**ooooo 솔루션 도입 사업**

개발표준정의서

2020.10.25



DOC-KK-A06

문서

작성자(인)

작성자

승인자(인)

수행승인

1.0

버전

고객사(인)

고객승인

|  |
| --- |
| 관리본 개정이력표 |
| **문서명** | 개발표준정의서 |
| **버전** | **날짜** | **내용** | **작성자** |
|  |  |  |  |

목차

[1 공통 사항 7](#_Toc86999112)

[1.1 Coding 원칙 7](#_Toc86999113)

[1.2 Coding Style 7](#_Toc86999114)

[1.2.1 Indent 7](#_Toc86999115)

[1.2.2 Space 7](#_Toc86999116)

[1.2.3 Brace 7](#_Toc86999117)

[1.2.4 용어 7](#_Toc86999118)

[1.2.5 Logging 8](#_Toc86999119)

[1.2.6 File 명명 및 Line 8](#_Toc86999120)

[1.3 Message 8](#_Toc86999121)

[1.3.1 Client Message 8](#_Toc86999122)

[1.4 개발환경 구성도 9](#_Toc86999123)

[1.5 프로젝트별 폴더관리 9](#_Toc86999124)

[2 사용자 화면 10](#_Toc86999125)

[2.1 공통 10](#_Toc86999126)

[2.2 웹 디렉토리 구조 11](#_Toc86999127)

[2.3 html 12](#_Toc86999128)

[2.3.1 명명규칙 12](#_Toc86999129)

[2.3.2 Coding Style 12](#_Toc86999130)

[2.4 jsp 12](#_Toc86999131)

[2.4.1 명명규칙 12](#_Toc86999132)

[2.4.2 Coding Style 13](#_Toc86999133)

[2.4.3 주석 13](#_Toc86999134)

[2.4.4 User Interface 14](#_Toc86999135)

[2.5 Java script 14](#_Toc86999136)

[2.5.1 명명규칙 14](#_Toc86999137)

[2.5.2 Coding Style 15](#_Toc86999138)

[2.5.3 주석 15](#_Toc86999139)

[3 자바 클래스 16](#_Toc86999140)

[3.1 공통 16](#_Toc86999141)

[3.2 명명규칙 16](#_Toc86999142)

[3.2.1 Java package 16](#_Toc86999143)

[3.2.2 Java 라이브러리 17](#_Toc86999144)

[3.2.3 Java class 18](#_Toc86999145)

[3.2.4 Interface class 18](#_Toc86999146)

[3.2.5 Java method 18](#_Toc86999147)

[3.2.6 변수 및 상수 19](#_Toc86999148)

[3.2.7 \*.do 명명규칙 19](#_Toc86999149)

[3.3 Coding Style 19](#_Toc86999150)

[3.3.1 공통 19](#_Toc86999151)

[3.3.2 String 20](#_Toc86999152)

[3.3.3 블록문 20](#_Toc86999153)

[3.3.4 반복문 20](#_Toc86999154)

[3.3.5 조건문 20](#_Toc86999155)

[3.4 주석 21](#_Toc86999156)

[3.4.1 자바 클래스 주석 21](#_Toc86999157)

[3.4.2 method 주석 22](#_Toc86999158)

[3.4.3 변수 주석 23](#_Toc86999159)

[3.5 각종 설정(config)파일 명명규칙 24](#_Toc86999160)

[3.5.1 파일 종류별 구분 24](#_Toc86999161)

[3.6 예외처리 25](#_Toc86999162)

[3.6.1 예외 발생 시 처리해야 하는 내용 25](#_Toc86999163)

[3.6.2 레이어별 Exception 처리 25](#_Toc86999164)

[3.7 로깅처리 29](#_Toc86999165)

[3.8 트랜잭션 처리 29](#_Toc86999166)

[3.8.1 트랜잭션처리 전략 29](#_Toc86999167)

[3.9 DataSource 설정 30](#_Toc86999168)

[4 SQL 작성규칙 31](#_Toc86999169)

[4.1 공통 31](#_Toc86999170)

[4.2 명명규칙 31](#_Toc86999171)

[4.2.1 SQL File 31](#_Toc86999172)

[4.2.2 SQL Id 31](#_Toc86999173)

[4.3 Coding Style 31](#_Toc86999174)

[4.3.1 SQL 31](#_Toc86999175)

[4.4 주석 33](#_Toc86999176)

[4.4.1 SQL File 주석 33](#_Toc86999177)

[4.4.2 SQL 문 주석 34](#_Toc86999178)

[4.4.3 기타주석 34](#_Toc86999179)

[5 레이어별 개발 가이드 34](#_Toc86999180)

[5.1 공통사항 34](#_Toc86999181)

[5.1.1 Annotation 가이드 34](#_Toc86999182)

[5.2 DAO 클래스 개발가이드 36](#_Toc86999183)

[5.3 Service 클래스 개발가이드 40](#_Toc86999184)

[5.4 Controller 클래스 개발가이드 41](#_Toc86999185)

[5.5 JSP 개발가이드 44](#_Toc86999186)

[5.6 표준 Sequence Diagram 45](#_Toc86999187)

[6 시큐어 코딩가이드 45](#_Toc86999188)

[6.1 진단항목 및 보안수준 45](#_Toc86999189)

[6.1.1 진단 항목 45](#_Toc86999190)

[6.1.2 위험도 등급 47](#_Toc86999191)

[6.1.3 수행 방법 48](#_Toc86999197)

# 공통 사항

+

## Coding 원칙

주석은 모든 코드에 상세히 기술하는 것을 원칙으로 한다.

소스 코드는 불가피한 내용을 제외하고 원칙적으로 중복을 금지한다.

개발 및 테스트를 수행함에 있어 기능과 성능은 물론 보안에도 각별한 주의를 기울인다.

## Coding Style

코딩 스타일은 소스개발에 대한 규칙을 정의하는 것으로 표준화와 일관성을 확보하는 것을 목적으로 한다.

### Indent

Indent의 크기는 4로 한다.

### Space

한 줄에는 하나의 statement만 기술한다.

세미콜론, 콤마, 예약어 뒤에는 space를 둔다.

Unary operation은 space를 두지 않는다. (ex. i++;)

Binary operation은 양쪽에 space를 둔다. (ex. i = i + 1 ; )

괄호 안에 괄호가 있는 경우에는 괄호 사이에 space를 두지 않는다.

### Brace

 ‘{’는 기존의 ‘{’ 와 비교해서 indent(4-space)를 준다.

‘}’ 는 짝이 되는 ‘{’ 와 동일하게 indent(4-space)를 준다.

brace에 주석을 기입하는 경우 ‘//’ 주석을 사용한다.

### 용어

용어는 원칙적으로 용어사전을 준수하여 가급적 용어명에 맞는 영문을 사용한다.

하나의 단어를 사용하는 경우에는 용어영문명을 사용한다.

여러 개의 단어를 조합하여 사용하는 경우에는 영어약어명을 조합해서 사용한다.

사용될 서브시스템 약어는 다음과 같이 영문 3자로 정의한다.

|  |  |
| --- | --- |
| **코드** | **코드 설명** |
| CMN | 공통 |
| PRT | 포탈관리 시스템 |
| LMS | 인사관리 시스템 |

### Logging

Log는 반드시 Framework에서 제공한 Logger만을 사용한다.

Log 는 debug, info, warn, error로 구별하여 사용한다.

Log 는 반드시 발생 시간과 위치 그리고 내용을 포함한다.

Log 는 한 줄만 출력한다. (debug log 제외)

Debug log는 개발자가 개발 시에만 사용하고, 운영 중에는 사용하지 않는다.

Info log는 운영자에게 도움이 될 내용을 기록한다.

Warn log는 error 는 아니나 잠재적인error의 발생이 가능한 내용을 기록한다.

Error log 는 error code와 함께 error에 대한 내용을 기록한다.

### File 명명 및 Line

File 이름은 50자를 넘지 않도록 한다.

File 명은 용어사전의 내용을 기준으로 작성하며, 한글의 영문표기나 영문 이외의 특수기호는 사용하지 않는다.

## Message

메시지의 내용은 반드시 “~바랍니다.”, “~주십시오.”, “~하시겠습니까?” 와 같은 존칭어로 끝나는 종결자를 사용해야 하며, “~세요.”,”~주시오.” 와 같은 비 종결 언어 및 특수문자는 사용하지 않는다.

### Client Message

Java script는 alert기능을 이용한다.

## 개발환경 구성도



## 프로젝트별 폴더관리

프로젝트별로 아래의 예시대로 폴더/파일 관리를 하도록 한다.

**\* ex)** 프로젝트명 : **erp**,홈페이지코드 : **1**

 데이터베이스 : **oracle**, 지원스크린 : **반응형웹(resp)**

|  |  |
| --- | --- |
| **구분** | **경로** |
| java | /src/Main/java/upxframework/com/erp |
| jsp | /src/Main/webapp/WEB-INF/jsp/erp |
| xml (query) | /src/Main/resources/sqlmap/upxframework/oracle/erp |
| xml (sql-map-config) | /src/Main/resources/sqlmap/config/oracle/sql-map-erp-config.xml |
| image | /src/Main/webapp/images/resp/home/home1 |

# 사용자 화면

## 공통

절대 path사용을 원칙으로 한다.

directory에서 사용하는 system명과 sub-system명은 소문자만 사용한다.

File Encoding은 UTF-8을 기본으로 한다.

기본 해상도는 1024 X 768로 한다.

## 웹 디렉토리 구조



## html

### 명명규칙

Html은 jsp로 변환하여 사용하도록 한다.

jsp가 위치하는 논리적 위치는 [web-root]/WEB-INF/jsp를 기본으로 한다.

jsp 파일의 명명규칙은 가급적 *[용어사전]*.html로 한다.

iFrame 사용 시 window명은 if\_*[윈도우위치]*를 사용한다. 윈도우 위치는 top, bottom, left, right, Main, hidden을 사용하며 여러개가 필요한 경우 if\_*[윈도우위치]*\_#를 사용한다. (ex. if\_top\_1, if\_bottom\_9 )

pop-up 사용시 window명은 pop\_*[pop-up명]*을 사용한다.

pop-up은 스크롤이 없이 주소가 나타나지 않도록 하는 것이 원칙이나, 보안 설정에 의해서 주소가 나타나더라도 상하 scroll이 나타나지 않도록 한다.

필요에 따라서 scroll을 사용하도록 한다.

### Coding Style

스타일은 css를 사용한다.

css이 위치하는 논리적 위치는 [web-root]/css를 기본으로 한다.

## jsp

### 명명규칙

jsp가 위치하는 논리적 위치는 [web-root]/WEB-INF/jsp를 기본으로 한다.

\* jsp 파일을 WEB-INF 안에 두는 이유는 보안상 브라우저로 부터의 파일 접근을 막기 위함이다.

Jsp 파일의 명명규칙은 가급적 [용어사전].jsp로 한다.

용어사전을 사용할 경우에는 postfix를 제외하고는 가능한 full name을 활용하도록 한다.

용어사전의 postfix명명 규칙은 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Postfix명** | **설명** | **적용 예** | **비고** |
| List | 목록 | changeRequestList |  |
| Regist | 등록 | changeRequestRegist |  |
| Detail | 상세 | changeRequestDetail |  |
| Updt | 수정 | changeRequestUpdt |  |
| Popup | 팝업 | changeRequestPopup |  |
| Search | 조회 | changeRequestSearch |  |

Jsp 내의 변수명은 소문자로 시작하고 새로운 단어의 시작은 대문자를 사용한다.

session변수일경우 prefix로 ‘s\_’를 parameter변수인 경우 ‘p\_’를 사용한다.

Ex. s\_usrid, p\_lang\_cd

### Coding Style

Jsp 내에서 jsp include는 원칙적으로 사용하지 않는다.

\* 단, 공통 jsp파일(예, top, bottom 등)을 include 하는 경우에는 허용한다.

기본 스크립틀릿 사용을 지양하며, JSTL tag library를 사용하도록 한다.

스타일은 css를 사용한다.

폼을 서버로 전송할 경우 get방식은 사용하지 않고, post방식을 사용한다.

해당 jsp file에서만 사용하는 java script를 제외하고는 java script 소스를 jsp에 작성하지 않는다. (별도의 js파일에 java script를 작성하여 다른 jsp file과 공유하여 소스 중복을 최소화 하기 위함)

해당 jsp가 위치한 sub-system 외의 다른 sub-system의 java-script는 사용하지 않는다.

웹화면은 웹표준 및 접근성 지침을 준수하도록 한다.(Cross-Browsing 지원)

### 주석

Jsp 파일 주석

<%--

ClassName : login.jsp

Description : 사용자로그인을 담당

Modification information

수정일 수정자 수정내용

--------- ---------------- -----------------

2014.07.25 홍길동 최초생성

0214.08.24 김철수 그룹웨어결재연동기능 추가

Author : 각 팀 명(예. 기술개발) 홍길동

Since : 2014.07.25

--%>

### User Interface

테이블 열의 내용이 숫자인 경우 오른쪽 정렬한다. (ex. 나이, 은행잔고)

테이블 열의 데이터가 동일한 길이인 경우 가운데 정렬한다.(ex. 주민등록번호)

테이블 열의 데이터가 숫자도 아니고 동일한 길이도 아닌 경우 왼쪽 정렬한다.(ex. 계좌번호, 주소, 성명, 전화번호 등)

## Java script

### 명명규칙

Js 가 위치하는 논리적 위치는 [web-root]/js를 기본으로 한다.

js파일의 명명규칙은 가급적 [용어사전].js로 한다.

function명은 [동사][명사]를 사용한다.

(ex. checkValid(), 동사 및 명사는 가급적 용어사전을 기준으로 한다.)

변수명은 소문자로 시작하고 새로운 단어의 시작은 대문자를 사용한다. (ex. reqMsg)

### Coding Style

js내에서 다른 include는 원칙적으로 사용하지 않는다.

다른 js에서 공통으로 include하는 js는 허용

### 주석

js 파일 주석

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

파일명 : common.js

설명 : 회사소프트 공통 javascript

수정일 수정자 version funcion명

---------- ---------- ------------ --------------

2014.07.24 홍길동 1.0 최초 생성

2014.08.24 김갑수 2.0 fn\_check\_preiod

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function 주석

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

함수명 : checkPeriod

설 명 : 행정달력으로 받은 날짜값이 정상적인 허용범위에 속한 값인지를 검사

 검색시작일과 끝나는 일자의 크기 비교

인 자 : sdt( 시작일 control명) edt( 시작일 control명)

사용법 : checkPeriodBeforeSearch(input1, input2) 텍스트 입력값은 반드시 10자리로 구성되어야 함(2014-06-24)

작성일 : 2014-06-24

작성자 : 각 팀명 홍길동

수정일 수정자 수정내용

--------- ----------- ----------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# 자바 클래스

## 공통

Java class file은 필요 시 wrapper를 생성하여 사용함으로써 객체에 대한 추상화와 구현 시의 유연성을 확보한다. Java class 작성 시에 해당 package, file 및 method의 명명 규칙과 coding style을 준수하여 일관성 있는 개발이 되도록 한다.

## 명명규칙

### Java package

기본 패키지 명명규칙은 다음과 같다.

|  |  |
| --- | --- |
| **package 명** | **설 명** |
| upxframework | 개발프레임워크 총괄 도메인명 |
| 서브시스템명 | 예: 공통-common, 역량진단-comtc, 프레임워크코어-core, 컨텐츠관리-lcms, 학습관리 – lms, 포탈관리- portal, 집단지성 - social ) |
| 용도별 클래스 | common |
| 용도별 클래스 | common | 각 서브시스템별로 공통으로 사용하는 클래스를 가지고 있는 패키지예.upxframework.com.common upxframework.com.common.web |
| exception | exception : exception 클래스를 가지고 있는 패키지. 실행 환경의 에러처리 패키지 서비스를 제공하는 패키지를 이용한다.( upxframework.com.core.exception ) |
| service | service클래스를 가지고 있는 패키지예. upxframework.com.lms.member.service |
| service.impl | 서비스 구현 클래스와 DAO 클래스를 가지고 있는 패키지예. upxframework.com.lms.member.service.impl |
| web | controller 클래스를 가지고 있는 패키지예. upxframework.com.lms.member..web |

기타 패키지는 위의 규칙에 따라 용도별 클래스를 추가 확장하여 부여하도록 한다.

### Java 라이브러리

Library 명명 규칙은 Maven의 규칙에 따라 부여하며

[*GroupID.ArtifactID]*-*[version]*.확장자를 사용한다.

ex) upxframework.com.security-1.0.jar

개발이 수행될 때는 SNAPSHOT버전을 이용하고, 향후 통합테스트가 완료되어 정식버전이 배포가 될 경우에는 1.0부터 시작하도록 한다.

### Java class

Java file 의 명명규칙은 가급적 용어사전을 사용하고, [용어사전][postfix].java로 한다.

(ex. LoginCheckService.java)

postfix는 framework에서 정의하는 class요소에 대한 구분으로 해당 java class의 성격을 나타낸다.

(ex. postfix : Controller, Wrapper, Service, ServiceImpl, DAO, Test 등)

\* Test는 테스트 클래스 및 관련 xml파일에 붙는 postfix임

### Interface class

Interface는 java 클래스명과 동일한 방법으로 부여된다.

### Java method

Java method의 명명규칙은 [동사][용어사전]으로 한다.

※ 클래스 명명은 필수로 하되 메소드 명명규칙은 용어사전을 최대한 준용하도록 한다.

첫 글자는 소문자를 사용하며 이후 용어의 첫 글자만 대문자를 사용하며, 특수문자는 사용하지 않는다.

method에서 사용하는 동사는 다음 표와 같이 명명한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **구 분** | **유형** | **동사(prefix)** | **비고** |
| business처리 관련 | 내용검증 | validate |  |
| 조건확인 | check |  |
| 검색 | search |  |
| 연계 | contact |  |
| action | action |  |
| 파일관리-읽기 | read |  |
| 파일관리-쓰기 | write |  |
| data처리관련(Controller,Business,DAO 공통) | 등록 | insert |  |
| 조회(단건) | select |  |
| 조회(멀티건) | select | postfix로List를 사용 |
| 수정 | update |  |
| 삭제 | delete |  |
| 등록/수정 | merge |  |
| 등록/수정/삭제 | multi | 작업을 동시에 수행하는 경우 |
| Value Object(model) | 값읽기 | get |  |
| 값설정 | set |  |

### 변수 및 상수

Java 변수의 명명규칙은 가급적 [용어사전]을 조합하여 30자 이내로 명명한다.

첫 글자는 소문자를 사용하며 이후 용어의 첫 글자만 대문자를 사용하며, 특수문자는 사용하지 않는다.

loop index에서 사용하는 변수는 i,j,k,l,x,y,z 등을 (관용적으로) 사용할 수 있다.

일반 변수는 "\_" 또는 "$" 사용을 하지 않는다.

(단, 데이터베이스의 속성명을 그대로 사용하는 경우에는 ‘\_’ 사용을 허용)

상수는 Static Final 변수는 가급적 용어사전을 사용하여 대문자로만 작성하며 단어 사이는 '\_'를 사용하여 구분한다. (단, SUCEESS=1, FAIL=0으로 정의한다.)

### \*.do 명명규칙

Controller의 매핑정보를 관리하는 \*.do의 명명규칙은 다음과 같다.

* “/서브시스템/…/동사명+용어사전.do”(단, 동사명은 add, select, get, remove 등의 의미있는 단어 사용)

ex. /cop/bbs/adm/addBBSMaster.do

## Coding Style

### 공통

코드는 한줄에 하나의 문장만 기술한다.

Pubic 변수는 사용하지 않는다.

변수 선언 시 class type별로 모은다.

변수 선언 시에는 항상 초기화를 한다. (ex. String sampleString = null;)

배열 선언은 [class][] [variable-name]로 한다.(ex. String[] sampleArray = {“1”,”2”};)

배열 선언 순서

* Constant 변수
* Private 변수

Protected method는 사용하지 않는다.

method선언 순서

* Constructor (default 기술 후 argument 가 적은 순으로)
* Main method
* Private method
* Public method

Return 값에는 연산을 수행하지 않는다.

### String

String 의 연결 시에는 StringBuffer를 사용한다.

### 블록문

내용이 없는 블록문은 사용하지 않는다.(꼭 사용해야 하는 경우 log 라도 입력한다.)

### 반복문

반복문 내는 가급적이면 신규변수를 생성하지 않는다.

반복문 조건절에는 연산이 들어가지 않도록 한다.

### 조건문

조건문은 가급적 3중을 초과해서 사용하지 않는다.

조건작성시 필터링가능성이 높은 조건부터 앞쪽에 기술하며,

문자리터럴과 변수를 비교시(equals) 문자리터럴을 앞쪽에 기술한다.

## 주석

### 자바 클래스 주석

주석은 아래의 표준을 준수하여 작성하여 자바 문서화 주석(javadoc)으로 생성 및 사용한다.

클래스선언부상단, 멤버변수상단, 생성자상단, 멤버메소드상단에서 /\*\* 입력후 Enter키 입력후 주석작성

\* 클래스선언부상단 : 클래스설명, 작성자, 최초작성일을 입력한다.

\* 멤버변수상단 : 멤버변수 설명

\* 생성자상단 : 파라미터와 생성자 설명

\* 멤버메소드상단 : 메소드설명과 파라미터설명, 리턴에 대한 설명입력

클래스(인터페이스)명, 클래스 설명, 작성자, 최초작성일을 명시한다.

### method 주석

/\*\*

\* 사용자 계정을 처리하는 비즈니스 구현 클래스

\* NOTE : Exception 종류를 EgovBizException, RuntimeException 에서만 동작한다.

\* fail.common.msg 메시지키가 MessageResource에 정의 되어 있어야 한다.

\* @author 개발팀 홍길동

\* @since 2014.07.25

\*/

interface class 는 method 주석을 기재하지 않는다.

Template

### 변수 주석

/\*\*

\* userInfo 사용자ID에 해당하는 사용자명을 조회

\* @Param userID 검사항목에 대한 구분자

\* @param role 사용자 권한정보 구분

\* @return 사용자명

\* @exception MyException

\*/

Public String userInfo(String usrID, String role)

Class 변수 주석 template

/\*\* 이름 \*/

private String name;

/\*\* 객체 값 \*/

private Object val = null;

Method 내의 변수 주석 template

int ilevel; // indentation level

int iSize; // size of table

Object currentEntry //currently selected table entry

## 각종 설정(config)파일 명명규칙

서블릿 프로그램 혹은 스프링 컨테이너에서 사용하는 config파일에 대한 종류 및 관련 명명규칙을 정의한다.

### 파일 종류별 구분

#### Spring 공통 설정파일

Spring 관련 공통 설정 파일은 /src/Main/resources/spring/밑에 위치하며 다음과 같이 명명규칙을 준용하여 사용하도록 한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **설명** | **명명규칙** | **사용샘플** |
| AOP관련 설정 | AOP관련 Aspect 공통 설정 정보를 관리한다. | context-aspect.xml | 예. context-aspect.xml |
| common 설정 | 공통 설정 정보를 관리한다. | context-common.xml | 예.context-common.xml |
| datasource 설정 | 데이터소스 설정정보 를 관리한다. | context-datasource.xml | 예.context-datasource.xml |
| sqlMap 설정 | iBATIS SQL Map관련 설정정보를 관리한다. | context-sqlMap.xml | 예. context-sqlMap.xml |
| 트랜잭션 설정 | 트랜잭션 설정정보를 관리한다. | context-transaction.xml | 예.context-transaction.xml |

1. Springmvc 패키지(WEB-INF/config/springmvc)

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **설명** | **명명규칙** | **사용샘플** |
| Dispatcher Servlet 설정 | dispatcher servlet 관련 설정정보를 관리 한다.(SimpleMappingExceptionResolver,ViewResolver 등의 관련 클래스 정보) | dispatcher-servle t.xml | 예.dispatcher-servlet.xml |

#### SQL Map 설정파일

SQL Map 은 iBATIS프레임워크에서 사용하는 SQL 문을 담고 있는 것이다.

SQL Map 파일의 위치는 “/src/Main/resources/sqlmap/config/[DB종류]/” 경로에 위치한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **설명** | **명명규칙** | **위치** |
| 공통파일 | sqlmap파일의 위치를 담고 있는 메타정 보 관리 파일 이다. | sql-map-common-config.xml | /src/Main/resources/sqlmap/upxframework/[DB종류]/../\*.xml |
| 업무별SQLMAP 파일 | 각 업무에서SQL정보를 담 고 있는 파일 이다. | sql-map-[서스시스템명]-config.xml | /src/Main/resources/sqlmap/ upxframework /[DB종류]/../\*.xml |

## 예외처리

### 예외 발생 시 처리해야 하는 내용

해당 Exception 에 대한 Error Log 기록(파일 혹은 console)한다.

CUD transaction 이 수행중인 경우에는 트랜잭션 서비스와 연계되어 rollback을 수행한다.

해당 Exception 에 대해서 사용자에게 message를 제공한다.

Checked Exception 이 발생하는 경우에는 사용자가 인지할 수 있는 메시지를 페이지에 출력한다.

### 레이어별 Exception 처리

#### DAO 단에서 Exception 처리

DAO 단에서는 스프링에서 제공하는 SqlMapclientDaoSupport를 상속받아 사용하면 SQLException 대신 스프링이 제공하는 예외 클래스(Runtime Exception인 DataAccessException)로 변환해 주기 때문에 별도의 SQL 관련 Exception로직을 사용하지 않는다.

#### Service단에서 Exception 처리

Service단에서 DAO에서 Exception을 받아서 Exception을 처리할 수 있다.

프레임워크에서 추가로 제공하는 Exception은 다음과 같다. (추후 적용)

|  |  |  |
| --- | --- | --- |
| **구분** | **설명** | **사용규칙** |
| EgovBizException | 업무에서 **Checked Exception**인 경우 에 공통으로 사용하는 Exception이다. 개발자가 특정한 오류에 대해서 throw 하여 특정 메시지를 전달하고자 하는 경우에는 processException() 메소드를 이용하도록 한다. | Exception을 강제로 throw하고자 하는 경우에AbstractServiceImpl:processException(“메시지코드”)을 이용한다. |
| ExceptionTransfer | AOP기능을 이용하여**Unchecked Exception**인 경우 ServiceImpl 클래스 에서 Exception이 발생한 경우 (after-throwing인 경우)에 trace()메소드에서 처리한다.내부적으로 EgovBizException인지 runtimeException(ex.DataAccessException)인지 구분하여 throw한다.참조) ExceptionTransfer는 내부적으로 DefaultExceptionHandleManager 클래스에 의해서 정의된 패턴에 대해서Handler에 의해서 동작한다. | context-aspect.xml에 AOP로 설정되어 있다.DataAccessException발생 시 메시지 코드는 “fail.sql.msg" 메시지 코드로 매핑되어 있다. |
| upxframework.com.core.exception.**AccessException** | 접근을 제한하고자 하는 경우에 사용한다.errorMsg.jsp 로 이동하여에러메세지를 출력한다. | throw new AccessException("에러메세지"); |
| upxframework.com.core.exception.**AuthException** | 접근권한이 없는 경우에 사용한다.errorMsg.jsp 로 이동하여에러메세지를 출력한다. | throw new AuthException ("에러메세지"); |
| upxframework.com.core.exception.**PageNotFoundException** | 존재하지 않는 .do 에 접근시 발생한다.error.jsp 로 이동하여에러메세지를 출력한다. | throw new PageNotFoundException ("에러메세지"); |
| upxframework.com.core.exception.**SessionException** | 세션이 유효하지 않은경우에 사용한다.errorSession.jsp로 이동하여에러메세지를 출력한다. | throw new SessionException ("에러메세지"); |
| 그 외의 예외 | Error.jsp로 이동하여에러유형을 출력하도록 한다.Ex) java.lang.NullPointerException |  |

#### Controller 단에서 Exception 처리

Controller단에서 처리하는 Exception은 스프링에서 제공하는

SimpleMappingExceptionResolver(org.springframework.web.servlet.handler.SimpleMap pingExceptionResolver)를 이용하게 된다.

Controller에서 Error가 발생하면 dispatcher-servlet.xml에서 설정한 방법과 같이 해당

JSP로 포워딩 되도록 한다. 에러의 유형은 더 추가될 수 있다.

|  |  |  |
| --- | --- | --- |
| **구분** | **설명** | **페이지명** |
| 기본 에러페이지 | 기본적인 에러 발생 시에 표시되는 화면이다.(특수 에러인 경우에는 아래 참조) | error.jsp |
| 데이터 처리 오류(org.springframework.dao. DataAccessException) | 데이터 처리관련 Exception이 발생한 경우에 표시되는 화면이다. | error.jsp |
| 트랜잭션 처리 오류 |  |  |
| (org.springframework.tran saction.TransactionException) | EgovBizException이 발생한 경우 에 표시되는 화면이다. | error.jsp |
| HTTP 세션오류(org.springframework.web.HttpSessionRequiredException) | HTTP 세션이 유효하지 않을 경우 표시한다. | error.jsp |
| 접근오류upxframework.com.core.exception.**AccessException** | 접근을 제한하고자 하는 경우에 사용한다. | errorMsg.jsp |
| 접근권한 오류upxframework.com.core.exception.**AuthException** | 접근권한이 없는 경우에 사용한다. | errorMsg.jsp |
| 존재하지않는 URL(\*.do)upxframework.com.core.exception.**PageNotFoundException** | 존재하지 않는 .do 에 접근시 발생한다. | error.jsp |
| 유효하지않은 세션upxframework.com.core.exception.**SessionException** | 세션이 유효하지 않은경우에 사용한다. | errorSession.jsp |

## 로깅처리

로깅은 기본적으로 SLF4J + log4j라이브러리를 이용한다.

Logging은 유형에 따라 “INFO, DEBUG, WARN, ERROR" 레벨로 나누어 저장한다.

예를들어, DEBUG 레벨로 메시지를 Logging하고자 한다면 다음과 같이 작성하도록 한다.

## 트랜잭션 처리

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

protected Logger LOGGER = LoggerFactory.getLogger(ResAuthServiceImpl.class);

LOGGER.debug("XXXXXX");

LOGGER.debug("XXXXXX: {}", "XXXXX");

### 트랜잭션처리 전략

트랜잭션 처리는 여러 방식 중에 스프링의 선언적 트랜잭션 처리 방식을 적용한다. Annotation을 사용하지 않는 이유는 트랜잭션 처리가 다른 메소드와의 관계를 한 눈에 파악하는 것이 유리하기 때문이다.

트랜잭션 설정은 context-transaction.xml 파일에 명시되어 있다.

선언적 트랜잭션 시 AOP기능을 이용하여 통합 처리가 가능하다.

다음은 관련 설정파일(context-transaction.xml)의 샘플 내용이다.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?><beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.0.xsd">**// 트랜잭션 관리자를 설정한다.**<bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"> <property name="dataSource" ref="dataSource"/> < /bean >**// 트랜잭션 Advice를 설정한다.**<tx:advice id="txAdvice" transaction-manager="txManager"> < tx:attributes >**<tx:method name="\*" propagation=REQUIRED” rollback-for="Exception"/>** < /tx:attributes >< /tx:advice >**// 트랜잭션 Pointcut를 설정한다.**< aop:config ><aop:pointcut id="requiredTx" expression="**execution(\* upxframework.com..impl.\*Impl.\*(..))**"/><aop:advisor advice-ref="txAdvice" pointcut-ref="requiredTx" /> < /aop:config >< /beans > |

## DataSource 설정

DBMS와 연동하기 위해서는 DataSource를 설정해야 한다.

DataSource 설정은 WAS의 기능을 이용하는 방법과 Spring의 기능을 이용하는 방법이 있는데, 본 프로젝트에서는 WAS의 기능을 이용하여 JNDI를 이용하는 것을 기본으로 한다.

Spring의 DataSource 기능은 커넥션 풀을 이용하는 기능, JNDI를 이용하는 기능, DriverManager를 이용하는 기능이 있는데, 성능을 보장하기 위하여 커넥션 풀을 이용하도록 한다.

드라이버 클래스는 **net.sf.log4jdbc.DriverSpy** 기능을 이용하는데 이는 이 클래스가 요청 SQL을 중간에 가로채서 로그로 남기는 기능이 있기 때문이다. 개발단계에 SQL을 수행되는 SQL을 조회함으로써 개발생산성을 향상할 수 있다.

# SQL 작성규칙

## 공통

SQL은 개발자간 기술의 차이가 크고 Framework을 사용하여 개발하는 application에 비해서 표준화가 어려우므로 Coding 표준으로 제정된 내용을 준수하여 품질의 일관성을 확보한다.

## 명명규칙

### SQL File

3.5.1의 서브시스템명의 하위기능 폴더명을 사용하도록 한다.

Ex. Upxframework/com/lms/member => member.xml

### SQL Id

가급적 DAO클래스의 method명을 사용한다.(DAO의 method는 하나의 SQL만을 수행한다.)

Ex. selectList

## Coding Style

### SQL

|  |
| --- |
| <?xml version=*'1.0'*?><!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN" "http://www.ibatis.com/dtd/sql-map-2.dtd"><!-- SQL File Name : test.xml Description : 테스트용입니다.--><sqlMap namespace=*"test"*> <!-- 테스트용1 --> <select id=*"test.selectMemberInfoTest1"* resultClass=*"cmap"* parameterClass=*"String"*> <![CDATA[ /\* *test.selectMemberInfoTest1* \*/ SELECT EMAIL FROM TB\_LMS\_MEMBER WHERE USERID = #p\_usrid# ]]> </select>   <!-- 테스트용2 --> <select id=*" test.selectMemberInfoTest2"* resultClass=*"cmap"* parameterClass=*"cmap"*> <![CDATA[ /\* *test.selectMemberInfoTest2*\*/ SELECT NAME FROM TB\_LMS\_MEMBER WHERE USERID = #p\_usrid# ]]> </select>  <!-- 테스트용3 --> <select id=*" test.selectMemberListTest"* resultClass=*"cmap"* parameterClass=*"cmap"*>  <include refid=*"pageingJqTop"*/> <![CDATA[ /\* *test.selectMemberListTest* \*/ SELECT USERID, NAME, EMAIL FROM TB\_LMS\_MEMBER   ]]>  <include refid=*"pageingJqBottom"*/> </select> </sqlMap> |

SQL 예약어는 대문자를 사용한다.

SELECT, UPDATE시 컬럼 나열은 가급적 1줄당 1개씩 소문자로 작성을 기본으로 하며, 컬럼이 많은 경우에 한하여서 예외로 허용한다.

SELECT의 들여쓰기는 SELECT를 기준으로 FROM, WHERE, INTO등의 시작점을 맞춥니다. 단 GROUP BY나 ORDER BY는 SELECT(6자)보다 더 긴 관계로 BY 이후 스페이스 1칸을 띄우고 관련내용을 기술한다.

FROM 절의 테이블명은 1줄당 1개씩 소문자로 작성하는 것을 기본으로 하며, 테이블이 많은 경우에 한하여서, 예외로 허용한다.

WHERE 절의 조건은 1줄당 1개의 조건씩 작성한다.

INSERT 문장의 경우 전체 컬럼에 대해서 insert를 하더라도 컬럼명을 반드시 적어준다.

DML의 들여쓰기는 UPDATE, DELETE, INSERT를 기준으로 SET, FROM, WHERE등의 시작점을 맞추고, 컬럼 구분은 가장 길이가 긴 컬럼의 쉼표(,)나 등호(=등)을 기준으로 다른 컬럼을 맞춘다.

## 주석

### SQL File 주석

Template

<!--

 SQL File Name : UpxUserLogin\_SQL.xml

 Description : 사용자로그인을 담당

-->

### SQL 문 주석

Template

<!-- 민원코드를 조회하기 위한 쿼리 -->

<select id=*" test.selectMemberListTest"* resultClass=*"cmap"* parameterClass=*"cmap"*>

### 기타주석

Xml 내의 queryId별 설명 주석은 ‘<!-- -->’ 을 사용한다.

쿼리내에 주석사용시 /\* 내용 \*/ 를 사용하도록 한다.

/\*내용\*/ 🡸 이렇게 작성하면 안되고,

/\* 내용 \*/ 🡸 이렇게 공백을 앞뒤로 넣어야한다.

# 레이어별 개발 가이드

## 공통사항

### Annotation 가이드

스프링 프레임워크를 이용할 때 빈의 설정정보 등을 관리할 때 사용할 수 있는 기능이

Annotation 기능이다.(JDK 1.5 이상에서 지원)

Annotation의 기능은 클래스, 메소드 등의 상단에 필요한 정보를 설정함으로써 Spring 프레임웍이 Container를 구동할 때 해당 정보를 설정정보가 아닌 클래스 정보를 파싱할 때 활용할 수 있다.

본 프로젝트는 주요 설정정보를 Annotation으로 구현하는 것을 권장한다. 설정정보를 이 용하는 것과 Annotation 사용하는 것이 상호 장점과 단점을 갖고 있으나 개발 단계의 효율성을 더 보장할 수 있기 때문이다.

Annotation 태그는 org.springframework.beans.factory.annotation 패키지에 정의되어 있다.

다음은 주요 Annotation 사용 방법이다.

|  |  |  |
| --- | --- | --- |
| **태그명** | **설명** | **사용 예** |
| @Resource | 어플리케이션에서 필요로 하는 자원을 자동으로 연결할 때 사용된다.스프링에서는 의존하는 빈 객체를 전달할 때 사용된다. | //sqlMap객체에 com.ibatis.sqlmap.client.SqlMapClient를 전달하고자 할 경우@Resource(name = "sqlMapClient")SqlMapClient sqlMap; |
| @Component | 일반적인 클래스를 빈으로 자동으로 등록하고자 하는 경우 사용한다. | // systemTest이름으로 빈에 자동설정 하고자 하는 경우@Component public class SystemTest {} |
| @Controller | Controller 클래스로 spring이 등록한다. | // egovCategoryController 클래스를 Controller로 등록@Controller public classCommonBConteroller{} |
| @RequestMapping | HTTP Request에 대한 매핑 정보를 바로 해당 Controller 클래스의 메소드로 지정한다. | // \*.do 요청을 해당 함수에서 처리@RequestMapping("/back/common/Main.do") public String Main(…){ |
| @RequestParam | HTTP 요청 파라미터를 매핑하는 경우에 사용한다. | // selectedId 요청 파라미터를 id 로 전달public String updateCategoryView(@RequestParam("selectedId"), String id, ModelMap model){} |
| @Service | 서비스 클래스의 구현체를 지정 한다. | // Service 클래스를 구현하는 Impl 클래스임을 명시@Service("common.commonService") public class CommonServiceImpl |

\* context-common.xml파일에서는 다음과 같은 설정을 통하여 클래스를 자동으로 스캔하여 빈으로 등록한다.

 - <context:component-scan base-package=”upxframework”/>

## DAO 클래스 개발가이드

DAO는 iBATIS 프레임워크의 기능을 이용하여 Data의 Access를 처리하는 클래스이다.

업무 DAO는 EgovAbstractDAO를 상속받아 개발한다.

EgovAbstractDAO는 스프링에서 제공하는 SqlMapClientDaoSupport 클래스를 상속받은 공통기능을 추상화한 클래스이다.

DAO 내에서는 스프링 프레임워크에서 관련 Exception을 자동으로 Throw해 주므로 별도 의 Exception처리가 필요한 경우를 제외하고는 데이터 처리 오류관련 Exception을 처리 하지 않아도 된다.

DAO에서는 DataSource의 생성, Connection생성, 관련 자원의 해제 등의 로직을 사용하지 않아도 된다. 이는 SqlMapClientDaoSupport 프레임워크 클래스에서 이러한 처리를 모두 담당하기 때문이다.

다음은 DAO 클래스의 구현 샘플이며, 이를 기본으로 확장하여 사용하도록 한다.

|  |
| --- |
| package upxframework.com.common.service.impl;import java.util.List;import java.util.Map;import javax.swing.Box;import org.springframework.stereotype.Repository;import upxframework.com.core.data.DataSet;import upxframework.com.core.mvc.dao.ParentDAO;import upxframework.com.core.persist.ListDTO;@Repository(value = "common.commonDAO")public class CommonDAO extends ParentDAO { public Map selectTest(String usrid) throws Exception { return queryForMap("selectMemberInfoTest1", usrid); } public DataSet selectTest(Map map) throws Exception { return queryForDataSet("selectMemberInfoTest2", map); }  public ListDTO selectListTest(Map map) throws Exception { int pageno = Integer.*parseInt*((String)map.get("page")); int listscale = Integer.*parseInt*((String)map.get("rows")); return queryForPageList("selectMemberListTest", map, pageno , listscale); }}} |

DAO에서는 iBATIS기반의 기본적인 처리유형(템플릿)을 제공한다. 각 클래스는 EgovAbstractDAO를 상속받은 ParentDAO의 다음 기본 기능을 호출하여 개발할 수 있다.

|  |  |
| --- | --- |
| **메소드 Signature** | **설명** |
| public Object insert(String queryId, [Object parameterObject]) | 입력 처리 SQL mapping 을 실행한다.@param queryId - 입력 처리 SQL mapping 쿼리 ID@param parameterObject - 입력 처리 SQL mapping 입력데이터를 세팅한 파라메터 객체(보통 VO 또는 Map) @return 입력 시 selectKey를 사용하여 key를 딴 경우 해당 keyparameterObject가 없는경우의 오버로딩함수도 제공 |
| public int update(String | 수정 처리 SQL mapping 을 실행한다. |
| queryId, [Object parameterObject]) | @param queryId - 수정 처리 SQL mapping 쿼리 ID@param parameterObject - 수정 처리 SQL mapping 입력데이터(key 조건 및 변경 데이터)를 세팅한 파라메터 객체(보통 VO 또는 Map)@return DBMS가 지원하는 경우 update 적용 결과 countparameterObject가 없는경우의 오버로딩함수도 제공 |
| public int delete(String queryId, [Object parameterObject]) | 삭제 처리 SQL mapping 을 실행한다.@param queryId - 삭제 처리 SQL mapping 쿼리 ID@param parameterObject - 삭제 처리 SQL mapping 입력데이터(일반적으로 key 조건)를 세팅한 파라메터 객체(보통VO 또는 Map)@return DBMS가 지원하는 경우 delete 적용 결과 countparameterObject가 없는경우의 오버로딩함수도 제공 |
| public Object queryForObject(String queryId, [Object parameterObject]) | 단건조회 처리 SQL mapping을 실행한다.@param queryId - 단건 조회 처리 SQL mapping 쿼리 ID@param parameterObject - 단건 조회 처리 SQL mapping입력 데이터(key)를 세팅한 파라메터 객체(보통 VO 또는Map)@return 결과 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 단일 결과 객체(보통 VO 또는 Map)parameterObject가 없는경우의 오버로딩함수도 제공 |
| public Map queryForMap(String queryId, [Object parameterObject]) | 단건조회 처리 SQL mapping을 실행한다.@param queryId - 단건 조회 처리 SQL mapping 쿼리 ID@param parameterObject - 단건 조회 처리 SQL mapping입력 데이터(key)를 세팅한 파라메터 객체(보통 VO 또는Map)@return 결과 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 단일 결과 객체(보통 VO 또는 Map)parameterObject가 없는경우의 오버로딩함수도 제공 |
| public DataSet queryForDataSet(String queryId, [Object parameterObject]) | 리스트 조회 처리 SQL mapping 을 실행한다.@param queryId - 리스트 조회 처리 SQL mapping 쿼리 ID@param parameterObject - 리스트 조회 처리 SQL mapping 입력 데이터(조회 조건)를 세팅한 파라메터 객체(보통 VO 또는 Map)@return 결과 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 결과 객체(보통 VO 또는 Map)의 List를 Wrapped한 DataSet클래스 parameterObject가 없는경우의 오버로딩함수도 제공 |
| public ListDTO queryForPageList(String queryId, Object parameterObject, int pageno, [int listscale], [int pagescale]) | 부분 범위 리스트 조회 처리 SQL mapping 을 실행한다.@param queryId - 리스트 조회 처리 SQL mapping 쿼리 ID@param parameterObject - 리스트 조회 처리 SQL mapping 입력 데이터(조회 조건)를 세팅한 파라메터 객체(보통 VO 또는 Map)@param pageno - 현재 페이지 번호@param listscale - 한 페이지 조회 수@param pagescale – 페이지네비게이션에 노출될 수@return 부분 범위 결과 List 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 부분 범위 결과 객체(보통 VO 또는 Map) List를 Wrapped한 DataSet클래스를 또다시 Wrapped한 페이징정보를 포함하는 ListDTO클래스 |
| public List queryForList(String queryId, [Object parameterObject]) | 리스트 조회 처리 SQL mapping 을 실행한다.@param queryId - 리스트 조회 처리 SQL mapping 쿼리 ID@param parameterObject - 리스트 조회 처리 SQL mapping 입력 데이터(조회 조건)를 세팅한 파라메터 객체(보통 VO 또는 Map)@return 결과 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 결과 객체(보통 VO 또는 Map)의 ListparameterObject가 없는경우의 오버로딩함수도 제공 |
| public void startTransaction() | AOP의 자동트랜젝션처리를 이용하지 않고 수동으로 할경우 사용한다.트랙젝션이 시작 |
| public void commitTransaction() | AOP의 자동트랜젝션처리를 이용하지 않고 수동으로 할경우 사용한다.트랙젝션 커밋 |
| public void endTransaction() | AOP의 자동트랜젝션처리를 이용하지 않고 수동으로 할경우 사용한다.트랙젝션 롤백 / 종료 |
| public void startBatch()public int executeBatch() | AOP의 자동트랜젝션처리를 이용하지 않고 수동으로 할경우 쿼리를 모아 배치로 실행할 때 사용한다.Ex.startTransaction();startBatch();delete("queryId", parameterObject);executeBatch();commitTransaction();endTransaction(); |

## Service 클래스 개발가이드

Service 및 ServiceImpl 클래스는 비즈니스 로직을 처리하는 클래스이다.

ServiceImpl 클래스는 EgovAbstractServiceImpl 공통 클래스를 상속받는다.

Exception이 발생한 경우에는 EgovAbstractServiceImpl 클래스의 processException 메소드를 이용하여 처리하도록 한다.

다음은 ServiceImpl클래스의 구현 샘플이며, 이를 기본으로 확장하여 사용하도록 한다.

|  |
| --- |
| package upxframework.com.common.service.impl;import java.util.Map;import javax.annotation.Resource;import org.springframework.stereotype.Service;import upxframework.com.common.service.CommonService;import upxframework.com.core.data.DataSet;import upxframework.com.core.persist.ListDTO;import egovframework.rte.fdl.cmmn.EgovAbstractServiceImpl;@Service("common.commonService")public class CommonServiceImpl extends EgovAbstractServiceImpl implements CommonService { @Resource(name = "common.commonDAO") private CommonDAO commonDAO;  @Override public Map selectTest(String usrid) throws Exception { return commonDAO.selectTest(usrid); } @Override public DataSet selectTest(Map map) throws Exception { return commonDAO.selectTest(map); } @Override public ListDTO selectListTest(Map map) throws Exception {  return commonDAO.selectListTest(map); } } |

## Controller 클래스 개발가이드

Controller 클래스는 DispatcherServlet 요청을 받아서 처리하는 클래스이다.

Exception이 발생한 경우에 관련 에러 페이지로 포워딩 된다.

다음은 Controller 클래스의 구현 샘플이며, 이를 기본으로 확장하여 사용하도록 한다.

|  |
| --- |
| package upxframework.com.common.web;import java.util.HashMap;import java.util.Map;import javax.annotation.Resource;import javax.servlet.http.HttpServletRequest;import javax.servlet.http.HttpServletResponse;import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.RequestMapping;import org.springframework.web.servlet.ModelAndView;import upxframework.com.common.service.CommonService;import upxframework.com.core.data.DataSet;import upxframework.com.core.mvc.view.View;import upxframework.com.core.persist.ListDTO;import egovframework.rte.ptl.mvc.bind.annotation.CommandMap;@Controllerpublic class CommonBController { @Resource(name = "common.commonService") private CommonService commonService;  /\*\* \* 관리자 메인페이지 \* @return sample.jsp \* @throws Exception \*/ @RequestMapping("/back/common/Main.do") public String Main(@CommandMap Map<String, Object> commandMap, HttpServletRequest request, HttpServletResponse response) throws Exception{  return "/jsp/back/templateMainMaster"; }  /\*\* \* 관리자 샘플페이지 \* @return sample.jsp \* @throws Exception \*/ @RequestMapping("/back/common/sample.do") public String sample(@CommandMap Map<String, Object> commandMap, HttpServletRequest request, HttpServletResponse response) throws Exception{  commandMap.put("body", "/jsp/back/common/sample.jsp"); // 개발할 화면 return "/jsp/back/templateMaster"; // 바꾸지 마시오. }  /\*\* \* 테스트 \* @return test.jsp \* @throws Exception \*/ @RequestMapping("/back/common/test.do") public String test(HttpServletRequest request, HttpServletResponse response) throws Exception{  return "/jsp/back/common/test"; }  /\*\* \* 테스트2 \* @param request \* @param response \* @return \* @throws Exception \*/ @RequestMapping("/back/common/test2.do") public ModelAndView test2(@CommandMap Map<String, Object> commandMap, HttpServletRequest request, HttpServletResponse response) throws Exception{  HashMap map = new HashMap(); map.put("message", "호출성공하였습니다."); map.put("usrid", commandMap.get("p\_usrid")); map.put("pwd", commandMap.get("p\_pwd[]"));  return View.*jsonView*(map); }  /\*\* \* 테스트3 \* @param request \* @param response \* @return \* @throws Exception \*/ @RequestMapping("/back/common/test3.do") public ModelAndView test3(@CommandMap Map<String, Object> commandMap, HttpServletRequest request, HttpServletResponse response) throws Exception{  HashMap map = new HashMap(); String[] arrStr = (String[])commandMap.get("p\_str"); System.*out*.println("### arrStr[0] : " + arrStr[0]); System.*out*.println("### arrStr[1] : " + arrStr[1]);  map.put("message", "호출성공하였습니다."); map.put("str", commandMap.get("p\_str"));  Map mapData = commonService.selectTest((String)commandMap.get("p\_usrid")); DataSet data = commonService.selectTest(commandMap); data.next();  map.put("email", mapData.get("EMAIL")); map.put("name", data.getString("NAME"));  return View.*jsonView*(map); }  /\*\* \* jqGrid test 콜백 \* @param request \* @param commandMap \* @return \* @throws Exception \*/ @RequestMapping("/back/common/testList.do") public ModelAndView testList(@CommandMap Map<String, Object> commandMap, HttpServletRequest request, HttpServletResponse response) throws Exception {   ListDTO list = commonService.selectListTest(commandMap); int page = Integer.*parseInt*((String)commandMap.get("page"));  int totalCount = list.getTotalItemCount();  int total\_page = list.getTotalPageCount();  HashMap map = new HashMap(); map.put("rows", list.getItemList().getDataSet()); map.put("page", page); map.put("total", total\_page); map.put("records", totalCount);  return View.*jsonView*(map); }} |

## JSP 개발가이드

JSP는 ViewResolver를 통해서 전달된 Model 정보를 화면에 표시하는 로직을 담당한다.

JSP에서는 기본적으로 JSTL 태그를 사용하도록 한다.

JSP 페이지의 charset은 UTF-8을 사용한다.

JSP에서 만들어진 HTML은 반드시 W3C 호환성 표준을 따르도록 한다.(w3c validator이 용)

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %><%@ taglib uri="/tags/upx\_taglib" prefix="upxtag" %><script type="text/javascript"></script><form id="form1" name="form1" method="post" action=""><div class="list\_index\_box"> <h3 class="h3tit\_pot">메뉴</h3> <span class="list\_index">  총 <em class="color\_red" id="countCnt">0</em> 건 </span></div></form> |

## 표준 Sequence Diagram

위의 각 레이어별 처리 sequence는 다음과 같다.



# 시큐어 코딩가이드

## 진단항목 및 보안수준

### 진단 항목

웹 취약점 진단 및 소스코드진단은 OWASP(Open Web Application Security Project) TOP 10 기반의 취약점 및 안전행정부 기준 체크리스트를 포함하여 세부 진단 항목으로 나누어 수행한다.

|  |  |  |
| --- | --- | --- |
| **진단 항목** | **설명** | **비고** |
| SQL-Injection취약점 | ■ 비정상적인 요청에 대한 유효성 점검- DB 정보획득 및 저장된 데이터 열람 및 조작- DB의 시스템 명령을 이용하여 시스템 실행 |  |
| XSS / CSRF취약점 | ■ 게시판 등에서 글쓰기 기능을 통한 스크립트 실행 여부 점검■ 스크립트를 통한 데이터 수정, 삭제 가능 여부 점검 |  |
| 파일 업로드취약점 | ■ 악성파일 업로드 및 실행 여부 점검- 서버에서 실행 가능한 확장자(asp, jsp, php)를 가진 파일의 업로드 가능 여부 확인- 업로드 파일의 실행 가능 여부 확인 |  |
| 파일 다운로드취약점 | ■ 파일 다운로드 알고리즘 보안성 점검- 다운로드 스크립트의 파라미터를 변경하여 주요 파일 다운로드 시도 |  |
| 관리자 페이지노출(접근) 취약점 | ■ 관리자 페이지 점검- 외부 사용자가 관리자 페이지로 접근가능 여부 확인 |  |
| 디렉토리 인덱싱취약점 | ■ 디렉토리 나열 점검- 기본 페이지가 없는 URL 처리 여부 체크 |  |
| 에러처리 취약점 | ■ URL에 존재하지 않는 경로나 임의의 파일 요청 등 비정상적인 값에 대한 에러처리 점검- Error 코드에 웹 어플리케이션의 종류 및 버전 사용하는 DB정보, 시스템 정보 노출 체크 |  |
| 암호화 전송 미흡취약점 | ■ 패킷을 통해 전송되는 데이터의 암호화 여부 점검- 고객정보의 SSL 및 인증과 관련된 데이터의 암호화 체크 |  |
| 파라미터 변조취약점 | ■ 비정상적인 요청에 대한 유효성 점검- 파라미터 변조를 통한 데이터 전송 점검■ 권한 인증 우회 여부 점검- 파라미터 변조를 통한 인증 우회 여부 체크 |  |
| 정보누출취약점 | ■ html 소스 내에 개인정보, 인증정보, DB접속 정보 등의 주요 정보 노출 점검 |  |
| 계정관리취약점 | ■ 사용자의 이름이나 패스워드 등의 단순 조합의 값을 사용한 인증 여부 점검 |  |
| 쿠키변조취약점 | ■ 쿠키 변조를 통한 인증 우회 여부 점검 |  |
| 불필요한 파일취약점 | ■ 시스템/애플리케이션 설치 시 기본 설정 파일 점검■ 개발 시 테스트를 하거나 백업 및 툴에 의해 생성되는 파일 점검 |  |
| 악성 콘텐츠취약점 | ■ 악성 콘텐츠 및 악의적인 프로그램 다운로드 점검■ 악의적인 사이트로의 이동 발생 여부 점검 |  |
| 불충분한 인증취약점 | ■ 개인정보 수정 및 패스워드 수정 기능의 페이지에 접근 전에 본인인증에 대한 재인증 여부 점검 |  |
| 취약한 패스워드 복구취약점 | ■ 재발행 되는 패스워드의 일정패턴 점검 |  |
| 불충분한 세션만료취약점 | ■세션 만료기간에 대한 적절한 설정 점검 |  |

[표 ] 웹 취약점 진단 진단 항목

|  |  |
| --- | --- |
| **진단 항목** | **비고** |
| Cross-Site Scripting : Persistent, Reflected |  |
|  LDAP Injection |  |
| Open Redirect |  |
| Path Manipulation |  |
| Privacy Violation |  |
| SQL Injection |  |
| SQL Injection : iBatis Data Map |  |
| XPath Injection |  |
| Access Control : Database |  |
|  Header Manipulation |  |
|  Insecure Randomness |  |
|  Null Dereference |  |
|  Password Management : Password in Configuration File |  |
|  Unreleased Resource : Database, File, Sockets, Streams |  |
|  Weak Encryption |  |
|  Cookie Security : Persistent Cookie |  |
|  J2EE Misconfiguration : Missing Error Handling |  |
|  Password Management : Password in Comment |  |
|  Poor Error Handling : Empty Catch Block / Overly Broad Catch |  |
|  Poor Logging Practice : Use of a System Output Stream |  |
|  Redundant Null Check |  |
|  Redundant Null Check |  |
|  Weak Cryptographic Hash |  |

[표 ] 소스코드 진단항목

### 위험도 등급

각 진단 대상에 대한 보안 수준을 산출하기 위해 위험도를 세 등급으로 구분한다.

|  |  |  |
| --- | --- | --- |
| **평가등급** | **평가기준** | **비고** |
| 높음(High) | ■ 해당 취약점으로 인해 권한 상승 및 직접적인 관리자 권한 획득이 가능한 취약점■ 해당 취약점으로 인해 데이터 변조 및 서비스 가용성으로 큰 영향을 줄 수 있는 취약점 |  |
| 보통(Medium) | ■ 인가된 사용자에 의한 허가되지 않은 작업을 허용하는 취약점■ 비인가자의 정보 수집으로 인해 시스템에 접근하거나 피해를 줄 수 있는 취약점 |  |
| 낮음(Low) | ■ 해당 취약점으로 인해 시스템에 영향을 주지는 않으나 시스템에 대한 일부 정보를 수집할 수 있는 취약점 |  |

[표 ] 보안 수준

### 수행 방법



 [그림 ] 수행 방법