->key2 = "value1";
$oObj->adds($obj);
```

mixed Object::get(string KEY)

Object の変数のうち、キーが KEY である変数のみ返します。

stdClass Object::gets(string KEY[, string KEY , ...])

Object の変数のうち、キーの値が KEY で宣言された変数を stdClass で括って返します。

```
$obj = $oObj->gets('key1','key2','key3');
// $obj->key1, $obj->key2, $obj->key3
```

6.6 FileHandler クラス

このクラスは、フォルダとファイルを扱うためのメソッドが含まれています。

FileHandler::copyDir(string SOURCE_DIR, string TARGET_DIR [, string FILTER] [, string TYPE])

SOURCE_DIR から TARGET_DIR へフォルダをコピーします。

- FILTER: 正規表現式を使用して、フォルダ内の下位フォルダとファイルをコピーするときに一致するファイルはコピーされません。
- TYPE: オプションが 'force' であれば、下位フォルダにある重複されたファイルをすべて上書きします。

FileHandler::copyFile(string SOURCE _FILE, string TARGET_FILE [, string FORCE])

SOURCE_FILE から TARGET_FILE へファイルをコピーします。

- FORCE: オプションが 'Y' であれば、重複されたファイルをすべて上書きします。

string FileHandler::readFile(string FILE_NAME)

ファイルの内容を読み込んで返します。

FileHandler::writeFile(string FILE_NAME, string BUFFER [, string MODE])

BUFFER の内容をファイルに書き込みます。

- FILE_NAME: 保存されるファイル
- BUFFER: 保存される内容
- MODE: 'w' は、新しく保存、'a' は既存ファイルの最後に内容追加

FileHandler::makeDir(string PATH)

PATH のフォルダとその下位のフォルダを再帰的な方式で生成します。

```
FileHandler::makeDir(_XE_PATH_ . 'files/cache/nhn/openuitech/sol');
```

FileHandler::removeDir(string PATH)

PATH のフォルダとその下位のフォルダを再帰的な方式で削除します。

```
FileHandler::removeDir(_XE_PATH_ . 'files/cache/openiuthech');
```

bool FileHandler::getRemoteFile(string URL, string TARGET_FILE)

リモートファイルを受け取り、ローカルに保存します。

- URL: http://で始まるパスを入力します。
- TARGET_FILE: 保存されるファイル

bool FileHandler::createImageFile(string SOURCE_FILE, string TARGET_FILE ,int WIDTH, int HEIGHT, string FILE_TYPE, string THUMBNAIL_TYPE)

既存の画像ファイルを利用してサイズと生成方式(縦横比率維持、切り取り)を指定してサムネイルを生成します。

- SOURCE_FILE: 原画像ファイル
- TARGET_FILE: 保存される画像ファイル
- WIDTH: 保存される画像の幅
- HEIGHT: 保存される画像の高さ
- FILE_TYPE: 保存される画像のタイプ

- THUMBNAIL_TYPE: ratio、crop、または thumbnail