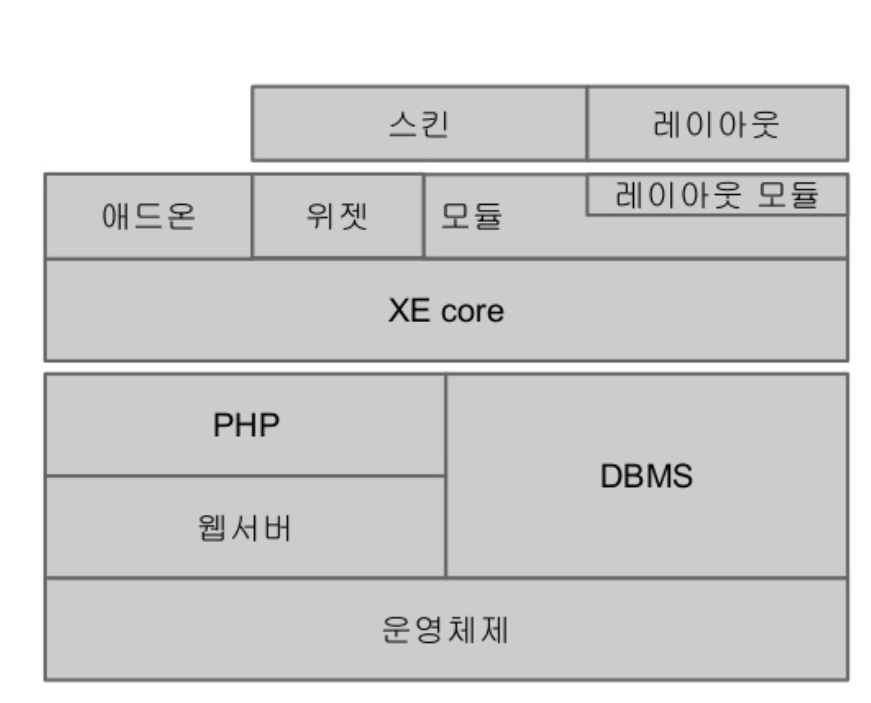
xe모듈 개념, 간단한 모듈 개발

XE구조

os, 웹서버 등 플렛폼 스택 위에 xe core가 위치 하며 모듈, 애드온, 위젯과 같은 구성요소가 xe core를 통해 실행되는 구조



\*Context

사전적 의미 : 맥락, 문맥

XE에서는 : XE구동에 필요한 여러 정보를 담고 조작할 수 있는 클래스

초기화

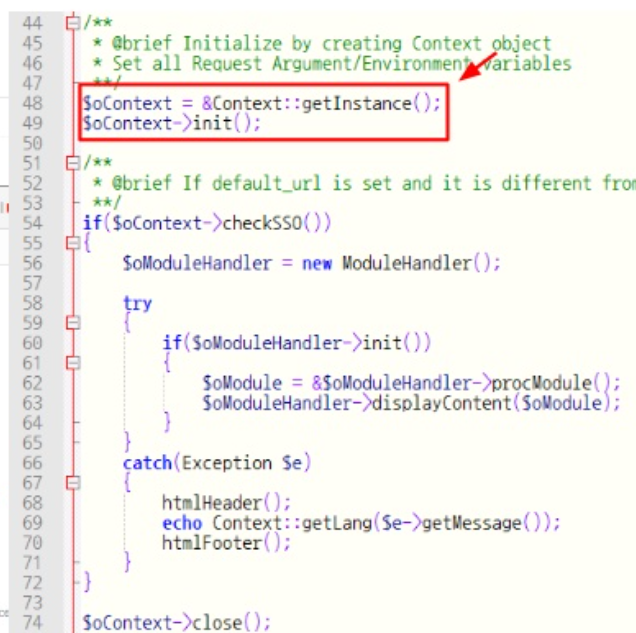
요청/응답 유형(GET,POST,XML,JSON)

요청 파라미터/업로드 파일

DB초기화

언어

인증정보



ModuleHandler

모듈을 선택하고 실행하기 위한 클래스

실행 모듈 결정

Module, mid, act, document\_srl기, module\_s기 등 파라미터(매개변수) 정보를 기반으로 모듈과 클래스(controller, view, model)를 결정

Module.xml

모든 모듈이 가지고 있어야 할 파일로 모듈의 퍼미션, 액션, 룰셋 정보를 가지고 있음

모듈 실행

퍼미션 체크, 룰셋 체크 후 모듈 실행

모듈 실행 시 Context의 정보를 이용하며 화면에 보여주고 싶은 정보 또한 Context에 세팅

출력 형식도 결정 지을 수 있음

ModuleObject

모든 모듈 클래스가 상속받는 상위 클래스

결과 출력

현재 모듈의 레이아웃/메뉴 정보 Context에 세팅

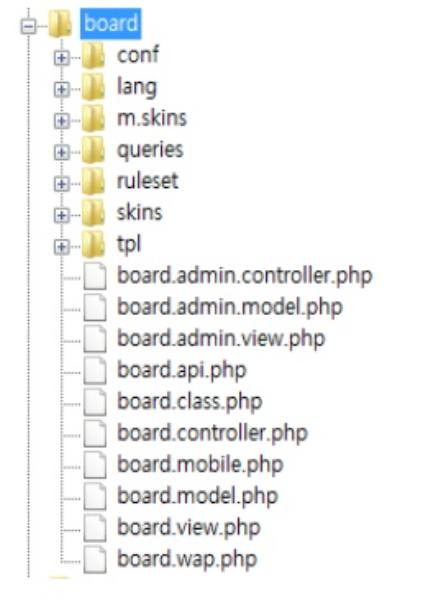
Context에 설정된 출력 형식에 따라 적합한 DisplayHandler를 선택한 후 결과를 출력

DisplayHandler

XE 실행 결과를 출력하기 위한 클래스

HTML/XML/JSON 출력이 가능

모듈 구조



Conf : 모듈 설명, 액션 권한 설정을 포함한 폴더

Info.xml : 모듈 제작자 정보와 설명

Module.xml:모듈 동작과 관련한 정보

Lang : 모듈에서 사용하는 언어 파일을 포함한 폴더

Lang.xml : 언어 파일

m.skins : 모바일 스킨을 포함하는 폴더

skin.xml : 스킨 정보와 설정 등을 담은 파일

-그 외 파일과 폴더는 자유롭게 구성할 수 있으며

관습적으로 css, img(images), js등의 폴더로 구성

Queries : DB조작을 위한 쿼리 파일을 담은 폴더

\*.xml : 쿼리 xml파일

-xe가 여러 DBMS지원할 수 있는 이유는 쿼리를 추상화(간략하게,핵심만)하여 xml로 표현하기 떄문

Ruleset : 유효성 검사를 위한 룰셋을 담은 폴더

\*.xml:룰셋 xml파일

-룰셋은 유효성 검사를 위한 규칙들을 정의한 것

-프론트엔드에서 js를 이용한 검사 뿐 아니라

모듈 액션 구동 단계에서도 룰셋을 적용할 수 있음

-프론트엔드 유효성 검사는 자용자 편의로 생각하고

무조건 서버 측 유효성 검사가 수행되어야 함.

Skins: PC 스킨을 담은 폴더

Skin.xml: 스킨 정보, 설정 등을 담은 파일

tpl : 템플릿 파일을 담은 폴더

-주로 관리자 페이지의 View를 위한 템플릿

-특별한 경우 스킨과 상관없이 동일한 템플릿도 사용

\*[NAME].class.php : 모듈의 최상위 클래스

[NAME].admin.controller.php : 관리자 컨트롤러 클래스

[NAME].admin.model.php : 관리자 모델 클래스

[NAME].admin.view.php : 관리자 뷰 클래스

[NAME].controller.php : 컨트롤러 클래스

[NAME].model.php : 모델 클래스

[NAME].view.php : 뷰 클래스

[NAME].mobile.php : 모바일 페이지 제공을 위한 클래스

[NAME].api.php : API제공을 위한 클래스

-MVC 모델 본래의 의미와는 약간 다른 형태

-Model: get계열 메소드

-view : 출력(disp) 계열 메소드

-controller: insert/modify/delete 계열 메소드

Document, Comment 모듈

Document/Comment 모듈 : XE에서 만들어지는 모든 “문서/댓글”을 관리하기 위한 모듈

모듈 제작 시에 자체적으로 DB스키마를 정의하여 데이터를 저장할 수도 있겠지만

Document/Comment 모듈을 사용하면 한 곳에서 통합 관리할 수 있고 최신글(댓글) 위젯 등에서도 우리가 만든 모듈의 글(댓글)을 아무런 수정 없이 사용할 수 있음.

모듈 구상

목표 : XE의 기본 요소들을 활용하는 간단한 모듈을 제작

요구사항

-Document, Comment 모듈 사용

-사이트 메뉴 편집과 호환

나의 게시판

목록 보기 사이트 메뉴 편집을 통해 추가/설정

내용보기

글 쓰기

댓글 쓰기

모듈 폴더 생성

XE(ROOT)/modules/myboard 폴더 생성

Info.xml 생성

MODULE ROOT/conf/info.xml 파일 생성

<?xml version="1.0" encoding="utf-8"?>

<module version="0.2">

    <title xml:lang="ko">나의 게시판</title>

    <description xml:lang="ko">내가 직접 만든 게시판</description>

    <version>1.0</version>

    <link>127.0.0.1</link>

    <category>service</category>

    <date>2020-11-09</date>

    <license link="http://licenseurl.com">FREE</license>

    <author email\_address="somej@naver.com" link="http://127.0.0.1">

        <name xml:lang="ko">woojungho</name>

    </author>

</module

Module.xml생성

MODULE/conf/module.xml 파일 생성

Module.xml

* 모듈 동작에 관한 정보를 가진 파일

<grants>

-권한 관련 정보

<permissions>

-액션 실행 퍼미션 관련 정보

<actions>

-모듈이 가진 액션 관련 정보

<?xml version="1.0" encoding="utf-8"?>

<module>

    <grants>

        <grant name="list" default="guest">

            <title xml:lang="ko">목록 보기</title>

        </grant>

        <grant name="view" default="guest">

            <title xml:lang="ko">내용 보기</title>

        </grant>

        <grant name="write\_document" default="guest">

            <title xml:lang="ko">글 쓰기</title>

        </grant>

        <grant name="write\_comment" default="guest">

            <title xml:lang="ko">댓글 쓰기</title>

        </grant>

    </grants>

    <permissions>

        <permission action="dispMyboardAdminList" target="manager" />

        <permission action="dispMyboardAdminInsert" target="manager" />

        <permission action="dispMyboardAdminInfo" target="manager" />

        <permission action="procMyboardAdminUpdateSimpleSetup" target="manager" />

        <permission action="procMyboardAdminInsert" target="manager" />

        <permission action="getMyboardAdminSimpleSetup" target="manager" />

    </permissions>

    <actions>

        <action name="dispMyboardAdminList" type="view" admin\_index="true" menu\_name="myboard" menu\_index="true" />

        <action name="dispMyboardAdminInsert" type="view" menu\_name="myboard" />

        <action name="dispMyboardAdminInfo" type="view" menu\_name="myboard" />

        <action name="procMyboardAdminUpdateSimpleSetup" type="controller" />

        <action name="procMyboardAdminInsert" type="controller" />

        <action name="getMyboardAdminSimpleSetup" type="model" simple\_setup\_index="true" />

        <action name="dispMyboardWrite" type="view" standalone="false" />

        <action name="dispMyboardContent" type="view" standalone="false" index="true" />

        <action name="procMyboardWrite" type="controller" standalone="false" ruleset="write" />

    </actions>

    <menus>

        <menu name="myboard">

            <title xml:lang="ko">나의 게시판</title>

        </menu>

    </menus>

</module>

Myboard.class.php 생성

MODULE/myboard.class.php 파일 생성

[MODULENAME].class.php

-XE의 모든 모듈 class 파일은 Module.Object를 상속받아야 함

[MODULENAME]::moduleInstall()

-xe 초기 설치/쉬운설치 시에 호출 됨

[MODULENAME]::checkUpdate()

-모듈 업데이트 여부를 체크하기 위해 호출 됨

[MODULENAME]:: moduleUpdate()

* 모듈 업데이트 시 호출 됨

[MODULENAME]::moduleUninstall()

-모듈 삭제 시 호출 됨

-모듈이 쉬운 설치에 올라가 있을 때만 삭제가 가능

메뉴 타입으로 등록

menu.getModuleListInSitemap after 트리거를 등록하여 모듈명을 파라미터로 받은 배열에 추가해 주면 메뉴 타입으로 등록 됨

트리거 : 특정 액션 이전이나 이후 어떠한 동작을 하고 싶을 때 사용하라 수 있음

예)

글 작성 완료 시 포인트 차감

글 삭제 시 관련된 데이터 DB에서 삭제

메뉴 타입으로 등록

MODULE/myboard.class.php 파일 수정

트리거 등록을 위한 $triggers 멤버 변수 추가

-트리거 정보를 가진 배열의 이름

-name:사용할 트리거 이름

-module : 트리거 발생 시 실행할 모듈 이름

-type : 메소드가 포함된 클래스 타입

-func : 트리거 발생 시 실행할 메소드 이름

-position : 해당 트리거 이전/이후 실행인지 여부

§ before 혹은 after만 가진 트리거 존재

moduleInstall()메소드에 트리거 등록 추가

-멤버 변수 $triggers 배열을 순회하면서 트리거를 등록

-XE 설치 시에 우리 모듈이 modules 안에 존재한다면 이 메소드가 실행

Module Controller

-모듈과 관련된 조작 메소드를 가지고 있다.

ModuleController::insertTrigger()

-트리거를 등록하는 메소드

checkUpdate() 메소드에 트리거 확인 추가

-$triggers를 순회하먀 등록되었는지 확인

-등록 되지 않았다면 true를 반환하여 관리자 페이지에 모듈 업데이트가 나오도록 함

ModuleModel

-모듈과 관련되니 get 계열 메소드를 가지고 있음

ModuleModel::getTrigger()

-트리거 정보를 받는 메소드

ModuleUpdate() 메소드에 트리거 등록 추가

-$triggers를 순회하며 등록되었는지 확인한 후 안되어 있다면 등록

moduleUninstall() 메소드에 트리거 제거 추가

-$triggers를 순회하며 트리거를 제거

-우리 모듈이 쉬운설치에 올라가 있다면 이 메소드를 통해 모듈을 삭제할 수 있음.

ModuleController::deleteTrigger()

-트리거를 제거하는 메소드

MODULE/myboard.model.php 파일 생성

[MODULENAME]Model

-해당 모듈의 get관련 메소드를 가진 클래스

-해당 모듈 클래스를 상속 받아야 함

triggerModuleListInSitemap()

-메뉴 타입으로 나오게 하기 위한 메소드

-myboard 클래스에서 menu

getModuleListInSitemap 트리거에 이 메소드를 등록했기 때문에 해당 트리거 발생 시 이 메소드가 호출됨

* 파라미터는 메뉴 타입으로 나올 수 있는 모듈 이름을 담은 배열로 반드시 레퍼런스 타입으로 받은 후 배열에 모듈 이름을 추가해 주어야함

