
# 개발자 매뉴얼

# 저작권



이 문서는 정보 제공의 목적으로만 제공됩니다. NHN㈜는 이 문서에 수록된 정보의 완전성과 정확성을 검증하기 위해 노력하였으나, 발생할 수 있는 내용상의 오류나 누락에 대해서는 책임지지 않습니다. 따라서 이 문서의 사용이나 사용 결과에 따른 책임은 전적으로 사용자에게 있으며, NHN㈜는 이에 대해 명시적 혹은 묵시적으로 어떠한 보증도 하지 않습니다.
관련 URL 정보를 포함하여 이 문서에서 언급한 특정 소프트웨어 상품이나 제품은 해당 소유자가 속한 현지 및 국내외 관련법을 따르며, 해당 법률을 준수하지 않음으로 인해 발생하는 모든 결과에 대한 책임은 전적으로 사용자 자신에게 있습니다.

NHN㈜는 이 문서의 내용을 예고 없이 변경할 수 있습니다.

**오픈 소스 라이선스 관련 고지**
XE는 여러 종류의 오픈 소스 라이선스 중 LGPL(GNU Lesser General Public License) v2를 따르고 있습니다. LGPL v2와 v3에는 약간의 차이가 있으므로 버전까지 기억해야 합니다. LGPL은 기본적으로 GPL과 동일하나 적용 범위가 더 제한적입니다. LGPL도 GPL처럼 해당 라이선스를 가진 소프트웨어를 포함한 소프트웨어도 같은 라이선스를 갖도록 강제하는 효력이 있습니다. 그러나 GPL이 GPL 라이선스를 가진 소프트웨어를 포함한 모든 소프트웨어에 무조건적인 소스 공개를 강제하는데 비해, LGPL 라이선스를 가진 프로그램은 특정 조건 내에서 사용할 경우에는 소스 공개 의무가 없습니다. 따라서 LGPL 라이선스를 가진 소프트웨어는 독점 소프트웨어를 개발하는 데에도 사용할 수 있습니다. 자세한 내용은 아래 사이트를 참조하시기 바랍니다.
LGPL 라이선스: http://www.gnu.org/copyleft/lesser.html
GPL 라이선스: http://www.gnu.org/licenses/gpl.html

# 문서 정보

**문서 개요**

이 문서는 XE의 모듈과 애드온, 위젯 등 XE 추가 기능을 개발하는 방법을 설명합니다. 이 문서의 내용은 XE core 1.5.x 버전을 기준으로 합니다.

**독자**

이 문서는 XE의 추가 기능을 개발하고자 하는 개발자를 대상으로 합니다. 이 문서에서는 웹 서버 및 PHP 기술에 대해서 상세히 설명하지 않습니다. 웹 서버와 PHP 기술에 대해서는 관련 서적을 참조하시기 바랍니다.

**문의처**

이 문서의 내용에 오류가 있거나 내용과 관련한 의문 사항이 있으면 아래의 연락처로 문의합니다.

연락처: developers@xpressengine.com

**문서 버전 및 이력**

| 버전 | 일자 | 이력사항 |
|---|---|---|
| 1.0 | 2011.07.29 | 1.0 배포 |
| 1.1 | 2011.12.15 | XE core 1.5 기준 개정 |
| 1.1 | 2012.09.06 | Sub-Query 작성법 수정 |

# 표기 규칙

**참고 표기**

**참고**

독자가 참고해야 할 내용을 기술합니다.

**주의 표기**

**주의**

독자가 반드시 알아야 할 사항, 시스템 에러를 유발할 수 있는 사항, 수행하지 않았을 때 재산상의 피해를 줄 수 있는 사항을 기술합니다.

**윈도(창) 이름/사이트 이름/메뉴 이름/필드 이름/선택 값, 사용자 입력 값 및 기호 표기**

이 문서에서 윈도(창) 이름, 사이트 이름, 메뉴 이름, 입력 필드 이름, 선택 값, 사용자 입력 값은 다음과 같이 표기합니다.

- 윈도(창) 이름 : **윈도 이름** 창
- 사이트 이름 : '네이버 데스크톱 다운로드' 사이트
- 메뉴 이름 : **메뉴 > 하위메뉴**
- 선택 값 : **NBoard 1.0**을 선택합니다.
- 사용자 입력 값: *localhost*를 입력합니다.

**소스 코드 표기**

이 문서에서 소스 코드는 회색 바탕에 검정색 글씨로 표기합니다.

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

# 목차

# 표 및 그림 목록

## 표 목록



## 그림 목록

# 1. XE core 이해하기

이 장에서는 XE core의 기본 정보를 소개하고, XE의 요청 라이프사이클과 폴더 구조를 설명합니다.

## 1.1 개요

XE core는 개발자들이 원하는 웹 애플리케이션을 만드는 기반이 되는 프레임워크입니다. XE core는 회원 관리, 문서/댓글 관리뿐 아니라, DBMS의 종류와 상관없이 데이터를 관리할 수 있는 기능도 제공합니다. XE 자체가 MVC(Model-View-Controller) 구조로 되어 있어 완벽한 SoC(Separation of Concerns)를 구현할 수 있습니다.

XE 애플리케이션으로 유입되는 모든 요청은 index.php에서 처리합니다. 이 페이지는 요청 컨텍스트를 초기화하고 적절한 모듈을 찾아서 클라이언트(브라우저)로 응답을 보내는 역할을 담당합니다.

XE의 기능 대부분이 모듈로 이루어져 있습니다. XE는 요청이 들어 오면 모듈 이름과 액션 이름(없으면 기본값 사용)을 기준으로 어떤 모듈을 사용할지 정합니다. 예를 들어, 관리자 페이지를 표시하려면 URL은 <root_url>?module=admin&act=dispBoardAdminContent이 됩니다.

이 장에서는 XE 기반의 추가 기능인 모듈, 애드온, 위젯을 개발하기 위해 알아야 하는 XE의 기본 구조와 요청 라이프사이클을 설명합니다.

## 1.2 XE 요청 라이프사이클

XE 요청 라이프사이클이란 URL에 접속한 순간부터 클라이언트에 응답을 보낸 순간까지 XE가 거치는 일련의 과정을 나타냅니다. 다음은 XE 요청 라이프사이클을 나타낸 그림입니다.

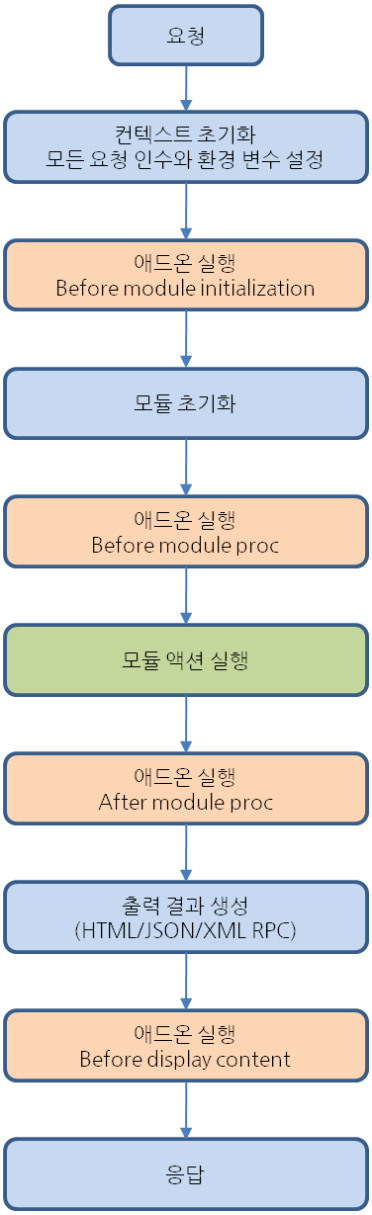


**그림 1-1 XE 요청 라이프사이클**

XE 요청 라이프사이클의 주요 과정은 다음과 같습니다.

1. 컨텍스트 초기화
2. 모듈 초기화
3. 요청된 모듈 액션 실행
4. 응답 결과 생성

개발자는 애드온을 사용해서 이 라이프사이클의 특정 순간에 커스텀 코드를 실행할 수 있습니다. 애드온은 XE 추가 기능의 일종으로서, PHP include 메커니즘으로 작동합니다. 코드가 직접 코어 메서드에 포함되기 때문에 라이프사이클을 조작하는 기능을 구현할 수 있습니다. 애드온에 관한 자세한 설명은 "2.2 애드온"을 참조하십시오.

### 1.2.1 컨텍스트 초기화

컨텍스트 초기화는 Context 클래스에서 처리합니다. 이 클래스는 XE 액션이 실행되는 환경을 캡슐화합니다. 주요 역할은 다음과 같습니다.

- $GLOBALS의 컨텍스트 변수 설정(display handler에서 사용)
- 언어 유형에 따라 언어 파일 포함
- 컨텍스트와 세션의 인증(authentication) 정보 설정
- 서버에서 rewrite 모드를 사용하는지 확인
- 자바스크립트 사용을 위한 위치 설정

Context 클래스의 위치는 ./classes/context/Context.class.php입니다.

### 1.2.2 모듈 초기화

모듈 초기화는 ModuleHandler 클래스의 init() 메서드에서 처리합니다. init() 메서드의 역할은 다음과 같습니다.

- 모듈 초기화 전에 애드온 실행(before_module_init hook)
- 요청 인수에서 변수 설정
- XSS를 방지하기 위한 변수 인증
- module_srl, mid, document_srl를 기준으로 요청 모듈 검색
- 현재 모듈 정보를 컨텍스트로 설정

ModuleHandler 클래스 위치는 ./classes/module/ModuleHandler.class.php입니다.

### 1.2.3 요청 모듈 액션 실행

모든 모듈은 ModuleHandler 클래스의 procModule() 메서드를 통해 실행됩니다. 이 메서드의 역할은 다음과 같습니다.

- 모듈 실행 전에 후크된 애드온 실행(before_module_proc hook)
- 현재 모듈 액션 실행

### 1.2.4 응답 결과 생성

DisplayHandler 클래스는 결과 생성을 담당합니다. 요청 종류에 따라 HTML이나 XML/JSON 콘텐츠를 출력할 수 있습니다. HTML의 경우, 이 클래스는 지정된 템플릿 파일을 검색하고 분석하여 완성된

HTML 형식을 만듭니다. XML/JSON의 경우 ModuleObject 속성은 다른 포매팅 없이 XML/JSON으로 직렬화(serialize)됩니다.

## 1.3 XE 폴더 구조

XE를 설치하면 루트에 다음과 같은 파일과 폴더가 생성됩니다.

**표 1-1 XE의 파일과 폴더**

| 폴더/파일 | 설명 |
|---|---|
| addons | 모든 XE 애드온을 포함 |
| classes | XE core 클래스를 포함 |
| common | 모든 XE 모듈에 공통으로 사용되는 정적 파일과 템플릿을 포함. 글로벌 언어 파일도 이 폴더에 포함됩니다. |
| config | 기본 설정과 공통 기능을 포함 |
| files | 이 폴더는 설치 중에 생성되며, 업로드된 파일과 내부 캐시 파일, DB/환경 설정 파일을 저장합니다. |
| layouts | 기본, 커스텀 XE 레이아웃을 모두 포함 |
| libs | XE core 에서 사용하는 모든 라이브러리를 포함(예: ftp, tar). |
| m.layouts | 모바일 레이아웃 |
| modules | 모든 모듈(XE core 모듈, 커스텀 모듈)을 포함 |
| themes | 테마(레이아웃과 각종 모듈의 스킨을 모두 포함)를 포함 |
| widgets | 모든 XE 위젯을 포함 |
| widgetstyles | 위젯을 꾸미는 데 필요한 모든 위젯스타일을 포함 |
| index.php | XE 의 모든 입출력을 위한 게이트웨이 기능을 포함 |
| .htaccess | Apache Web Server 의 rewrite mod 를 사용하기 위한 설정 파일 |
| LICENSE | XE 의 라이선스를 포함 |

### 1.3.1 addons 폴더

애드온은 단순히 활성/비활성으로 설정할 수 있으며, 추가 설정이 필요하면 모듈과 연동할 수 있습니다. addons 폴더 구조는 다음과 같습니다.

**표 1-2 addons 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| addons | 애드온의 최상위 폴더 |
| addon_name | 애드온 이름으로 된 폴더 |
| conf | 애드온의 설정 파일 폴더 |

| 폴더/파일 | 설명 |
|---|---|
| info.xml | 애드온의 설명과 제작자, 버전 및 생성일자 정보를 작성합니다. |
| addon_name.addon.php | 애드온 실행 코드. 애드온 실행 시점에 삽입됩니다. |
| queries | 애드온에 사용하는 쿼리 모음 |
| queryID.xml | 쿼리 파일. 쿼리 스키마는 모듈에서 사용하는 것과 동일합니다. |

자세한 내용은 "2.2 애드온"을 참조합니다.

### 1.3.2 classes 폴더

classes 폴더에는 XE의 모듈과 애드온, 위젯 등의 컴포넌트가 공통으로 사용하는 라이브러리 클래스가 포함되어 있습니다.

폴더별로 다음과 같은 클래스를 제공합니다.

**표 1-3 classes 폴더별 클래스**

| 폴더 | 설명 |
|---|---|
| cache | XE core 에서 사용할 수 있는 모든 캐시 클래스(CacheAPC, CacheMemcache, CacheHandler) 포함. 기본 클래스는 CacheHandler 입니다. |
| context | 요청 인수나 환경 변수 같은 컨텍스트를 관리하는 컨텍스트 클래스 포함. |
| db | CUBRID, MySQL, Firebird, MySQL Innodb, MySQLi, PostgreSQL, SQLite2, SQLite3 with PDO 등 XE core 에서 지원하는 모든 DB 포함. |
| display | 실행 결과 출력을 담당하는 클래스 포함(요청 타입에 따라 다름: HTML, JSON 또는 XMLRPC). |
| editor | 에디터 핸들러 클래스 포함 |
| extravar | 게시물, 회원 등에 사용되는 확장 변수를 처리하는 클래스 포함 |
| file | 파일과 폴더 처리에 사용하는 클래스 포함 |
| handler | (*)Handler 의 추상 클래스 포함 |
| httprequest | 외부 서버로 HTTP 요청을 보내고 응답을 수집하는 데 사용되는 클래스 포함 |
| mail | 메일링 관련 클래스 포함 |
| mobile | 모바일 최적화를 위한 클래스 포함 |
| module | 모듈 핸들러와 모듈 오브젝트 클래스 포함 |
| object | XE 모듈 간 오브젝트 인스턴스를 전달하는 기본 클래스 포함 |
| page | 페이지 탐색(navigation)을 처리하는 기본 클래스 포함 |
| template | 정규 표현식을 이용해서 템플릿 파일을 PHP 코드로 변환하고 추후 사용을 위해 XE 캐 |

| 폴더 | 설명 |
|---|---|
| | 시로 만드는 클래스를 포함 |
| widget | 위젯 실행을 위한 핸들러 클래스 포함 |
| xml | XML 을 파싱하고 생성하는 클래스 포함 |

### 1.3.3 common 폴더

common 폴더에는 XE에 꼭 필요한 자원이 포함되어 있습니다.

**표 1-4 common 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| common | XE 에 사용되는 공통 JS, CSS 파일 모음 |
| css | XE 에 사용되는 공통 CSS 파일 모음 |
| default.css | 기본 스타일과 XE 에 특화된 스타일 정의 |
| button.css | XE 에서 사용되는 기본 버튼 스타일 정의 |
| js | XE 에 사용되는 공통 JS 파일 모음 |
| common.js | XE 에서 사용되는 여러 가지 유형의 자바스크립트 함수 정의 |
| jquery.js | XE 에서 사용되는 자바스크립트 프레임워크인 jQuery (http://jquery.com) 파일 |
| js_app.js | XE 에서 사용되는 자바스크립트 애플리케이션 프레임워크인 JAF 파일 |
| x.js | 크로스브라우징을 위한 자바스크립트 라이브러리 파일. 추후 삭제될 예정이므로 이 파일은 가급적 사용하지 않습니다. |
| xml_js_filter.js | XE 에서 사용되는 XML JS 필터 파일 |
| lang | XE 에서 지원되는 언어 파일을 포함하는 폴더 |
| tpl | XE 의 공통 레이아웃/템플릿 파일 모음 |
| common_layout.html | XE 에서 사용되는 공통 레이아웃 |
| default_layout.html | 사용 중인 레이아웃 스킨이 없을 때 콘텐츠만 출력하는 빈 레이아웃 |
| mobile_layout.html | XE 모바일 환경에서 사용되는 레이아웃 |
| popup_layout.html | XE 에서 팝업 창을 열 때 사용되는 레이아웃 |
| redirect.html | 다른 페이지로 이동해야 할 때 사용되는 템플릿 파일 |
| refresh.html | 새로 고칠 때 사용되는 템플릿 파일 |

### 1.3.4 config 폴더

설정 폴더에는 기본 설정 내용과 자주 사용되는 함수 모음이 저장된 파일이 포함되어 있습니다.

**표 1-5 config 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| config.inc.php | 개발자를 위한 XE 버전과 디버그 설정 파일 |
| config.user.inc.php | 개발자를 위한 디버그 설정 저장 파일(개발자가 직접 만들어 사용) |
| func.inc.php | XE 에서 자주 사용되는 함수를 포함하는 파일 |

### 1.3.5 files 폴더

캐시 파일, 업로드된 파일, 기타 모듈에서 필요한 파일을 포함합니다.

**표 1-6 files 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| _debug_message.php | XE 로그 파일.<br>./config/config.inc.php 의 _DEBUG_OUTPUT_ 상수에 설정된 값에 따라 PHP 에러 메시지와 DB 에러 등을 표시합니다. 이 파일은 기본적으로 존재하지 않으며, 디버그 로그가 발생하는 순간 생성됩니다. |
| attach | 파일 첨부(업로드 파일)에 사용되는 폴더 |
| binaries | gif, jpg, jpeg, png, swf, mpeg 외의 확장자로 된 첨부 파일을 저장합니다(악의적인 공격을 피하기 위해 fpassthru() 함수를 사용해서 파일을 실행(execute)하지 않고 콘텐츠를 클라이언트에게 전달). |
| images | 브라우저에서 직접 접근할 수 있는 이미지와 동영상 파일을 저장합니다. 하위 폴더 이름은 ./$module_srl/$document_srl/$file_name 와 같은 형식으로 짓습니다. |
| cache | 캐시 폴더 |
| addon | 애드온과 관련된 캐시파일 모음 |
| mobileactivated_addons.cache.php | 활성화된 애드온을 실행하는 PHP 코드 포함(모바일 환경). |
| pcactivated_addons.cache.php | 활성화된 애드온을 실행하는 PHP 코드 포함(PC 환경). |
| document_category | 문서 카테고리용 XML, PHP 캐시 파일 |

| 폴더/파일 | 설명 |
|---|---|
| editor | 에디터 컴포넌트의 정보 캐시 파일 |
| js_filter_compiled | XE 의 XML JS 필터의 캐시 파일 |
| lang_defined | 사용자 정의 언어 코드의 캐시 파일 |
| layout | XE 의 레이아웃 캐시 파일. 레이아웃 편집에서 수정된 레이아웃 콘텐츠는 이곳에 저장됩니다. |
| menu | XE 의 메뉴 모듈에서 생성한 메뉴 정보를 위한 XML 과 PHP 캐시 파일 |
| module_info | 각 XE 모듈의 정보 캐시 파일 저장 |
| opage | XE 의 외부 페이지 모듈용 캐시 파일 |
| optimized | CSS 와 JS 파일을 통합해서 트래픽을 줄이고 페이지 로딩 속도를 향상시키기 위한 최적화 캐시 파일 |
| page | XE 의 페이지 모듈용 캐시 파일 |
| queries | XE 의 XML Query 컴파일용 캐시 파일 |
| template_compiled | XE 의 템플릿 캐시 파일 |
| thumbnails | XE 의 문서 섬네일 이미지 |
| widget | XE 의 위젯 정보용 캐시 파일 |
| widget_cache | 생성된 위젯의 정보를 저장하고 활용하는 캐시 파일. 캐싱 시간이 위젯 내에 지정되면 캐시 파일을 저장합니다. |
| triggers | XE 의 트리거 함수용 캐시 파일 |
| widgetstyles | 위젯스타일의 정보를 저장하고 활용하는 캐시 파일 |
| newest_news.language.cache.php | 관리자 페이지에 있는 최신 뉴스의 임시 저장 파일 |
| config | DB, FTP 등 사이트 최고 관리자의 설정 정보 모음 |
| db.config.php | DB 설정 파일 |
| ftp.config.php | XE 가 설치된 서버의 파일을 저장하는 FTP 정보 |
| lang_selected.info | 관리자가 작업하고자 하는 특정 사이트의 언어 목록 저장 |
| member_extra_info | 회원 정보의 확장 변수에 사용된 파일 모음 |
| image_mark | 회원 이름 앞에 붙는 표시의 이미지 파일 |
| image_name | 회원 이미지의 이름 파일 |
| profile_image | 회원이 등록한 프로파일 이미지 파일 |

| 폴더/파일 | 설명 |
|---|---|
| signature | 회원의 서명 |
| point | 각 회원의 포인트 |
| new_message_flags | 특정 회원에게 새 메시지가 도착했는지 여부를 나타내는 임시 파일 위치 |
| agreement.txt | 회원 관리 모듈에서 설정한 약관을 저장하는 파일 |
| ruleset | 동적 룰셋 파일을 저장합니다. |
| theme | 현재 테마의 정보를 저장합니다. |

### 1.3.6 layouts 폴더

레이아웃은 콘텐츠(모듈)를 감싸는 껍데기입니다. 레이아웃은 모듈 인스턴스별로 설정해서 적용하거나, 제작자가 설정한 메뉴 인스턴스와 연동해서 메뉴 인스턴스에 포함된 모든 모듈 인스턴스에 일괄적으로 적용할 수 있습니다. 레이아웃 관리 메뉴에서 위젯 및 레이아웃 편집 기능을 통해 레이아웃 템플릿 파일을 수정할 수 있습니다.

**표 1-7 layouts 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| Layout Name | 레이아웃 루트 폴더 |
| conf | 레이아웃 정보가 포함된 설정 파일 포함 |
| info.xml | 레이아웃 제작자와 설명, 확장 변수, 연동 메뉴의 개수와 이름 정의 |
| layout.html | 레이아웃의 템플릿 파일 |

### 1.3.7 modules 폴더

modules 폴더 구조는 다음과 같습니다.

**표 1-8 modules 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| module_name | 모듈 이름으로 된 폴더 |
| conf | 모듈 설명, 액션과 권한(permission) 설정 포함 |
| info.xml | 모듈 제작자 정보와 설명 |
| module.xml | 모듈 동작과 관련한 정보를 포함하는 액션 모듈 정의 |
| lang | 언어 팩 파일 |
| en.lang.php | 영어 언어 팩 |

| 폴더/파일 | 설명 |
|---|---|
| schemas | 모듈 설치에 사용되는 DB 테이블 스키마.<br>현재 모듈이 새 DB 를 사용할 때만 사용되는 옵션 폴더 |
| table.xml | 테이블 스키마(테이블 이름으로 파일을 생성) |
| queries | insert, select, update 에 사용되는 쿼리를 정의하는 XML 문법 파일 |
| ruleset | 모듈에서 사용할 룰셋 XML 파일 |
| tpl | 모듈의 관리자 뷰(administrator view)를 위해 사용되는 템플릿 파일 |
| css | 스타일 시트 |
| images | 템플릿 이미지 저장 |
| js | 템플릿 JS 파일 저장 |
| filter | 처리 파일에 전달될 폼에서 노드와 파라미터 선언 |
| template_files.html | 스킨이 사용되지 않는 화면의 XE 템플릿 문법으로 제작한 스킨 파일(모듈의 관리자 화면 등) |
| skins | 모듈의 프런트 엔드에 출력되는 스킨 파일 |
| Skin Name | 스킨 이름 |
| css | 스타일 시트 |
| images | 스킨 이미지 저장 |
| js | 스킨 JS 파일 저장 |
| skin.xml | 스킨 제작자 정보와 스킨의 확장 변수 선언 포함 |
| template_files.html | XE 템플릿 문법으로 제작한 스킨 파일 |
| module_name.class.php | 설치, 업데이트, 삭제 함수를 포함하는 모듈의 기본 클래스 |
| module_name.view.php | 모듈의 프런트 엔드를 출력하는 뷰 함수 |
| module_name.model.php | 모듈 모델 클래스와 함수 정의 |
| module_name.controller.php | 사용자 인터페이스를 위한 컨트롤러 |
| module_name.admin.view.php | 모듈의 백 엔드를 출력하는 데 사용되는 뷰 클래스와 함수 |
| module_name.admin.model.php | 관리자용 모델 클래스와 함수 선언 |
| module_name.admin.controller.php | 관리자 함수의 컨트롤러 액션 |
| module_name.api.php | 뷰 기능과 비슷하게 화면 출력을 위한 데이터를 준비합니다. 좀 더 정확하게는 출력 결과에서 내부 데이터를 삭제하고, 웹뿐만 아니라 아이폰 앱처럼 다른 종류의 앱을 생성하기 위 |

| 폴더/파일 | 설명 |
|---|---|
| | 한 인스턴스에 사용되는 JSON 이나 XML 을 반환합니다. |
| module_name.wap.php | 출력 결과가 다른 WAP 휴대폰을 위한 클래스 |
| module_name.smartphone.php | 아이폰 등의 스마트폰을 위한 특수 클래스 |

모듈에 관한 자세한 내용은 "2.1 모듈"을 참조하십시오.

### 1.3.8 themes 폴더

테마는 레이아웃과 모듈 스킨을 편리하게 관리하는 기능입니다. 사이트 디자인의 통일성을 높이기 위해 사용됩니다.

**표 1-9 themes 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| Theme Name | 테마 루트 폴더 |
| conf | 테마 정보가 있는 설정 파일 포함 |
| info.xml | 테마 제작자와 설명, 테마에 포함된 스킨을 정의 |
| layouts | 레이아웃 스킨의 루트 폴더 |
| Layout Name | 레이아웃의 폴더 |
| conf | 레이아웃 정보가 포함된 설정 파일 포함 |
| info.xml | 레이아웃 제작자와 설명, 확장 변수, 연동 메뉴의 개수와 이름 정의 |
| layout.html | 레이아웃의 템플릿 파일 |
| modules | 모듈 스킨 모음의 루트 폴더 |
| Module Name | 해당 스킨이 적용될 모듈의 이름 |
| css | 스타일 시트 |
| images | 스킨 이미지 저장 |
| js | 스킨 JS 파일 저장 |
| skin.xml | 스킨 제작자 정보와 스킨의 확장 변수 선언 포함 |
| template_files.html | XE 템플릿 문법으로 제작한 스킨 파일 |

### 1.3.9 widgets 폴더

위젯은 화면에 표시되는 작은 프로그램입니다. 위젯 중 일부는 최신 글이나 회원 정보(로그인 폼)와 연동하고, 일부는 외부 오픈 API와 연동하기도 합니다.

위젯 폴더의 이름은 해당 위젯의 이름과 같아야 합니다. 폴더 구조는 다음과 같습니다.

**표 1-10 widgets 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| Widget Name | 위젯 루트 폴더 |
| widget_name.class.php | 위젯의 클래스 파일. 데이터를 처리하고 템플릿 파일 지정 |
| conf | 설정 폴더 |
| info.xml | 위젯 정보(이름, 설명)와, 위젯 클래스에 사용할 수 있는 변수 정의 |
| skins | 스킨 폴더 |
| Skin Name | 위젯 스킨용 파일 포함. 이 폴더의 이름은 스킨 이름과 같아야 합니다. |
| skin.xml | 스킨의 이름, 설명, 제작자, 컬러셋 정보를 포함하는 설정 파일 |

### 1.3.10 widgetstyles 폴더

이 폴더는 위젯스타일을 포함합니다. 위젯스타일은 위젯 컨테이너를 꾸미는 데 사용되며, 사용자는 위젯스타일을 이용해 위젯의 배경, 테두리, 제목 등 위젯의 모양을 변경할 수 있습니다.

위젯스타일의 폴더 구조는 다음과 같습니다.

**표 1-11 widgetstyles 폴더 구조**

| 폴더/파일 | 설명 |
|---|---|
| widgetstyles | 위젯스타일 폴더 |
| Widgetstyle names | 위젯스타일 이름 |
| widgetstyle.html | 위젯스타일용 템플릿 파일 |
| skin.xml | 위젯스타일의 제목, 설명, 제작자, 확장 변수 등을 설정하는 파일 |
| preview.gif | 위젯스타일 미리 보기 |

# 2. XE 추가 기능

이 장에서는 XE 추가 기능인 모듈, 애드온, 위젯을 개발하는 방법을 설명합니다.

## 2.1 모듈

XE는 다양한 추가 기능을 사용해서 업그레이드할 수 있는 CMS(Contents Management System)입니다. 추가 기능 중 가장 중요한 것이 모듈입니다. 모듈은 플랫폼에 새로운 기능을 추가하는 파일 모음입니다.

모듈을 생성하려면 다음의 세 가지 규칙을 따라야 합니다.

- 모듈은 modules 폴더 아래의 폴더에 저장해야 합니다. 폴더 이름은 모듈 이름과 같게 지정합니다. 생성한 모듈을 배포하려면 다른 개발자가 만든 모듈과 이름이 충돌하지 않게 독창적으로 모듈 이름을 정합니다.
- info.xml 파일에 모듈 제작자와 모듈 설명, 버전, 제작일과 같은 일반적인 정보를 작성합니다.
- module.xml 파일에 설정 파라미터와 액션 정의 등을 저장합니다.

### 2.1.1 config/info.xml 작성

info.xml은 다음과 같은 형태로 되어 있습니다.

```
<?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> 요소는 관리자 메뉴에서 모듈 분류를 나타냅니다. 입력할 수 있는 옵션은 다음과 같습니다.

- service: 서비스 관리
- member: 회원 관리
- content: 정보 관리
- construction: 사이트 설정
- utility: 기능 설정
- accessory: 부가 기능 설정
- system: 시스템 관리/설정
- package: cafeXE, textyle 등의 패키지 모듈

### 2.1.2 액션 작성

XE에서 모든 입출력은 index.php를 통해 처리됩니다. 액션 요청 인수는 Module Handler에서 결정하며, 보통 $act 변수를 사용합니다. 모듈의 액션은 conf/module.xml 파일에 선언됩니다.

```
<?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">포럼</title>
      </menu>
   </menus>
</module>

<action>
```

conf/module.xml에서 사용되는 속성은 다음과 같습니다.

**표 2-1 액션 작성에 사용되는 속성**

| 속성 | 설명 |
|---|---|
| name | 모듈 이름도 포함하는 액션 이름. 관리자 권한이 필요한 액션의 이름에는 "Admin"을 포함합니다. |
| type | 액션이 어떤 타입이며 어떤 파일(뷰, 모델, 컨트롤러)에 저장돼야 하는지 정의. 이름에 "Admin"이 포함되어 있으면 관리자 뷰, 모델 또는 컨트롤러 PHP 파일에 있어야 합니다. |
| standalone | 이 속성이 "true"로 설정되면 현재 액션은 나머지 모듈에 독립적입니다. 이 속성이 "false"로 설정되면 요청이 수신되지 않고 모듈이 실행될 때 에러를 출력합니다.<br>현재는 사용되지 않는 속성으로 deprecated 예정입니다. |
| index | 모듈의 기본 액션 설정. 한 액션에만 적용되어야 합니다. |
| admin_index | 모듈 백 엔드의 기본 액션. 한 액션에만 적용됩니다. |

| 속성 | 설명 |
|---|---|
| setup_index | 모듈 설정 페이지로 사용되며, 관리자 액션에만 설정할 수 있습니다. |
| menu_name | 해당 액션이 속한 메뉴의 이름 |
| menu_index | 이 속성이 ""true""로 설정되면 이 액션이 현재 메뉴의 초기 액션이라는 의미입니다. |
| ruleset | 해당 액션에 적용할 룰셋 이름 |
| action | 권한(permission)이 선언된 액션의 이름 |
| target | 지원되는 권한은 다음과 같습니다.<br>• member: 회원<br>• manager: 관리자 |

### 2.1.3 Action Forward 사용

일반적으로 액션은 XE 모듈에 속합니다. 하지만 하나의 액션이 여러 모듈에서 사용되는 경우도 있습니다. 이를 Action Forward라고 부릅니다.

가장 전형적인 예는 RSS 모듈입니다. RSS 액션은 게시판 모듈에서 정의한 액션이 아니지만 Action Forward 기능에서 호출하고 실행합니다.

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

Action Forward를 사용해서 독립된 기능과 함께 모듈을 처리할 수 있습니다.

위의 요청에서 XE는 "board"라는 mid를 찾습니다. 이 mid가 rss 액션을 포함하지 않으면 XE는 DB에서 Action Forward 테이블을 통해 등록된 rss를 찾습니다. rss 액션은 rss 모듈의 뷰 타입으로 DB에 등록되어 있으므로 XE는 게시판을 위한 모든 mid 정보를 설정하고 rss 모듈의 뷰 객체를 생성해서 rss 메서드를 실행합니다.

이 Action Forward는 XE가 레이아웃이나 현재 요청된 모듈의 정보를 유지하면서 다른 메서드를 필요로 할 때 필요합니다. 다른 예제로는 친구 목록을 보기 위한 Communication 모듈의 dispCommunicationFriend 액션이 있습니다. 이 액션은 현재 모듈의 레이아웃을 유지하면서 해당 콘텐츠를 친구 목록으로 교체합니다.

즉, 콘텐츠 영역 출력은 지정된 액션에 의해 변경될 수 있으며, 요청된 모듈의 정보에 따라 다른 결과가 나올 수 있습니다.

#### Action Forward 등록

일반적으로 Action Forward는 module.class.php에서 moduleInstall()를 처리할 때 저장됩니다. 등록 방법은 다음과 같습니다.

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

#### Action Forward 검증

다음과 같이 Action Forward의 등록을 확인할 수 있습니다. 보통 module.class.php의 checkUpdate() 메서드에서 사용됩니다.

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

#### Action Forward 삭제

Action Forward가 필요 없어지면 다음과 같이 삭제합니다.

```
$oModuleModel = &getModel('module');
$oModuleModel = &getController('module');
if($oModuleModel->getActionForward('Action Name'))
   $oModuleController->deleteActionForward('Module Name','Type','Action Name');
```

액션 이름이 (disp|proc|get)+ModuleName+ActionName으로 되어 있다면 Action Forward를 등록하지 않아도 됩니다.

### 2.1.4 트리거 사용

어떤 모듈이 다른 모듈의 특정 액션에 어떤 동작을 하고 싶을 때 트리거를 사용합니다. 단, 해당 모듈에서 트리거를 제공해야 합니다. 예를 들면, document 모듈의 triggerDisplayDocumentAdditionSetup에 이미 있는 관리자용 뷰를 포럼 모듈에서 사용하고 싶은 경우가 있는데 이때 트리거를 사용합니다.

트리거를 사용하는 방법은 다음과 같습니다.

#### DB 에 트리거 삽입

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

#### 트리거 가져오기

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

#### 트리거 호출

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

#### 트리거 삭제

```
$ 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 룰셋 사용

룰셋은 HTML 폼의 정보를 PHP에 있는 처리 메서드로 전달할 때 클라이언트 측에서는 물론 서버 측에서도 정보의 유효성을 검증하기 위하여 사용합니다. 룰셋은 각 모듈 폴더의 ruleset 폴더에 있는 XML 파일에 저장됩니다. 다음은 룰셋의 예제입니다.

```
<?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>
```

룰셋에서 사용되는 요소와 속성은 다음과 같습니다.

**표 2-2 룰셋 작성에 사용되는 요소와 속성**

| 요소 | 속성 | 설명 |
|---|---|---|
| customrules | | 사용자 정의 규칙을 정의할 수 있습니다. |
| rule | | 사용자 정의 규칙 |
| | name | 사용자 정의 규칙의 이름 |
| | type | 사용자 정의 규칙의 유형. "regex", "enum", "expression" 중 하나를 사용할 수 있습니다.<br>• "regex": 정규표현식을 작성할 때<br>• "enum": 주어진 값 중 하나만 선택할 수 있을 때<br>• "expression": 수식이 필요할 때 |
| | test | 사용자 정의 규칙의 테스트 코드 |
| fields | | 유효성을 검사할 필드의 모임 |
| field | | 유효성을 검사할 필드 |
| | name | 폼 요소 이름 |
| | rule | 적용할 규칙 |
| | required="true" | 반드시 입력해야 합니다. |
| | length | 길이 제한. "최소:최대"와 같이 작성할 수 있습니다. |
| | default | 기본값. |
| | equalto | equalto 에 넣은 요소의 값이 현재 요소의 값과 같아야 함을 나타냅니다(비밀번호, 비밀번호 확인 등). |
| | modifier | 규칙을 사용하기 전에 입력값을 변경하거나 검사를 마친 후 결과를 변경할 수 있는 기능. |

더 자세한 내용은 "4 폼 사용"을 참조하십시오.

### 2.1.6 폼 필터 사용

필터는 HTML 폼에서 PHP에 있는 처리 메서드로 정보를 전달하고 자바스크립트 콜백 함수를 지정하기 위해 사용합니다. 필터는 tpl/filter 폴더 내에 있는 XML 파일에 저장됩니다. 다음은 폼 필터의 예제입니다. XE 1.5 버전부터는 폼 필터보다 룰셋을 사용하기를 권장합니다.

```
<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>
```

폼 필터에서 사용되는 요소와 속성은 다음과 같습니다.

**표 2-3 폼 필터에서 사용되는 속성**

| 요소 | 속성 | 설명 |
|---|---|---|
| form | | 입력값이 유효한지 확인하는 최상위 요소 |
| node | | HTML 폼 확인 |
| | required | 입력 요소가 반드시 필요한 값인지 설정. required 속성값이 true 로 설정된 경우, 해당 요소에 값이 입력되지 않으면 alert 가 발생합니다. |
| | filter = "filter type" | 필터에 사용할 수 있는 타입은 email(email_address), userid(user_id), url(homepage), korean, korean_number, alpha, number, alpha_number 입니다. |
| | equalto = "target person" | equalto 에 넣은 요소의 값이 현재 요소의 값과 같아야 함을 나타냅니다(비밀번호, 비밀번호 확인 등). |
| | maxlength | 최대 길이 |
| | minlength | 최소 길이 |
| parameter | | 서버로 전송할 때 폼 요소의 이름을 변경하거나, 폼 요소 중 개발자가 parameter 에 작성한 값만을 서버에 전송하고자 할 때 사용. |

| 요소 | 속성 | 설명 |
|---|---|---|
| | | 기본적으로 parameter 를 사용하지 않으면 모든 폼 요소를 서버에 전송합니다. |
| param | | 재정의하거나 서버에 전송할 폼 요소 정보 작성 |
| | name | 폼 요소 이름 |
| | target | 재정의할 요소 이름 |
| response | | |
| | callback_func | 자바스크립트 콜백 함수. 실제 구현되어야 합니다. |
| tag | | 콜백 함수로 전달될 변수 정의 |
| | name | 콜백 함수에 전달할 변수 이름. 이 변수들은 controller 에서 액션을 실행한 후 콜백 함수에 전달할 값들을 $this->add('변수명', '값')으로 구현합니다. |

더 자세한 내용은 "4 폼 사용"을 참조하십시오.

### 2.1.7 DB 쿼리 정의

XE는 커스텀 쿼리 언어를 사용해서 쿼리를 정의합니다. XML 코드는 ./classes/xml 폴더에 있는 XmlQueryParser.class.php에서 파싱합니다. 사용 예는 다음과 같습니다.

```
<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 애드온

XE에서 애드온은 후킹(hooking)을 수행합니다. 후킹이란 다른 정상적인 액션을 가져오는 행위를 말합니다. 후킹은 PHP 같은 인터프리터 기반 언어에서 사용하는 'include'를 사용합니다. XE에서는 애드온을 XE의 컨텍스트에 네이티브 코드로 삽입할 수 있도록 함수나 클래스 형태로 작성하지 않습니다. 이 때문에 XE의 애드온은 호출된 순간부터 강력한 효과를 발휘할 수 있습니다. 하지만 애드온은 XE의 전체 운영에 부하를 줄 수 있으므로 조심해서 생성해야 합니다.

애드온을 생성하려면 다음의 규칙을 따라야 합니다.

- 애드온은 addons 폴더 아래의 addon_name 폴더에 저장해야 합니다.
- 애드온 실행 파일의 이름은 addon_name.addon.php여야 합니다.
- info.xml 파일에 제작자 정보, 애드온 설명, 관리자(필요한 경우)로부터 받은 애드온 변수를 저장해야 합니다.

### 2.2.1 애드온 호출 시점

애드온을 호출하는 시점은 다음과 같습니다.

- before_module_init: 모듈 객체 생성 전: 사용자가 요청한 모듈을 찾은 후, 해당 모듈의 객체를 생성하기 전
- before_module_proc: 모듈 실행 전: 모듈의 객체를 초기화한 후, 해당 모듈을 실행하기 전
- after_module_proc: 모듈 실행 후: 생성된 모듈 객체를 실행하고 결과를 얻은 직후
- before_display_content: 결과 출력 전: 레이아웃이 적용된 모듈의 결과를 출력하기 직전

각 후크가 어떤 역할을 하고, 왜 XE 제어 경로(control path)에서 일부 애드온이 특정 시점만 사용하는지 이해하기 위해 몇 가지 예를 들어보겠습니다.

**태그 목록 - After module proc**

모든 문서의 태그 목록을 한 페이지에 출력하는 애드온이 있다고 가정해 봅시다. 이런 태그 목록을 생성하려면 먼저 현재 페이지의 module_srl이 포함된 문서를 가져와야 하고, 그 전에 module_srl을 알아야 합니다. 이를 위해 after_module_proc라는 지점을 선택해야 합니다. 모듈 정보가 처리된 후에 이전에 정의된 모든 작업을 실행할 수 있습니다.

**메타 태그 - Before module proc**

이 애드온은 메타 설명과 키워드, 제작자 등의 메타 태그를 모든 페이지에 삽입하는 역할을 합니다. 콘텐츠가 모듈 처리 작업에서 생성되기 전에 메타 태그를 삽입해야 하기 때문에 before_module_proc 지점을 후크로 사용합니다.

**포인트 레벨 아이콘 - before display content**

특정 회원이 쌓은 포인트 레벨에 따라 각 사용자에게 아이콘을 표시하는 애드온이 필요합니다. 이 애드온은 이미 처리된 일부 파라미터에 따라 콘텐츠 내의 HTML 코드를 수정해야 하기 때문에 before_display_content 지점을 후크로 사용합니다.

**카운터- Before Module Init**

이 애드온은 XE를 사용해서 구축한 웹사이트의 방문 통계를 보기 위해 개발됐습니다. 이 애드온은 카운터 모듈을 사용합니다. 카운터 애드온은 $is_logged 변수의 정보를 사용해서 웹사이트 방문 횟수를 계산합니다. 모듈 처리 작업으로부터 더 이상 정보를 얻을 필요가 없으므로, 이 모듈은 시간상으로 첫 번째 후크인 before_module_init을 사용합니다.

### 2.2.2 애드온 호출 시 전달되는 변수

앞에서 설명한 네 번의 호출 시점에 XE core는 다음과 같은 공통 변수를 애드온에 전달합니다.

- $called_position: 호출 시간 정보를 포함합니다. 값은 before_module_init, before_module_proc, after_module_proc, before_display_content 네 가지 중 하나입니다.
- $addon_path: 호출된 애드온의 경로를 포함합니다.
- $addon_info: XE의 애드온은 독립적으로 설정할 수 있고, 해당 애드온이 동작할 대상 모듈을 지정할 수 있습니다. $addon_info 변수는 애드온에서 선언된 extra_vars(info.xml 내)의 정보를 포함하며, 이는 애드온마다 다릅니다.

### 2.2.3 애드온 파일 작성

addons 폴더에 다양한 이름으로 된 파일을 저장할 수 있고, 폴더 내에서 클래스를 사용할 수도 있습니다. 하지만 함수 선언은 네이티브 코드로 동작하는 include 구조를 사용하기 때문에 허용되지 않습니다.

**config/Info.xml**

info.xml 파일은 다음과 같이 작성합니다.

```
<?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>
```

필요하면 extra_vars를 생성합니다. 세부 내용이 없으면 "<extra_vars />" 명령어를 사용해서 생략합니다. 위와 같이 작성한 파일을 info.xml이라는 이름으로 conf 폴더 아래에 저장합니다.

**addon_name.addon.php**

애드온이 어떤 액션을 수행하기로 되어 있다면 PHP 형식으로 애드온 파일을 작성합니다. 단, 애드온은 대개 클래스 객체의 메서드 내에서 호출되므로 함수는 선언할 수 없습니다. 애드온 내에서 클래스를 정의하고 사용할 수 있습니다.

애드온 파일의 시작 부분은 다음과 같아야 합니다.

```
<?php

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

XE의 모든 기능은 index.php를 통해 실행되며, index.php는 __ZBXE__상수가 true로 설정되면 시작됩니다. 따라서, index.php에 선언되어 있는 __ZBXE__상수가 true로 설정되어 있는지 기능을 실행하기 전에 확인합니다. 애드온 실행 시점은 called_position으로 확인할 수 있으며, 이 작업은 반드시 해당 애드온에서 수동으로 해야 합니다.

예를 들어 페이지 아래에 모든 문서의 태그 목록을 출력하는 애드온이 있다고 가정해 보겠습니다. 우선 애드온을 호출하려면 어떤 후크가 적당한지 확인해야 합니다. 문서의 목록을 얻으려면 다음 코드와 같이 페이지 모듈을 처리해서 'after_module_proc'를 사용해야 합니다.

```
<?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);
?>
```

위의 코드는 현재 보이는 HTML 페이지에 태그 목록 HTML 코드를 삽입합니다.

### 2.2.4 XE XML 쿼리 사용법

XE 애드온에서 다른 모듈이 생성한 DB에 있는 데이터는 XML 쿼리를 통해 사용할 수 있습니다. 이 경우 addon 폴더 아래에 queries라는 폴더를 하나 생성하고 XML 쿼리문을 정의한 XML 파일을 저장합니다. 쿼리를 실행하는 방법은 다음과 같습니다.

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

### 2.2.5 애드온 생성 시 고려 사항

애드온을 생성할 때 고려 사항은 다음과 같습니다.

- XE의 애드온은 모든 모듈의 여러 부분에 삽입되므로 <?php ... ?> 앞뒤로 공백이 없어야 합니다. 공백이 포함되면 before_display_content가 호출되어도 제대로 동작하지 않습니다.
- XE core는 애드온을 프로그래밍할 때 발생할 수 있는 예외를 별도로 처리하지 않습니다. 따라서 현재 호출 상황을 확인하는 루틴을 잘 구현해서 예외가 발생하지 않도록 해야 합니다.
- 애드온 코드 에러로 인해 웹 사이트에 심각한 에러가 발생한다면 files/cache/activated_addons.cache.php 파일을 수정해서 다시 업로드합니다.

XE 애드온은 강력한 액션을 수행할 수 있습니다. 하지만 코드를 적절하게 작성하지 않으면 예기치 않은 결과가 나오거나 XE가 중단될 수도 있습니다. 따라서 애드온을 생성할 때는 기본 애드온을 참조하는 것을 권장합니다.

## 2.3 위젯

위젯은 화면에서 데이터를 출력하는 데 사용되는 컴포넌트입니다. 위젯은 최신 글과 회원 정보 같은 기존 모듈이나 외부 API로부터 추출한 데이터와 함께 연동할 수 있습니다. 위젯은 모든 종류의 페이지에 추가할 수 있으며 레이아웃에 직접 추가할 수도 있습니다. 위젯을 통해 출력되는 콘텐츠를 쉽게 커스터마이즈할 수 있습니다.

위젯은 관리자가 수동으로 페이지 모듈에 입력하고 <img/> 요소에 저장합니다. 출력할 웹 페이지를 호출할 때 widgetController::triggerWidgetCompile() 트리거가 widgetproc()를 사용해서 <img/> 요소의 코드를 실행하고 올바른 HTML 코드로 변환합니다.

### 2.3.1 config/info.xml 작성

info.xml 파일은 위젯 제작자와 버전, 기타 설정 변수에 관한 정보를 저장합니다. 다음과 같이 작성합니다.

```
<?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 위젯 클래스 개발

위젯이 무슨 기능을 하는지는 widgetName.class.php라는 클래스 파일에 구현합니다. 위젯을 구현하는 모든 클래스는 WidgetHandler를 상속해서 proc() 메서드를 구현해야 합니다.

```
<?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 확장 변수 사용

확장 변수는 위젯을 페이지에 삽입하기 직전에 위젯의 관리 부분에서 데이터를 가져오기 위해 사용합니다. 페이지에서 자동으로 생성될 각 변수의 값을 얻기 위해 변수별로 입력 타입을 설정할 수 있다. 위젯의 확장 변수는 다음과 같습니다.

- text: 일반 문자열 타입
- textarea: 문단을 포함한 문자열 타입
- select: 여러 내용 중 하나를 선택
- select-multi-order: 다음 그림과 같이 선택 요소를 결정하고 순서를 바꾸는 경우 사용


- mid: 모듈을 하나만 선택
- mid_list: 모듈을 여러 개 선택
- menu: 사이트 메뉴 중 하나를 선택

# 3. DB 연동

이 장에서는 XE와 DB의 연동 방법을 설명합니다.

## 3.1 개요

XE에는 데이터베이스-애그노스틱(database-agnostic) DB 추상 레이어가 있습니다. 이 말은 XE를 다양한 DBMS와 함께 사용할 수 있고 DBMS 간 전환도 쉽게 할 수 있다는 뜻입니다. XE는 MySQL, MS SQL, CUBRID, PostgreSQL, SQLite3, Firebird를 지원합니다.

이를 위해서 XE의 XML 스키마 언어(XML Schema Language)와 XML 쿼리 언어(XML Query Language)를 사용해서 전체 DB 스키마와 쿼리를 XML로 작성합니다.

다음은 XML 스키마 파일의 예제입니다.

```
# 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>
```

XE를 처음 설치할 때 포함된 모듈 중 테이블.xml이 있으면 자동으로 테이블이 생성됩니다. XE를 설치한 후 추가 모듈을 설치할 때 테이블.xml이 있다면, 관리자 화면에 **모듈 설치** 버튼이 나타납니다. XML 파일을 통해 이 테이블에 대한 쿼리를 생성할 수 있습니다.

```
#./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>
```

이 쿼리를 PHP에서 호출하는 코드는 다음과 같이 매우 간단합니다.

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

## 3.2 XML 스키마 언어 레퍼런스

XE의 DB 테이블 스키마는 XML 파일로 정의됩니다. DB 테이블 스키마는 각 모듈의 schemas 폴더에 저장됩니다.

XML 스키마 파일은 하나의 루트 <table> 요소와 하나 이상의 자식 <column> 요소로 구성됩니다. 각 요소의 속성은 다음과 같습니다.

**표 3-1 <table> 요소의 속성**

| 속성 | 설명 |
|---|---|
| name | 생성될 테이블 이름. 접두어 xe_가 자동으로 추가되며 별도로 지정할 필요가 없습니다. XML 파일 이름과 동일해야 합니다. |

**표 3-2 <column> 요소의 속성**

| 속성 | 설명 |
|---|---|
| Name | 열(column) 이름 |
| Type | 열이 저장할 데이터 타입. 값은 다음 중 하나입니다.<br>• number<br>• bignumber<br>• varchar<br>• char<br>• text<br>• bigtext<br>• date<br>• float<br>파서가 이 데이터 타입을 각 DB 의 데이터 타입에 자동으로 매핑합니다. 예를 들어, bignumber 는 MySQL 의 bigint 에 해당합니다.<br>각 데이터 타입을 DB 별 데이터 타입에 매핑하는 방법에 대한 자세한 내용은 "표 3-4 XE-DBMS 간 데이터 타입 매핑"을 참조하십시오. |
| size | 열의 크기. 숫자나 문자 타입에 사용됩니다.<br>• 숫자 타입: 정확도를 나타냅니다.<br>• 문자 타입: 해당 문자열이 포함할 수 있는 문자의 개수를 나타냅니다. |
| default | 열의 기본값 지정 |

| 속성 | 설명 |
| --- | --- |
| notnull | 열이 널(null)값을 허용하는지 지정. 열이 널값을 허용하면 이 속성을 생략하고, 허용하지 않으면 다음과 같이 이 속성을 추가합니다.<br>예제) notnull = "notnull" |
| primary_key | 테이블의 primary key 로 사용될 열(column) 지정. 각 열에 primary_key="primary_key" 속성을 지정하면 두 속성이 묶여서 primary_key 가 됩니다. |
| index | 열의 인덱스 생성. 이 속성의 값은 생성될 인덱스의 이름을 나타냅니다.<br>인덱스 이름을 하나 이상의 열에 중복으로 사용하면 결합 인덱스가 생성됩니다.<br>예제) index="idx_list_order" |
| unique | 열에 대한 고유한 인덱스 생성. 이 속성의 값은 생성될 인덱스의 이름을 나타냅니다. |
| auto_increment | 열값이 자동으로 증가하는지 지정합니다.<br>예제) auto_increment="auto_increment" |

## 3.3 XML 쿼리 언어

XE는 다양한 DB를 지원하기 위해 SQL 쿼리를 그대로 사용하지 않고 XML로 작성합니다.

### 3.3.1 사용 방법

XML 쿼리는 모듈과 애드온, 위젯 등에서 다음과 같이 사용할 수 있습니다.

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

executeQuery() 함수는 ./classes/db/DB.classs.php에 있는 DB::executeQuery() 함수의 별칭(alias)입니다. 이 함수는 실제 DB 데이터를 조작하고 사용된 DB에 따라 XML 쿼리가 네이티브 SQL로 파싱된 후에 결과를 수신합니다.

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

- 첫 번째 파라미터는 실행될 XML 쿼리의 이름입니다. 값은 "모듈이름.쿼리ID"입니다.
- 두 번째 파라미터는 stdClass의 타입으로서, 데이터를 쿼리에 전달하는 데 사용됩니다. 이 파라미터는 널(null)이 될 수 있습니다.
- 결과값은 Object 클래스의 객체로 반환됩니다.
  - $output->toBool()이 FALSE이면 쿼리 실패를 의미하며, $output->toBool()이 TRUE이면 쿼리가 정상적으로 실행되었다는 뜻입니다.
  - select문의 결과는 $output->data 변수에 넣어 반환됩니다.

### 3.3.2 XML 요소

```
<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 쿼리에 사용되는 XML 요소와 속성은 다음과 같습니다.

**표 3-3 XML 쿼리에 사용되는 XML 요소와 속성**

| 요소 | 속성 | 설명 |
|---|---|---|
| <query> | | 쿼리 XML 의 최상위 요소 |
| | id | 쿼리 검색을 위한 아이디. module.query_id 를 사용해서 쿼리 XML 파일을 검색하고 사용합니다. |
| | action | 액션은 select, update, delete, insert 이렇게 네 가지 타입입니다. |
| | alias | 서브쿼리를 사용할 때 쿼리문의 alias 명입니다. |
| <tables> | | 쿼리에 사용될 테이블의 모음 |
| <table> | | 테이블 요소 |
| | name | 원래 테이블 이름(XE 의 접두어는 무시) |
| | alias | 열 지정이나 검색 등에서 사용할 테이블 별칭 |
| <columns> | | 쿼리에 사용될 열 모음 |
| <column> | | 열 요소 |
| | name | 열 이름 |
| | alias | 원래 열의 이름을 변경할 때 사용 |
| <conditions> | | 조건문을 만들 때 사용. <group> 요소를 사용해서 조건문을 여러 그룹으로 구분할 수 있습니다. |
| <group> ...<br></group> | | 조건문이 그룹으로 사용될 때 pipe="and\|or"를 사용해서 그룹 간 조건을 지정할 수 있습니다. |
| <condition> | | 조건문 |
| | operation | 다음과 같은 연산자로 처리할 수 있습니다.<br>• equal : column = (var\|default)<br>• more : column >= (var\|default)<br>• excess : column > (var\|default)<br>• less : column <= (var\|default)<br>• below : column < (var\|default)<br>• notequal : column != (var\|default)<br>• notnull : column is not null<br>• null : column is null<br>• like_prefix : column like '%var\|default'<br>• like_tail : column like 'var\|default%'<br>• like : column like '%var\|default%'<br>• in : column in (var\|default) |

| 요소 | 속성 | 설명 |
|---|---|---|
| | | • notin : column not in (var\|default) |
| | column | 열 이름을 지정합니다. |
| | var | executeQuery(Array)() 함수의 두 번째 파라미터인 stdClass 의 키값을 지정합니다. |
| | filter | var 값의 조건 필터링. 지원되는 필터는 다음과 같습니다.<br>• email, email_address: 메일 형식<br>• homepage: http\|https:// 같은 웹사이트 주소 형식<br>• userid, user_id: XE 의 사용자 id 형식(첫 두 글자는 알파벳이어야 합니다. 세 번째 문자부터 number+alphabet+ _ 형식으로 합니다.)<br>• number: 숫자 허용<br>• alpha: 알파벳 허용<br>• alpha_number: 숫자와 문자 모두 허용 |
| | default | var 값이 널이면 기본값으로 대체. 기본값은 일반 문자열, 숫자 모두사용 가능하며, 아래와 같은 함수를 사용할 수도 있습니다.<br>• ipaddress(): IP 주소<br>• unixtime(): 유닉스 시간(php 내 time() 함수)<br>• curdate(): YYYYMMDDHHIISS<br>• plus(int count): column = column + count<br>• minus(int count): column = column − count<br>• multiply(int arg): column = column * arg<br>• sequence(): XE 의 getNextSequence()를 실행 |
| | notnull | 널인지 확인. 지정하면 반드시 var 의 값이 있어야 합니다. |
| | minlength | 최소 길이 확인 |
| | maxlength | 최대 길이 확인 |
| | pipe | and\|or 같은 조건 지정 |
| <navigation> | | 정렬 순서나 페이징을 지원 |
| <index> | | 정렬될 열과 정렬 방식 지정 |
| | var | 열 이름이 값인 변수 이름 |
| | default | var 값이 지정되지 않은 경우 사용되는 기본으로 정렬된 열 이름 |
| | order | 정렬 방식. asc\|desc 가 아닌 변수의 이름을 사용하면, 변수의 값에 따라 정렬됩니다. 단, 변수의 값은 asc\|desc 로 전달되어야 합니다(오름차순 "asc", 내림차순 "desc") |

| 요소 | 속성 | 설명 |
|---|---|---|
| <list_count> | | 페이징 결과를 수신할 수 있게 합니다. |
| | var | 행의 개수가 값인 변수 이름 |
| | default | var 값이 지정되지 않으면 사용되는 행의 개수 기본값 |
| <page_count> | | 페이징을 계산할 때 내비게이션 개수 지정 |
| | var | 페이징 내비게이션의 개수가 값인 변수 이름 |
| | default | var 값이 지정되지 않으면 사용되는 기본 페이징 내비게이션의 개수 |
| <page> | | 현재 페이지 번호 지정 |
| | var | 현재 페이지 번호를 값으로 가지는 변수 이름 |
| | default | var 값이 지정되지 않으면 사용되는 기본 페이지 번호 |
| <groups> | | 조건문에 따라 그룹을 사용할 수 있게 합니다.<br>group by 절 사용 시 작성 |
| | column | group by 기준 열 이름 |

### 3.3.3 XML 서브쿼리 사용 예제

XE 1.5 버전부터 서브쿼리를 작성할 수 있습니다. 아래는 서브쿼리 타입별 작성 예입니다.

#### Select 절 사용

SQL 사용 예

```
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 서브쿼리 작성 예

```
<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 절 사용

SQL 사용 예

```
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 서브쿼리 작성 예

```
<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 절 사용

SQL 사용 예

```
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 서브쿼리 작성 예

```
<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 데이터 타입 매핑

XE와 각 DBMS의 데이터 타입은 다음과 같이 매핑됩니다.

**표 3-4 XE-DBMS 간 데이터 타입 매핑**

| 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 클래스는 XML 쿼리 파일을 입력 받아서 파싱한 후 SQL 쿼리(select, update, insert, delete 같은 쿼리 타입, 사용된 표현식, 연결/필터링 조건, 조건문에 따른 그룹/순서)를 생성하는 데 필요한 모든 정보를 관련 클래스 오브젝트 형태로 포함하는 PHP 파일을 생성합니다. 이 PHP 파일은 각 DB 클래스의 입력값으로 사용되고, 각 DB 클래스는 DBMS별로 적절한 이스케이프 문자와 커스텀 언어 구조를 사용해서 SQL을 생성합니다.

예를 들어, 다음과 같은 XML 쿼리가 있다고 가정해 보겠습니다.

```
# ./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 함수를 사용해서 이 쿼리를 호출하면 XE는 파싱한 결과를 포함하는 PHP 형태의 캐시 파일이 생성되었는지 확인합니다. XML Query Parser 클래스를 호출하지 않았으면 PHP 파일을 생성해서 ./files/cache/queries에 저장합니다.

```
# ./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; ?>
```

그런 다음 DB별 executeQuery 메서드가 호출되고 위의 파일의 출력 값이 이 메서드의 입력으로 사용됩니다. DB 클래스는 SQL 쿼리를 생성하고 실행합니다. 예를 들어 위의 쿼리는 다음과 같은 SQL 쿼리가 됩니다.

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

cache.php 파일 역시 열(column) 타입에 관한 정보를 포함하고 있습니다. 이 정보는 테이블 스키마 파일에서 추출합니다. XE는 먼저 ./modules/<module_name>/schemas/<table_name> 안에서 해당 스키마 파일을 찾습니다. 찾는 파일이 없으면 <table_name>라는 파일을 찾을 때까지 각 모듈을 검색합니다.

## 3.6 XE DB 클래스

XE는 지원하는 모든 DBMS에 대해 커스텀 클래스를 제공합니다. 커스텀 클래스는 DBMS별로 커스텀 SQL 문법을 생성합니다.

예를 들어 XE core와 함께 사용되는 클래스는 다음과 같습니다.

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

모든 커스텀 클래스는 ./classes/db에 저장됩니다.

모든 커스텀 DB 클래스는 공통 DB 클래스를 상속합니다. 코드를 작성할 때 일반 DB 클래스를 사용하면, XE가 어떤 DB 클래스 구현체를 사용할지 실행시간에 결정합니다.

# 4. 폼 사용

이 장에서는 폼을 사용하는 방법을 설명합니다.

## 4.1 개요

폼은 사용자 입력값을 서버에 전송할 때 사용됩니다. XE에서는 폼을 전송할 때 입력값의 유효성을 확인하기 위한 룰셋 기능을 제공합니다. XE의 룰셋 기능을 사용하면 입력값의 유효성 확인 스크립트를 별도로 제작할 필요가 없습니다.

폼별로 다음과 같은 항목이 필요합니다.

- 폼 마크업과 설계
- 폼 전송 시 호출되는 서버 측 메서드

이와 같은 폼 전송을 위해서 연동하는 XE의 구성 요소는 다음과 같습니다.

- 폼 템플릿 파일: 폼의 레이아웃과 필드 정의
- 폼 전송을 처리할 controller 메서드(controller 파일 내)
- 폼 유효성 검사를 위한 룰셋 XML 파일

## 4.2 XE 폼 작성

사용자에게 이름을 물어 입력하게 한 다음 환영 인사를 보여 주는 페이지를 나타내는, 아주 간단한 폼을 만들어 보겠습니다. 모듈에는 하나의 뷰만 있습니다. 이 뷰는 사용자가 이름을 입력한 후 hello 메시지를 표시하거나 이름을 입력하는 폼을 다시 출력합니다.

이 예제 모듈의 완성된 버전은 hello.zip에서 다운로드할 수 있습니다. 이 절에서 설명할 튜토리얼을 따라 하면서 직접 모듈을 완성하려면 시작 파일(hello-tutorial.zip)만 다운로드하십시오.

### 4.2.1 폼 뷰 생성

먼저 입력 상자와 등록 버튼만 있는 폼 형태를 만들어 보겠습니다. 파일 이름(name.html)을 정한 후 ./modules/hello/tpl/에 저장합니다.

```
<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>
```

XE에서 폼을 전송하려면 기본적으로 어떤 모듈의 어떤 액션으로 데이터를 전송할 지 추가하고, 데이터 유효성 검사를 위해 룰셋을 지정하여야 합니다. 위의 예제에서는 hello 모듈의 procHelloGreet 액션에 데이터를 전송하도록 하고, say_hello라는 룰셋 파일로 유효성 검사를 하도록 설정했습니다.

**참고**

form 태그의 ruleset 속성은 적용할 룰셋 파일의 파일 이름입니다. 해당 속성값 앞에 @를 붙이면 XE에서 동적으로 생성되는 룰셋을 참고하라는 뜻이 됩니다. 예를 들어, XE 1.5에서는 로그인 계정으로 user_id 또는 email_address를 선택하도록 되어 있습니다. 이때 설정값에 따라 로그인 데이터 유효성 검사 방식이 달라져야 하므로 동적 룰셋을 적용합니다. 동적 룰셋 파일은 files/ruleset에 저장됩니다.

템플릿 파일을 출력하는 뷰 메서드를 생성합니다. ./modules/hello/hello.view.php에 다음의 메서드를 추가합니다.

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

./modules/hello/conf/module.xml에 다음과 같이 추가해서 최종 사용자가 사용할 수 있게 합니다.

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

이제 /?module=hello로 접근하면 다음과 같은 폼을 확인할 수 있습니다.


**그림 4-1 이름 입력 폼**

### 4.2.2 XML 룰셋 파일과 컨트롤러 액션 추가

아직 우리가 만든 폼은 아무 기능도 하지 않습니다. 이 폼에 사용자 이름을 조회하고 hello 메시지를 출력하는 메서드를 추가해 보겠습니다.

./modules/hello/hello.controller.php에 다음과 같은 메서드를 추가합니다. 룰셋 파일을 사용할 때는 액션 실행 후 이동할 액션을 명시해야 합니다. procHelloGreet 함수의 실행이 완료되면 dispHelloName 화면으로 이동하도록 아래와 같이 setRedirectUrl을 설정합니다.

```
 /**
  * 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.xml의 <actions> 요소에 다음과 같이 추가합니다.

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

폼 내용의 유효성을 검사하려면 XML 룰셋 파일을 추가해야 합니다. 파일 이름을 say_hello.xml로 짓고 ./modules/hello/ruleset/ 아래에 저장합니다.

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

룰셋 파일에 사용되는 요소와 속성에 대한 자세한 설명은 "2.1.5 룰셋 사용"을 참조하십시오.

### 4.2.3 인사 메시지 출력

./modules/hello/hello.view.php에 있는 dispHelloName 메서드를 다음과 같이 수정합니다.

```
    /**
     * @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');
     }
```

템플릿 파일(./modules/hello/tpl/name.html)도 다음과 같이 수정합니다.

```
<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>브라우저에서 해당 페이지를 다시 로드하면 다음과 같은 인사 메시지를 볼 수 있습니다.

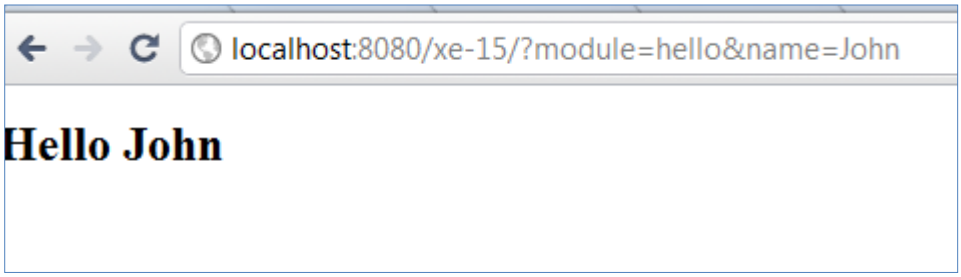


**그림 4-2 인사 메시지 출력**

이제 폼이 완성되었습니다.

# 5. Document 모듈 사용

이 장에서는 XE에서 기본적으로 제공하는 document 모듈을 사용하는 방법을 설명합니다.

## 5.1 개요

XE는 모듈식 구조로 되어 있어서 이미 작성된 모듈을 사용해서 XE core의 기능을 쉽게 확장할 수 있습니다. 콘텐츠와 관련된 추가 기능을 구현할 때 가장 중요한 요소는 document 모듈입니다.

document 모듈에서 제공하는 기능은 다음과 같습니다.

- 콘텐츠를 생성하고 조회하는 기능
- 댓글 수와 조회 수, 다른 유용한 통계에 관한 정보
- 수정 이력
- 카테고리나 태그를 통해 콘텐츠를 쉽게 구성하는 기능
- 배치(batch) 편집
- XE의 다른 모듈과 쉽게 통합

이 모듈을 활용하는 방법을 더 자세히 알고 싶으면 콘텐츠를 저장하는 데 document 모듈을 사용하는 포럼, 위키, textyle, 이슈 트래커 등의 모듈을 참고하십시오.

## 5.2 document 모듈 작성

### 5.2.1 문서 생성

문서 생성 메서드는 documentController - ./modules/document/document.controller.php에 정의되어 있습니다. 예제는 다음과 같습니다.

```
$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);
```

모든 문서는 [DB플래그]_documents 테이블에 저장됩니다. 기본 필드 외에도, extra_var 기능을 사용해서 필요한 커스텀 필드를 쉽게 추가할 수 있습니다. extra_vars는 모듈 인스턴스를 기준으로 생성됩니다. 따라서, 이 모듈 인스턴스에 포함된 모든 문서는 extra_vars로 정의된 필드를 사용할 수 있습니다.

커스텀 필드의 이름과 타입 정보는 [DB플래그]_document_extra_keys 테이블에 저장됩니다. documentController에 있는 insertDocumentExtraKey 메서드를 사용해서 새 키를 추가할 수 있습니다. 새 키의 값은 [DB플래그]_document_extra_vars 테이블에 저장됩니다. documentController 클래스의 insertDocumentExtraVar 메서드를 사용해서 새 키의 값을 추가할 수 있습니다.

### 5.2.2 문서 속성

사용되는 문서 속성은 다음과 같습니다.

**표 5-1 문서 속성**

| 속성 | 설명 |
|---|---|
| document_srl | 문서의 고유한 ID |
| module_srl | 문서가 연결되는 모듈 인스턴스 |
| category_srl | 문서 카테고리의 ID. 문서 카테고리는 [DB 플래그]_document_categories 테이블에 저장됩니다. |
| lang_code | 문서의 언어 코드. 동일한 문서를 다른 언어로 된 여러 개의 버전으로 만들 때 사용됩니다. |
| is_notice | 문서에 중요 표시를 할 때 사용되는 속성. 예를 들면 문서 목록의 맨 위에 공지를 표시할 때 이 속성을 사용할 수 있습니다. |
| title | 문서 제목 |
| content | 문서 내용 |
| readed_count | 문서를 본 횟수 |

| 속성 | 설명 |
|---|---|
| voted_count | 문서의 추천 횟수. 이 속성은 포인트 모듈과 통합해서 구현합니다. |
| blamed_count | 문서의 신고 횟수 |
| comment_count | 문서에 연결된 댓글 개수 |
| trackback_count | 문서의 트랙백 개수 |
| uploaded_count | 문서의 첨부 횟수 |
| password | 비밀 문서에 사용. 비회원이 글을 쓸 때 비밀번호를 저장하고, 글을 수정하거나 삭제할 때 사용합니다. 비밀글은 글보기에서 사용됩니다. |
| user_id, user_name, nick_name, member_srl | 문서 소유자에 관한 정보 |
| tags | 문서 태그. 값을 쉼표(,)로 구분해서 저장함. |
| regdate | 문서가 생성된 날짜 |
| last_updated | 문서가 마지막으로 수정된 날짜 |
| ipaddress | 문서를 생성한 사용자의 IP 주소 |
| comment_status | 문서에 댓글 허용 여부(ALLOW: 허용, DENY: 제한) |
| status | 문서의 상태 값(PRIVATE: 비공개, PUBLIC: 공개, SECRET: 비밀글, TEMP: 임시 저장) |

이 속성들은 모두 [DB플래그]_documents 테이블의 필드를 나타냅니다. 문서 항목의 모델 클래스는 document.item.php입니다.

### 5.2.3 문서 URL

문서는 다양한 방법으로 접근할 수 있습니다.

우선, 다음과 같은 구조로 불변 주소(permalink)를 표시합니다.

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

XE의 모든 문서는 다음과 같이 사용자 친화적인 이름으로 접근할 수도 있습니다.

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

문서 제목이 매우 길거나 공백을 포함하고 있으면 관리자 제어판의 **정보관리 > 문서**에서 문서의 별칭을 정의할 수도 있습니다. 한 문서에 하나 이상의 별칭을 붙일 수 있습니다. 별칭으로 문서에 접근할 때 URL 구조는 다음과 같습니다.

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

위와 같은 내장된 문서 접근 방법 외에도, 커스텀 모듈에서 자신만의 뷰 메서드를 정의할 수 있습니다.

**참고**

위의 예제들은 XE를 설치할 때 mod_rewrite가 가능하게 설정되어 있어야 사용할 수 있습니다.

### 5.2.4 문서 카테고리

각 문서는 카테고리에 포함될 수 있습니다. 카테고리는 [DB플래그]_document_categories 테이블에 저장되며, 계층 구조로 만들 수 있습니다. 기본적으로는 비계층적입니다.

카테고리는 documentController와 documentModel 클래스를 사용해서 관리합니다. documentController 클래스는 카테고리 관리와 관련된 다음과 같은 메서드를 포함합니다.

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

documentModel 클래스는 카테고리 관리와 관련된 다음과 같은 메서드를 포함합니다.

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

### 5.2.5 문서 개정 이력

document 모듈은 문서의 개정 이력을 유지하는 메커니즘을 가지고 있습니다. 문서가 수정될 때마다 documentController 클래스의 updateDocument 메서드가 로그 엔트리를 자동으로 추가합니다.

개정 이력은 기본적으로 비활성화되어 있습니다. 개정 이력을 활성화하려면 문서 부분 설정 페이지에서 **히스토리 사용** 옵션을 선택해야 합니다.

개정 이력은 [DB플래그]_document_histories 테이블에 저장됩니다. documentModel 클래스에 있는 다음의 메서드를 사용해서 문서의 로그를 조회할 수 있습니다.

- getHistories

- getHistory

### 5.2.6 문서 조회

문서를 조회할 때 사용되는 메서드는 documentModel의 getDocumentList입니다. 이 메서드를 사용해서 다음과 같은 기준으로 문서를 필터링할 수 있습니다.

- 모듈 srl
- 카테고리
- 문서를 생성한 회원
- 제목
- 내용
- 태그
- 타입 – 공지, 비밀
- 조회 수, 추천 수 등
- 생성 날짜, 수정 날짜

# 6. API 레퍼런스

이 장에서는 XE의 전역 함수와 클래스별 함수를 설명합니다.

## 6.1 XE 전역 함수

XE의 전역 함수는 XE_ROOT/config/func.inc.php 파일에 정의되어 있습니다.

**debugPrint(mixed OBJECT)**

디버깅 함수.

__DEBUG__의 값은 [XE_ROOT]/config/config.inc.php 파일에 1 이상으로 정의되어야 합니다.

__DEBUG_OUTPUT__값에 따라 결과값을 얻는 방법을 선택할 수 있습니다.

- 0: 출력할 파일/_debug_message.php에 연결
- 1: HTML 바닥에 댓글로 출력(응답 타입이 HTML인 경우)
- 2: Firebug 콘솔에 출력(PHP >= 5.2.0. Firebug/FirePHP 플러그인 필요)

**instance getController(string MODULE_NAME)**

모듈의 Controller 인스턴스를 가져옵니다.

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

**instance getAdminController(string MODULE_NAME)**

모듈의 Admin Controller 인스턴스를 가져옵니다.

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

**instance getView(string MODULE_NAME)**

모듈의 View 인스턴스를 가져옵니다.

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

**instance getAdminView(string MODULE_NAME)**

Admin View 인스턴스 함수를 가져옵니다.

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

**instance getModel(string MODULE_NAME)**

모듈의 Model 인스턴스를 가져옵니다.

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

**instance getAdminModel(string MODULE_NAME)**

모듈의 Admin Model 인스턴스를 가져옵니다.

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

**instance getAPI(string MODULE_NAME)**

모듈의 API 인스턴스를 가져옵니다.

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

**instance getWAP(string MODULE_NAME)**

모듈의 WAP 인스턴스를 가져옵니다.

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

**instance getClass(string MODULE_NAME)**

모듈의 클래스 인스턴스를 가져옵니다.

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

**Object executeQuery(string QUERY_ID, stdClass PARAM)**

XML 쿼리를 실행합니다. 결과 데이터는 Object 클래스의 인스턴스로 반환됩니다. Object::toBool()가 FALSE이면 쿼리에 실패했음을 나타내고, TRUE이면 쿼리가 정상적으로 실행되었다는 뜻입니다.

select문의 결과 데이터는 Object::data 변수에 담아 객체에 반환됩니다.

**Object executeQueryArray(string QUERY_ID, stdClass PARAM)**

executeQuery()와 같은 기능을 하지만, Object:data 변수의 결과가 한 행이더라도 배열로 반환합니다.

**int getNextSequence()**

다음 시퀀스 번호를 가져옵니다.

XE는 내부적으로 하나의 시퀀스를 사용하며, member_srl, module_srl, document_srl 같은 모든 primary_key는 이 함수를 사용해서 설정합니다. 즉, [DB플래그]_documents 테이블에서 document_srl을 1씩 증가(auto increment)시키지 않고, 이 시퀀스 번호를 사용합니다.

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

URL을 생성합니다.

XE는 현재 요청 URL에서 주어진 파라미터의 값을 변경한 후 새로운 URL을 반환합니다. 만약 첫 번째 파라미터가 ''라면 XE는 주어진 파라미터의 값만 사용해서 새로운 URL을 생성합니다.

```
// 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://로 시작하는 URL을 생성합니다.

```
// 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 ...])**

인코딩되지 않은 URL을 생성합니다. getFullUrl()과 같은 기능을 합니다.

```
// 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 ...])**

이미 인코딩되었으면 중복으로 인코딩되지 않게 URL을 생성합니다.

```
// 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 ...])**

가상 사이트 URL을 생성합니다. 첫 번째 매개변수는 도메인이나 vid를 사용합니다.

```
// 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...])**

인코딩되지 않은 URL을 생성합니다. getSiteUrl()과 같은 기능을 합니다.

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

가상 사이트에 대해 http://로 시작하는 URL을 생성합니다.

**int ztime(string STR)**

YYYYMMDDHHIISS 형식의 시간값을 유닉스 시간으로 변경합니다.

**string getTimeGap(string DATE, string FORMAT)**

YYYYMMDDHHIISS 형식의 시간값을 현재 시간과의 차이(분/시)로 보여줍니다. 시간 차이가 하루 이상이면 FORMAT에 설정한 형식으로 보여줍니다.

**string getMonthName(int MONTH, bool SHORT)**

월 이름을 표시합니다.

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

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

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

YYYYMMDDHHIISS 형식의 시간값을 원하는 시간 형식으로 변경합니다.

```
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)**

문자열을 특정 크기로 자르고 문자열 뒤에 꼬리말(tail)을 추가합니다.

```
print r(cut str('All roads lead to XE', 3, '...'));
// result : All…
```

**string removeHackTag(string CONTENT)**

해킹 시도로 의심되는 코드를 삭제합니다.

**bool isCrawler(string AGENT)**

로그인 사용자 에이전트와 IP를 검사해서 크롤러인지 확인합니다.

## 6.2 Context 클래스

Context는 GET/POST의 값을 수신하고 변수와 다양한 정보를 템플릿에 전달합니다. 또한, 요청이 XMLRPC, JSON, GET/POST 중 어디에 해당하는지 식별합니다.

**Context::set(string KEY, mixed VALUE)**

템플릿에 전달될 변수를 설정합니다.

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

템플릿에서는 {$user_id}로 전달된 값을 출력할 수 있습니다.

**mixed Context::get(string KEY)**

요청(Request)에 전달될 변수나 설정 결과값을 조회합니다.

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

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

여러 개의 값을 한 번에 조회하고 stdClass에 반환합니다.

**stdClass Context::getRequestVars()**

요청으로부터 전달되는 변수를 stdClass로 반환합니다.

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

JS 파일을 템플릿에 추가합니다. 확장자가 js인 파일만 추가합니다.

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

CSS 파일을 템플릿에 추가합니다.

**Context::addJsFliter(string FILTER_NAME)**

XML로 작성된 필터를 템플릿에 로드합니다.

**Context::setBrowserTitle(string TITLE)**

HTML의 제목을 지정합니다.

**Context::loadJavascriptPlugin(string PLUGIN_NAME)**

자바스크립트 플러그인을 템플릿에 로드합니다.

**Context::addHtmlHeader(string HEAD)**

HTML의 <head>와 </head> 사이에 문자열을 추가합니다.

## 6.3 Extravar 클래스

Extravar 클래스는 보통 확장 변수와 게시판 같은 모듈에 사용됩니다.

**ExtraItem::setValue(string VALUE)**

확장 변수의 값을 지정합니다.

**ExtraItem::getValueHTML()**

확장 변수의 타입에 따라 HTML에 적합한 마크업이 된 상태로 확장 변수의 값을 반환합니다.

**ExtraItem::getFormHTML()**

확장 변수의 타입에 따라 HTML 결과 파일의 입력 폼을 출력합니다.

## 6.4 Mail 클래스

Mail 클래스는 XE에서 메일 전송을 담당합니다. XE는 서버가 메일을 전송할 수 있게 설정되어 있을 때만 메일을 전송할 수 있습니다.

**Mail::setSender(string NAME, string EMAIL)**

메일의 발신자를 지정합니다.

**Mail::getSender()**

Mail::setSender() 함수에서 지정한 발신자를 반환합니다.

- 발신자는 base64으로 인코딩하며, 발신자 이름이 존재하면 반환합니다.
- 발신자 이름이 존재하지 않으면 빈 문자열(' ')을 반환합니다.

**Mail::setReceiptor(string NAME, string EMAIL)**

메일의 수신자를 지정합니다.

**Mail::getReceiptor()**

Mail::setReceiptor() 함수에서 지정한 수신자를 반환합니다.

- 수신자는 base64으로 인코딩하며, 수신자 이름이 존재하면 반환합니다.
- 수신자 이름이 존재하지 않으면 빈 문자열(' ')을 반환합니다.

**Mail::setTitle(string TITLE)**

메일의 제목을 지정합니다.

**Mail::getTitle()**

base64로 인코딩된 메일 제목을 반환합니다.

**Mail::setContent(string CONTENT)**

메일의 본문을 지정합니다.

**Mail::replaceResourceRealPath(mixed MATCHES)**

본문에 포함된 이미지의 주소를 절대 경로로 변환합니다.

**Mail::getPlainContent()**

메일 본문을 텍스트로 반환합니다.

**Mail::getHTMLContent()**

메일 본문을 HTML 형식으로 반환합니다.

**Mail::setContentType(string MODE)**

메일 본문의 형식을 지정합니다. 기본값은 HTML 형식입니다.

**Mail::send()**

메일을 전송합니다. 메일을 전송하기 전에 Mail::setSender(), Mail::setReceiptor(), Mail::setContent() 함수를 사용해서 발신자, 수신자, 메일 본문을 지정해야 합니다.

**Mail::checkMailMX(string EMAIL_ADDRESS)**

메일 주소가 유효한지 검사합니다. 메일 주소가 올바르지 않으면 false를 반환합니다.

**Mail::isVaildMailAddress(string EMAIL_ADDRESS)**

메일 주소가 유효한지 정규 표현식으로 빠르게 확인합니다. 메일 주소가 유효하면 전달받은 변수를 변경하지 않고 그대로 다시 반환합니다.

## 6.5 Object 클래스

Object 클래스는 모듈 간 데이터를 주고받는 데 사용합니다. 모듈은 Object 클래스를 상속하고 error와 message, variables 변수를 이용해서 값과 상태를 교환합니다.

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

Object 생성자.

- ERROR: 에러 코드(이 값이 0이면 에러 아님)
- MESSAGE: 에러 메시지(이 값이 success이면 에러 아님)

**bool Object::toBool()**

Object가 에러인지 확인합니다. 반환값이 true이면 해당 객체는 에러가 아닙니다.

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

**Object::add(string KEY, mixed VALUE)**

KEY를 키로 하여 변수를 Object에 추가합니다.

**Object::adds(stdClass OBJECT)**

전달받은 stdClass에 속한 모든 변수를 Object에 추가합니다.

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

**mixed Object::get(string KEY)**

Object의 변수 중 키가 KEY인 변수를 반환합니다.

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

Object의 변수 중 키의 값이 KEY로 선언된 변수들을 stdClass로 묶어 반환합니다.

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

## 6.6 FileHandler 클래스

이 클래스는 폴더와 파일을 다루기 위한 메서드를 포함합니다.

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

SOURCE_DIR에서 TARGET_DIR로 폴더를 복사합니다.

- FILTER: 정규 표현식을 사용해서 폴더 내의 하위 폴더와 파일을 복사할 때 일치하는 파일은 복사되지 않습니다.
- TYPE: 옵션이 'force'이면, 하위 폴더에 있는 중복된 파일을 모두 덮어 씁니다.

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

SOURCE_FILE에서 TARGET_FILE로 파일을 복사합니다.

- FORCE: 옵션이 'Y'이면, 중복된 파일을 모두 덮어 씁니다.

**string FileHandler::readFile(string FILE_NAME)**

파일의 내용을 읽어서 반환합니다.

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

BUFFER의 내용을 파일에 씁니다.

- FILE_NAME: 저장될 파일
- BUFFER: 저장될 내용
- MODE: 'w'는 새로 저장, 'a'는 기존 파일의 끝에 내용 추가

**FileHandler::makeDir(string PATH)**

PATH의 폴더와 그 하위의 폴더를 재귀적인 방식으로 생성합니다.

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

**FileHandler::removeDir(string PATH)**

PATH의 폴더와 그 하위의 폴더를 재귀적인 방식으로 삭제합니다.

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

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

원격 파일을 받아 로컬에 저장합니다.

- URL: http://로 시작하는 경로를 입력합니다.
- TARGET_FILE: 저장될 파일

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

기존 이미지 파일을 이용해서 크기와 생성 방식(가로세로 비율 유지, 잘라내기)을 지정해 섬네일을 생성합니다.

- SOURCE_FILE: 원본 이미지 파일
- TARGET_FILE: 저장될 이미지 파일
- WIDTH: 저장될 이미지의 너비
- HEIGHT: 저장될 이미지의 높이
- FILE_TYPE: 저장될 이미지의 타입
- THUMBNAIL_TYPE: ratio, crop, 또는 thumbnail