행정안전부

전자정부 표준 공통서비스 및 개발프레임워크 2단계 구축

개발표준가이드

컨소시엄 삼성SDS SAMSUNG LG CNS SK C&C

2009.07.15

본 개발 표준은 본 사업의 개발부문(오픈소스 활용부분은 제외)에 적용할 표준을 정의하는 것으로 개발자 상호간에 개발 표준을 공유하여 소스 코드의 일관성 확보를 목적으로 한다.

1. 공통 사항

1.1 Coding 원칙

○ 주석은 모든 코드에 상세히 기술하는 것을 원칙으로 한다.

○ 소스 코드는 불가피한 내용을 제외하고 원칙적으로 중복을 금지한다.

○ 소스 코드에서 사용하는 모든 단어는 용어사전의 내용을 기준으로 한다.

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

1.2 Coding Style

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

1.2.1 indent

○ indent는 tab을 사용하지 않고, space만 사용한다.

○ indent의 크기는 4로 한다.

1.2.2 space

○ 한 줄에는 하나의 statement만 기술한다.

○ semicolon, comma, 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.

1.2.3 brace

○ '{' 와 '}' 는 새로운 라인에 기술하며, 다른 내용과 함께 기술하지 않는다. (주석 제외)

○ '{' 는 기존의 '{' 와 비교해서 indent(4-space)를 준다.

○ '}' 는 짝이 되는 '{' 과 동일하게 indent(4-space)를 준다.

○ brace에 주석을 기입하는 경우 '//' 주석을 사용한다.

※ 위의 형식이 적용된 기 배포된 표준화된 소스 포매터(Code Formatter)를 Eclipse에 임포트하여 활용(ctrl+shift+F)하도록 한다.

1.2.4 용어

○ 용어는 원칙적으로 용어사전을 준수하여 용어명에 맞는 영문을 사용한다.

○ 하나의 단어를 사용하는 경우에는 용어영문명을 사용한다.

○ 여러개의 단어를 조합하여 사용하는 경우에는 영어약어명을 조합해서 사용한다.

○ 용어를 사용할 때 동음이의어에 주의한다.

○ 사용될 시스템 코드는 다음과 같이 영문 3자로 정의한다.

코드	코드 설명
MGT	관리환경
DEV	개발환경
COM	공통컴포넌트

1.2.5 Logging

○ Log는 반드시 Framework에서 제공한 Logger만을 사용한다.

○ Log는 debug, info, warn, error로 구별하여 사용한다.

○ Log는 반드시 발생 시간과 위치 그리고 내용을 포함한다.

○ Log는 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 error는 아니나 잠재적인 error의 발생이 가능한 내용을 기록한다.

○ error log는 error code와 함께 error에 대한 내용을 기록한다.

1.2.6 file명명 및 Line

○ file이름은 50자를 넘지 않도록 한다.

○ file명은 용어사전의 내용을 기준으로 작성하며, 한글의 영문표기나 영문 이외의 특수기호 및 숫자는 사용하지 않는다.

○ File의 크기는 500 Line을 넘기지 않는다. (주석 제외)

○ Function(Method)의 크기는 50 Line을 넘기지 않는다. (주석 제외)

○ Statement의 크기는 80 Character를 넘기지 않는다.

1.3 웹 시스템 보안 적용

○ 이 부분은 보안코딩 가이드를 참조하여 준용하도록 한다.

1.4 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.

1.4.1 Client Message

○ java script는 alert 기능을 이용한다.

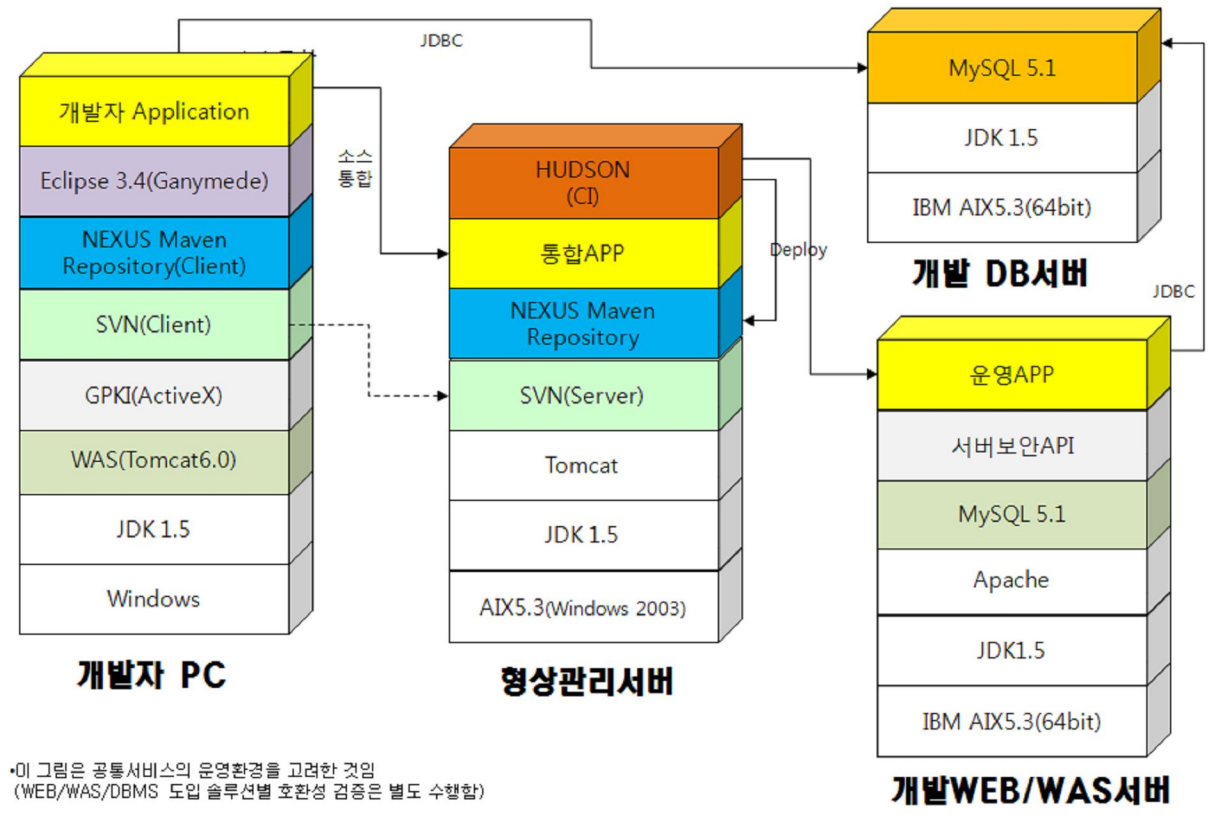
1.4.2 Server Message

○ 프레임워크에서 제공하는 서버 메시지 규약(XML파일로 정의)을 사용한다.

1.4 개발환경 구성도

○ 개발환경은 기본적으로 개발자 PC, 개발서버(형상관리, WEB/WAS, DB)로 구성되며 관련 환경은 통합 빌드 및 배포관리가 될 수 있도록 구성한다.

개발환경 구성도



•이 그림은 공통서비스의 운영환경을 고려한 것임
(WEB/WAS/DBMS 도입 솔루션별 호환성 검증은 별도 수행함)

2. 사용자 화면

2.1 공통

○ 절대 path 사용을 원칙으로 한다.

○ *[web-root]* 이하로 확장자 별로 directory를 구성하고, 하위로 system명과 sub-system을 파일의 기본 directory로 한다.

○ sub-system 이하로 directory는 3-level 이하로 구성한다.

○ 하나의 directory 내의 파일 개수는 200개 이하로 한다.

○ 파일은 최하위 directory에만 위치한다.

○ system명과 sub-system명은 용어 영문명이나 영문약어명의 첫 글자를 조합하여 가능한 3글자로 작성하되 6글자 이하로 작성한다.

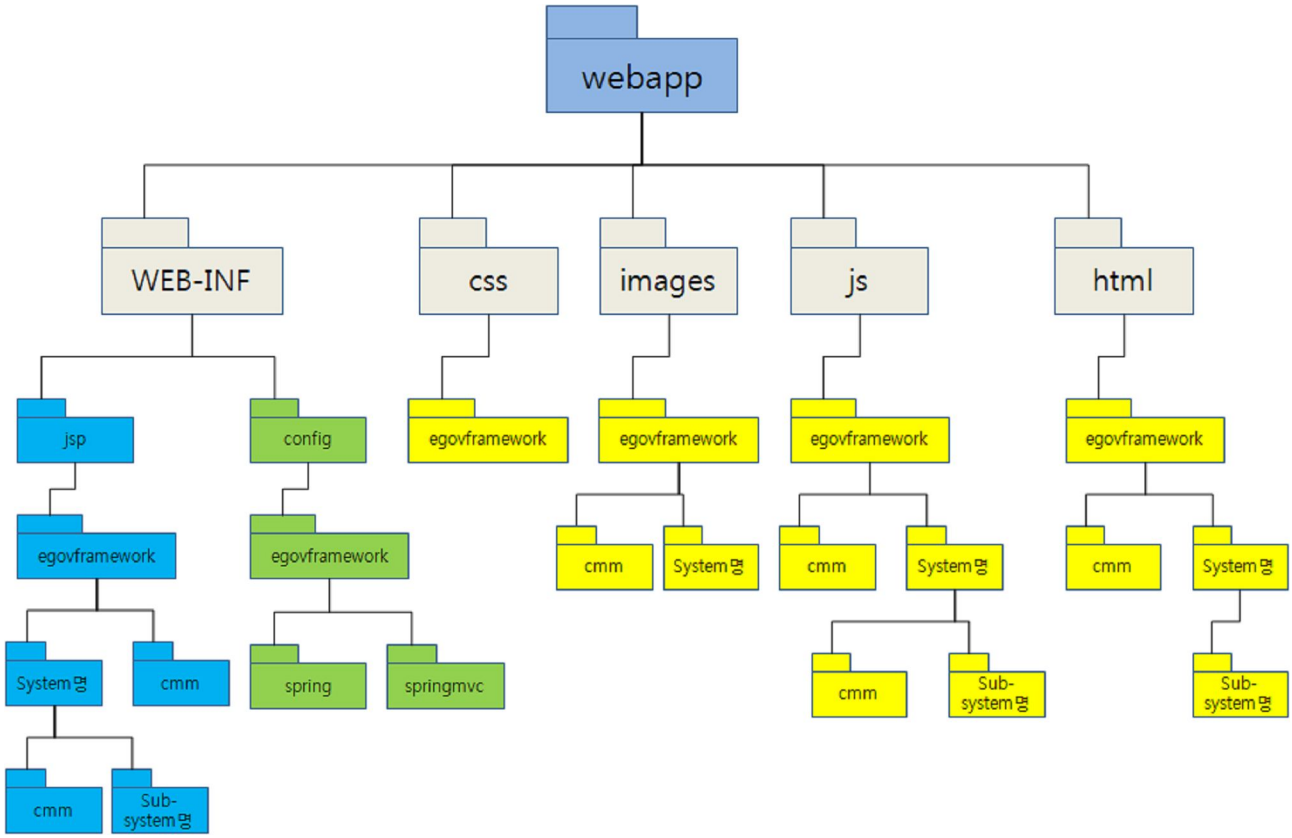
○ directory에서 사용하는 system명과 sub-system명은 소문자만 사용한다.

○ File Encoding은 UTF-8을 기본으로 한다. 단, File의 크기가 특정 크기 이상으로 되는 경우 jsp가 정상적으로 동작하지 않는 등의 경우에 대해서는 예외적으로 euc-kr을 사용한다.

○ 기본 해상도는 1024 X 768로 한다.

2.2 웹 디렉토리 구조

○ 기본 디렉토리 구조는 다음과 같다.



2.3 html

2.3.1 명명규칙

○ html이 위치하는 논리적 위치는 *[web-root]*/html/egovframework/*[system]*/*[sub-system]*을 기본으로 한다.

○ html 파일의 명명규칙은 Egov*[용어사전]*.html로 한다.

○ 용어사전의 단어를 사용하는 경우 첫 글자만 대문자로 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 p_*[pop-up명]*을 사용한다.

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

2.3.2 Coding Style

○ 스타일은 css를 사용한다.

○ css이 위치하는 논리적 위치는 *[web-root]*/css/egovframework을 기본으로 한다.

○ css 파일의 명명규칙은 Egov*[용어사전]*.css로 한다.

○ size는 상대값(%)을 사용하지 않고, 절대값을 사용한다.

2.4 jsp

2.4.1 명명규칙

○ jsp가 위치하는 논리적 위치는
[web-root]/WEB-INF/jsp/egovframework/*[system]*/*[sub-system]*을 기본으로 한다.

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

○ jsp 파일의 명명규칙은 Egov*[용어사전]*.jsp로 한다.

○ 용어사전의 단어를 사용하는 경우 첫 글자만 대문자로 한다.

○ 용어사전을 사용할 경우에는 Postfix를 제외하고는 가능한 full name을 활용하도록 한다.

○ 용어사전의 Postfix 명명규칙은 다음과 같다.

Postfix명	설명	적용 예	비고
List	목록	EgovChangeRequestList	
Regist	등록	EgovChangeRequestRegist	
Detail	상세	EgovChangeRequestDetail	
Updt	수정	EgovChangeRequestUpdt	
Popup	팝업	EgovChangeRequestPopup	
Search	조회	EgovChangeRequestSearch	

○ jsp 내의 변수명은 소문자만 사용하며 단어 조합 시에는 '_'를 사용하여 구분한다.

○ jsp의 각 컨트롤의 변수명에 대한 명명규칙은 다음과 같다.

컨트롤	Prefix명	적용 예	비고
버튼	btn	회원등록버튼 --> btnRegiUsr	
이미지	img	검색버튼 --> imgSearch	
Text	txt	사용자이름 Input --> txtName	
TextArea	txa	내용 Textarea --> txaContent	
Select	sel	Select에서 옵션선택 --> selOption1	
ListBox	lst	List에서 메뉴선택 --> lstMenu01	
Radio	rdo	Radio에서 타입선택 --> rdoTypeA	
CheckBox	chk	체크박스로 옵션선택 --> chkOption1	

2.4.2 Coding Style

○ jsp 내에서 jsp include는 원칙적으로 사용하지 않는다.

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

○ 다른 jsp에서 include하는 jsp는 공통 jsp(cmm경로에 있는 JSP)로 한정한다.

○ 기본 스크립트릿 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.(*[web-root]*/js/egovframework/*[system]*/cmm 제외)

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

2.4.3 주석

○ JSP 파일 주석

```
<%--
  Class Name : EgovLogin.jsp
  Description : 사용자 로그인을 담당
  Modification Information
```

```
      수정일          수정자                  수정내용
    -------     --------     ---------------------------
     2008.01.01     홍길동          최초 생성
     2008.02.01     길동이          그룹웨어 결재 연동기능 추가

    author   : 각팀명(예. 공통서비스 개발팀) 홍길동
    since    : 2009.02.01
--%>
```

2.4.4 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)

○ 버튼은 오른쪽 정렬한다.

2.5 java script

2.5.1 명명규칙

○ js가 위치하는 논리적 위치는 *[web-root]*/js/egovframework/*[system]*/*[sub-system]*을 기본으로 한다.

○ js 파일의 명명규칙은 *[용어사전]*.js로 한다.

○ 용어사전의 단어를 사용하는 경우 첫 글자만 대문자로 한다.
(ex. EgovCheckValid.js)

○ function 명은 fn_*egov[동사]_[명사]*를 사용한다.
(ex. fn_egov_check_valid(), 동사 및 명사는 용어사전을 기준으로 한다.)

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

2.5.2 Coding Style

○ js 내에서 다른 include는 원칙적으로 사용하지 않는다.

○ 다른 js에서 공통으로 include하는 js는 *[web-root]*/js/egovframework/cmm에 위치한다.

○ 여러 sub-system에서 include하는 js는 *[web-root]*/js/egovframework/*[system]*/cmm에 위치한다.

2.5.3 주석

○ file 주석

```
/*******************************************************
  파일명 : EgovCommon.js
  설  명 : 전자정부 공통 JavaScript
  수정일        수정자         Version        Function 명
 -------    --------    ----------   --------------
 2009.01.01   홍길동          1.0            최초 생성
 2009.02.01   길동이          2.0            fn_check_period
*******************************************************/
```

○ function 주석

```
/**********************************************************************/
  함수명 : fn_egov_check_period
  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 10자리(2004-05-01)로 구성되어야 한다.
           상황에 따라 구분자(sep)는 '-'(2004-05-01) or '.'(2004.05.01)
  작성일 : 2009-02-01
  작성자 : 각팀명(예. 공통서비스 개발팀) 홍길동
   수정일        수정자              수정내용
  ------      ------     -------------------
   2009.02.01    길동이         윤년 처리 추가
/**********************************************************************/
```

3. 자바 클래스

3.1 공통

java class file은 필요 시 전자정부 사업을 위한 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.

3.2 명명규칙

3.2.1 java package

○ 기본 패키지 명명규칙은 다음과 같다.

package 명		설 명
egovframework		전자정부 표준 공통컴포넌트 및 개발프레임워크 총괄 도메인명
시스템명		시스템 코드 3자리(예: 관리환경-mgt, 개발환경-dev)
서브시스템명		서브시스템 코드 3자리
용도별 클래스	common	각 서브시스템별로 공통으로 사용하는 클래스를 가지고 있는 패키지 예. egovframework.mgt.com.common egovframework.mgt.com.web.common
	aspect	AOP 관련 aspect 클래스를 가지고 있는 패키지 예. egovframework.mgt.com.aop
	exception	exception : exception 클래스를 가지고 있는 패키지. 실행환경의 에러처리 패키지 서비스를 제공하는 패키지를 이용한다.(egovframework.mgt.com.exception)
	service	service클래스, vo, model 클래스를 가지고 있는 패키지 예. egovframework.com.sec.userinfo.service 모델이 많거나 공통으로 관리하고자 하는 경우에 별도의 위치에 통합하여 분리할 수도 있음 예. egovframework.com.sec.vo
	service.impl	서비스 구현 클래스와 DAO 클래스를 가지고 있는 패키지 예. egovframework.com.sec.userinfo.service.impl
	web	controller 클래스를 가지고 있는 패키지 예.egovframework.com.sec.userinfo.web

○ test 및 eclipse plugin 관련 프로그램은 위의 일반자바클래스와 하위 패키지 구조는 동일하다.

(예. egovframework.rte.fdl.aop)

- 테스트 클래스는 클래스명명규칙으로 유일성을 식별하도록 한다.(~Test.java)

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

(예. egovframework.rtl.f0l.exception.handler)

3.2.2 java 라이브러리

○ library 명명규칙은 Maven의 규칙에 따라 부여하며,
[*GroupID.ArtifactID*]-[*version*].확장자를 사용한다.

ex) egovframework.com.security-1.0.jar

○ 개발이 수행될 때는 SNOPSHOT버전을 이용하고, 향후 통합테스트가 완료되어 정식 버전이 배포가 될 경우에는 1.0부터 시작하도록 한다.

3.2.3 java class

○ java file의 명명규칙은 Egov*[용어사전][postfix]*.java로 한다.

○ 용어사전의 단어를 사용하는 경우 첫 글자만 대문자로 한다.
(ex. EgovLoginCheckService.java)

○ postfix는 framework에서 정의하는 class 요소에 대한 구분으로 해당 java class의 성격을 나타낸다.

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

* Test는 테스트 클래스 및 관련 xml파일에 붙는 postfix임

○ 예외사항

- 다음과 같은 유형의 클래스에는 Egov를 Prefix로 붙이지 않는다.

1) DAO 클래스

2) Model 클래스

3) VO 클래스

4) 향후 산출물로 배포되지 않고 임시로 사용하는 테스트 클래스

5) 기타 외부(프레임워크를 사용하는 기관 또는 프로젝트)에서 공통서비스 혹은 프레임워크 서비스를 직접적으로 소스코드에 명시하여 사용하지 않는다고 판단되는 경우

3.2.4 interface class

○ Interface는 java 클래스명과 동일한 방법으로 부여한다.

3.2.4 java method

○ java method의 명명규칙은 *[동사][용어사전]*로 한다.

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

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

○ method에서 사용하는 동사는 다음 표와 같이 명명한다.

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

3.2.5 변수 및 상수

○ java 변수의 명명규칙은 *[용어사전]*을 조합하여 30자 이내로 명명 한다.

○ 첫 글자는 소문자를 사용하며 이후 용어의 첫 글자만 대문자를 사용하며, 숫자 및 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 용어사전을 사용하여 대문자로만 작성하며 단어 사이는 '_'를 사용하여 구분한다. (단, SUCEESS=1, FAIL=0으로 정의한다.)

3.2.6 test code (JUnit)

○ 기본적으로 개발환경에서 제공하는 JUnit 테스트 기능을 활용한다.

3.2.7 *.do 명명규칙

○ Controller의 매핑정보를 관리하는 *.do의 명명규칙은 다음과 같다.

- "/서브시스템/.../동사명+용어사전.do"(단, 동사명은 add, select, get, remove 등의 의미있는 단어 사용)

예. `/cop/bbs/adm/addBBSMaster.do`

3.3 Coding Style

3.3.1 공통

○ 코드는 한줄에 하나의 문장만 기술한다.

○ java class 내에서 Static 변수는 항상 static final로 선언한다.

○ public 변수는 사용하지 않는다.

○ 변수 선언 시 class type 별로 모은다.

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

○ 배열 선언은 *[class]*[] *[variable-name]*로 한다.
(ex. EgovString[] sampleArray = {"1","2"};)

○ 변수 선언 순서
- constant 변수
- private 변수

○ protected method는 사용하지 않는다.

○ method 선언 순서

- constructor (default기술 후 argument가 적은 순으로)
- main method
- private method
- public method

○ return 값에는 연산을 수행하지 않는다.

○ code상에서 의도적으로 Exception을 발생시키지 않는다.

○ code상에서 Exception을 사용해서 비즈니스 로직을 처리하지 않는다.

3.3.2 String

○ String의 연결 시에는 StringBuffer를 사용한다.

3.3.3 블록문

○ 내용이 없는 블록문은 사용하지 않는다. (꼭 사용해야 하는 경우 Log라도 입력한다.)

3.3.4 반복문

○ 반복문 내는 가급적이면 신규 변수를 생성하지 않는다.

○ 반복문 조건절에는 연산이 들어가지 않도록 한다.

3.3.5 조건문

○ 조건문은 3중을 초과해서 사용하지 않는다.

○ 조건절에 not(!)을 사용하지 않는다. (가독성 저하 우려)

3.4 주석

3.4.1 자바 클래스 주석

○ 주석은 아래의 표준을 준수하여 작성하여 자바문서화 주석(Javadoc)으로 생성 및 사용한다.

○ 클래스(인터페이스)명, 클래스설명, 수정이력(수정이 발생할 경우에만 해당), 작성자, 최초작성일, 버전, 참조클래스, 권리(Copyright)를 명시한다.

○ template

-배포되는 Code Template을 Eclipse에서 활용(alt+shift+J)하도록 한다.

```
/**
```

```
 * 사용자 계정을 처리하는 비즈니스 구현 클래스
 * <p><b>NOTE:</b> Exception 종류를 EgovBizException, RuntimeException 에서만 동작한다.
 * fail.common.msg 메세지키가 Message Resource 에 정의 되어 있어야 한다.
 * @author 공통컴포넌트 개발팀 홍길동
 * @since 2009.06.01
 * @version 1.0
 * @see
 *
 * <pre>
 *  == 개정이력(Modification Information) ==
 *
 *   수정일       수정자            수정내용
 *  -------    --------    ---------------------------
 *   2009.03.20  홍길동           최초 생성
 *
 * </pre>
 */
```

3.4.2 method 주석

○ Interface Class는 method 주석을 기재하지 않는다.

○ template

-배포되는 Code Template을 Eclipse에서 활용(alt+shift+J)하도록 한다.

```
/**
 * userInfo 사용자ID에 해당하는 사용자명을 조회
 * @param userID  검사 항목에 대한 구분자
 * @param role  사용자 권한정보 구분
 * @return 사용자명
 * @exception MyException
 * @see cmm.ROLE
 * @see SA_CHK_ITEMS
*/
public String userInfo(String usrID, String role)
```

3.4.3 변수 주석

○ 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
```

3.5 각종 설정(config) 파일 명명규칙

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

3.5.1 파일 종류별 구분

가. Spring 공통 설정파일

○ Spring 관련 공통 설정 파일의 배포위치는 WEB-INF/config/egovframework밑에 위치하며 다음과 같이 명명규칙을 준용하여 사용하도록 한다.

1) spring 패키지(WEB-INF/config/egovframework/spring)

구분	설명	명명규칙	사용샘플
AOP관련 설정	AOP관련 Aspect 공통 설정 정보를 관리한다.	context-aspect.xml	예. context-aspect.xml
common 설정	메시지 관련 빈설정 등의 공통 설정 정보를 관리한다.	context-common.xml	예. context-common.xml
datasource 설정	데이터 소스 설정정보를 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
빈설정정보	서비스의 빈관련 설정정보를 관리한다. *주의 : 본 사업은 Annotation 설정을 표준으로 하기 때문에 필요한 경우에만 사용하도록 한다.	system코드-용어명-beans.xml	예. com-hello-beans.xml

2) springmvc 패키지(WEB-INF/config/egovframework/springmvc)

구분	설명	명명규칙	사용샘플
Dispatcher Servlet 설정	dispatcher servlet 관련 설정정보를 관리한다.(SimpleMappingExceptionResolver, ResourceBundleMessageSource 등의 관련 클래스 정보)	dispatcher-servlet.xml	예 . dispatcher-servlet.xml

나. 메시지(message) 설정파일

○ 메시지 파일은 시스템 혹은 사용자 웹화면에 표시할 메시지 코드와 정보를 담고 있는 파일이다.(관련 파일 위치는 context-common.xml파일에서 정의)

○ 메시지 파일은 공통 메시지 파일과 업무별 파일로 나눌 수 있다.

○ 메시지 파일은 국제화 서비스를 고려하여 영문과 한글로 2개를 모두 작성한다.

○ 메시지 파일의 위치는 "리소스 HOME/message" 경로에 위치한다.

구분	설명	명명규칙	사용샘플
공통메시지파일	업무에서 공통으로 사용하는 메시지 정보를 관리한다.	Egov-message-common_en_US.properties, Egov-message-common_ko_KR.properties	좌동
업무별메시지파일	각 업무에서 고유하게 사용하는 메시지 정보를 관리한다. * system 코드로 통합 관리하기가 어려울 경우에는 sub_system별로 나누어 관리해도 된다. (예.Egov-message-subsysem코드_en_US.properties)	Egov-message-system코드_en_US.properties, Egov-message-system코드_ko_KR.properties	Egov-message-com_en_US.properties, Egov-message-com_ko_KR.properties

○ 메시지 파일의 내용은 다음과 같은 규칙으로 부여한다.

메시지 코드	메시지 내용	사용 예
fail.메시지특성명.msg	심각한 에러가 발생한 경우	fail.sql.msg=sql 에러가 발생했습니다! error code: {0}, error msg: {1}
warn.메시지특성명.msg	사용자에게 경고하고자 하는 경우	warn.insert.msg=허용되지 않은 자료에 대한 입력을 시도했습니다.
info.메시지특성명.msg	사용자에게 특정 정보를 알리는 경우	info.nodata.msg=해당 데이터가 없습니다.
message.메시지특성명.msg	사용자에게 특정한 메시지를 전달하고자 하는 경우	message.exception.msg=나중에 다시 시도하십시오.

다. SQL Map 설정파일

○ SQL Map은 iBATIS 프레임워크에서 사용하는 SQL문을 담고 있는 것이다.

○ SQL Map 파일의 위치는 “리소스 HOME/sqlmap" 경로에 위치한다.

구분	설명	명명규칙	위치	사용샘플
공통파일	sqlmap파일의 위치를 담고 있는 메타정보 관리 파일이다.	Egov-sql-map-config.xml	리소스 HOME/sqlmap	좌동
업무별SQL MAP파일	각 업무에서 SQL정보를 담고 있는 파일이다.	Egov-*subsystem명*_mapping.xml (단, subsystem이 커서 sql파일을 분리하고자 하는 경우에는 subsystem명 뒤에	리소스HOME/sqlmap/system패키지명	Egov-fdl.mapping.xml, Egov-sec01.mapping.xml

		2자리 일련번호(01~)를 붙일 수 있다.)		

3.6 예외처리

3.6.1 예외 발생 시 처리해야 하는 내용

○ 해당 Exception에 대한 Error Log 기록(파일 혹은 console)한다.

○ CUD transaction이 수행중인 경우에는 트랜잭션 서비스와 연계되어 rollback을 수행한다.

○ 해당 Exception에 대해서 사용자에게 message를 제공한다.

○ Unchecked Exception이 발생하는 경우는 Exception내용을 관리자에게 메일을 발송하는 기능을 제공할 수 있다.

○ Checked Exception이 발생하는 경우에는 사용자가 인지할 수 있는 메시지를 페이지에 출력하고, 시스템관리자에게 메일을 발송할 수 있다.

○ Checked 혹은 Unchecked Exception 중 처리되지 않는 Exception은 서블릿 컨테이너에서 제공하는 에러처리 기능을 활용한다.(500error.jsp 페이지에 출력)

○ 특정 URL에 접근할 수 없는 경우에는 404에러가 발생한다. 이 경우에는 404error.jsp에 에러내용을 출력하도록 한다.

3.6.2 레이어별 Exception처리

가. DAO단에서 Exception 처리

○ DAO단에서는 스프링에서 제공하는 SqlMapClientDaoSupport를 상속받아 사용하면 SQLException 대신 스프링이 제공하는 예외 클래스(Runtime Exception인 DataAccessException)로 변환해 주기 때문에 별도의 SQL관련 Exception로직을 사용하지 않는다.

나. Service단에서 Exception 처리

○ Service단에서 DAO에서 Exception을 받아서 Exception을 처리할 수 있다. 프레임워크에서 추가로 제공하는 Exception은 다음과 같다.

구분	설명	사용규칙
EgovBizException	업무에서 Checked Exception인 경우에 공통으로 사용하는 Exception이다. 개발자가 특정한 오류에 대해서 throw하여 특정 메시지를 전달하고자 하는 경우에는 processException() 메소드를 이용하도록 한다.	Exception을 강제로 throw하고자 하는 경우에 AbstractServiceImpl:processException("메시지코드")을 이용한다.
ExceptionTransfer	AOP 기능을 이용하여 Unchecked Exception인 경우 ServiceImpl 클래스에서 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 대해서 Handler에 의해서 동작한다.	context-aspect.xml에 AOP로 설정되어 있다. DataAccessException발생 시 메시지 코드는 "fail.sql.msg" 메시지 코드로 매핑되어 있다.

○ Exception 발생 시 메시지 리소스와 연계되어 메시지 코드와 메시지 내용이 출력되도록 한다.

다. Controller단에서 Exception 처리

○ Controller단에서 처리하는 Exception은 스프링에서 제공하는 SimpleMappingExceptionResolver(org.springframework.web.servlet.handler.SimpleMappingExceptionResolver)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.	DataAccessFailure.jsp
트랜잭션 처리 오류 (org.springframework.transaction.TransactionException)	EgovBizException이 발생한 경우에 표시되는 화면이다.	EgovBizException.jsp

HTTP 세션오류	HTTP 세션이 유효하지 않을 경우 표시한다.	EgovHttpSessionException.jsp

3.7 로깅처리

3.7.1 로깅처리 전략

○ 로깅은 기본적으로 log4j 라이브러리를 이용한다.

○ Unchecked Exception이 발생할 경우에는 Stack Trace내용을 로그에 남기도록 한다.

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

○ Logging 시에는 성능을 위하여 현재 Logging을 위한 Logging 레벨을 체크하는 구문을 반드시 추가한다. 예를들어, DEBUG 레벨로 메시지를 Logging하고자 한다면 다음과 같이 작성하도록 한다.

```
import org.apache.commons.logging.Log;
    import org.apache.commons.logging.LogFactory;
    protected Log log = LogFactory.getLog(this.getClass());
    if ( log.isDebugEnabled()) {
        log.debug("Logging... " + this);
   }
```

3.8 트랜잭션처리

3.8.1 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="insert*" propagation="REQUIRED"/>
                      <tx:method name="select*" read-only="true"/>
                     <tx:method name="*" rollback-for="Exception"/>
              </tx:attributes>
       </tx:advice>

      // 트랜잭션 Pointcut를 설정한다.
       <aop:config>
              <aop:pointcut id="requiredTx"
                     expression="execution(* egovframework.com..impl.*Impl.*(..))"/>
              <aop:advisor advice-ref="txAdvice"
                     pointcut-ref="requiredTx" />
       </aop:config>

</beans>
```

3.9 DataSource 설정

○ DBMS와 연동하기 위해서는 DataSource를 설정해야 한다.

○ DataSource 설정은 WAS의 기능을 이용하는 방법과 Spring의 기능을 이용하는 방법이 있는데, 본 프로젝트에서는 WAS의 기능에 대한 의존성을 줄이기 위해서 Spring의

DataSource 기능을 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.

○ Connection Pool을 이용하기 위해서는 다음과 DBCP(Jakarta Commons Database Connection Pool) API를 사용한다. 이는 org.apache.commons.dbcp.BasicDataSource 클래스를 이용한다. 관련 설정은 context-datasource.xml에 정의되어 있다.

○ 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.

4. SQL 작성규칙

4.1 공통

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

4.2 명명규칙

4.2.1 SQL File

○ 3.5.1의 명명규칙을 참조하도록 한다.

4.2.2 SQL Id

○ "*DAO클래스명.method*"의 형태를 사용한다. (DAO의 method는 하나의 SQL만을 수행한다.)
예. CategoryDAO.selectList

4.3 Coding Style

4.2.1 SQL

○ SQL 예약어는 대문자를 사용한다.

○ SELECT, UPDATE시 컬럼 나열은 1줄당 1개씩 소문자로 작성을 기본으로 하며, 컬럼이 많은 경우에 한하여서 1줄에 3개 컬럼까지 쓰는 것을 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, 1줄당 2개의 테이블까지 적는 것을 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.

○ 괄호를 사용할 경우, 즉 인라인 뷰, 스칼라 서브쿼리 등을 사용할 경우 괄호가 존재하는 라인은 라인을 변경하지 않는다.

```
ex
    SELECT f.unpay_pass_cnt, h.dept_nm
        FROM   tbblc_unpaymngdetail a,
               (SELECT a.pay_man_no, b.cust_nm, b.corp_nm,
                       b.cust_base_addr
                   FROM   tbmka_payer a, tbmka_cust b
                   WHERE a.pay_man_no LIKE '%' ) b,
               tbsgc_comcd d,
               tbblc_unpay f,
          ....

    SELECT /* IM_cm_zip_srch_pl */ *
        FROM   tbbla_billcdrtelenat
        WHERE svc_Detail_no IN ( SELECT svc_detail_no
                                 FROM    tbslo_isvcdm
                                 WHERE   subsvc_cd = 'S011' )
```

4.4 주석

4.4.1 SQL File 주석

○ template

```
<!--
    SQL File Name : EgovUserLogin_SQL.xml
    Description : 사용자 로그인을 담당
    Modification Information
     수정일        수정자         Version         Query Id
    -------     --------     ----------    --------------
    2008.01.01    홍길동          1.0             최초 생성
    2008.02.01    길동이          2.0             EgovCmmnDAO.sampleQuery 추가
```

```
-->
```

4.4.2 SQL 문 주석

○ template

```
<!-- 쿼리명 : 민원 코드 조회
     설명 : 민원코드를 조회하기 위한 쿼리
      수정일        수정자              수정내용
     ------       ------          --------------------
      2008.01.01    홍길동             최초 생성
      2008.02.01    길동이             민원명 조건 삭제
    작성자 : 홍길동
    최초작성일 : 2008.01.01
-->
```

4.4.3 기타 주석

○ SQL 내의 주석은 '--'을 사용한다.

○ xml 내의 주석은 '<!-- -->'을 사용한다.

5. 레이어별 개발가이드

5.1 공통사항

5.1.1 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 이용하는 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.	// sqlMap 객체에 com.ibatis.sqlmap.client.SqlMapClient를 전달하고자 할 경우 @Resource(name = "sqlMapClient") SqlMapClient sqlMap;
@Component	일반적인 클래스를 빈으로 자동으로 등록하고자 하는 경우 사용한다.	// systemTest이름으로 빈에 자동설정하고자 하는 경우 @Component public class SystemTest {}
@Controller	Controller 클래스로 spring이 등록한다.	// egovCategoryController 클래스를 Controller로 등록 @Controller public class EgovCategoryController {}
@SessionAttributes	HttpSession에 저장할 모델객체를 지정한다. *SessionStatus.setComplete() 메서드가 사용될 경우에는 SessionStatus 세션에 저장된 모델 객체는 삭제된다.	// CategoryVO 객체를 HttpSession에 저장하여 Controller에서 사용 @SessionAttributes(types=CategoryVO.class) public class EgovCategoryController {}
@RequestMapping	HTTP Request에 대한 매핑 정보를 바로 해당 Controller 클래스의 메소드로 지정한다.	// *.do 요청을 해당 함수에서 처리 @RequestMapping("/sale/insertCategor

		yView.do") public String insertCategoryView()
@ModelAttribute	뷰에서 필요로 하는 데이터를 모델에 전달하고자 하는 경우 혹은 뷰에서 전달하는 모델을 Controller로 전달하고자 하는 경우에 사용한다.	// searchVO 모델을 Controller로 전달하고 resultList를 Controller에서 뷰로 전달 public @ModelAttribute("resultList") List selectCategoryList(@ModelAttribute("searchVO") DefaultVO searchVO) {}
@RequestParam	HTTP 요청 파라미터를 매핑하는 경우에 사용한다.	// selectedId 요청 파라미터를 id 로 전달 public String updateCategoryView(@RequestParam("selectedId") String id , @ModelAttribute("searchVO") DefaultVO searchVO, Model model){}
@Service	서비스 클래스의 구현체를 지정한다.	// Service 클래스를 구현하는 Impl 클래스임을 명시 @Service("EgovCategoryService") public class EgovCategoryServiceImpl

* context-common.xml파일에서는 다음과 같은 설정을 통하여 클래스를 자동으로 스캔하여 빈으로 등록한다.
  - <context:component-scan base-package="egovframework"/>

5.2 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 별도의 Exception처리가 필요한 경우를 제외하고는 데이터 처리 오류관련 Exception을 처리하지 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 egovframework.rte.fdl.sale.service.impl;

import egovframework.rte.fdl.common.AbstractDAO;
import egovframework.rte.fdl.sale.service.CategoryVO;

import java.util.List;
import java.util.Map;

import org.springframework.stereotype.Repository;


@Repository("categoryDAO")
public class CategoryDAO extends EgovAbstractDAO{

    public String insertCategory(CategoryVO vo) throws Exception {
        return (String)insert("CategoryDAO.insertCategory", vo);
    }

    public void updateCategory(CategoryVO vo) throws Exception {
        update("CategoryDAO.updateCategory", vo);
    }

    public void deleteCategory(CategoryVO vo) throws Exception {
        delete("CategoryDAO.deleteCategory", vo);
    }

    public CategoryVO selectCategory(CategoryVO vo) throws Exception {
    return (CategoryVO) selectByPk("CategoryDAO.selectCategory", vo);
    }

    public List selectCategoryList(Map searchMap) throws Exception {
        return list("CategoryDAO.selectCategoryList", searchMap);
    }

}
```

○ DAO에서는 iBATIS기반의 기본적인 처리유형(템플릿)을 제공한다. 각 클래스는 추상클래스은 EgovAbstractDAO의 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 입력 데이터를 세팅한 파라메터 객체(보통 VO 또는 Map) @return 입력 시 selectKey를 사용하여 key를 딴 경우 해당 key
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 입력 데이터(key 조건 및 변경 데이터)를 세팅한 파라메터 객체(보통 VO 또는 Map) @return DBMS가 지원하는 경우 update 적용 결과 count
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 입력 데이터(일반적으로 key 조건)를 세팅한 파라메터 객체(보통 VO 또는 Map) @return DBMS가 지원하는 경우 delete 적용 결과 count
public Object selectByPk(String queryId, Object parameterObject)	pk 를 조건으로 한 단건조회 처리 SQL mapping을 실행한다. @param queryId - 단건 조회 처리 SQL mapping 쿼리 ID @param parameterObject - 단건 조회 처리 SQL mapping 입력 데이터(key)를 세팅한 파라메터 객체(보통 VO 또는 Map) @return 결과 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 단일 결과 객체(보통 VO 또는 Map)
public List list(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 List 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 결과 객체(보통 VO 또는 Map)의 List
public List listWithPaging(String queryId, Object parameterObject, int pageIndex, int pageSize)	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 pageIndex - 현재 페이지 번호 @param pageSize - 한 페이지 조회 수(pageSize) @return 부분 범위 결과 List 객체 - SQL mapping 파일에서 지정한 resultClass/resultMap 에 의한 부분 범위 결과 객체(보통 VO 또는 Map) List

5.3 Service 클래스 개발가이드

○ Service 및 ServiceImpl 클래스는 비즈니스 로직을 처리하는 클래스이다.

○ ServiceImpl 클래스는 AbstractServiceImpl 공통 클래스를 상속받는다.

○ Exception이 발생한 경우에는 AbstractServiceImpl 클래스의 processException 메소드를 이용하여 처리하도록 한다.

○ 다음은 ServiceImpl클래스의 구현 샘플이며, 이를 기본으로 확장하여 사용하도록 한다.

```
package egovframework.rte.fdl.sale.service.impl;

import java.util.List;
import java.util.Map;

import javax.annotation.Resource;

import org.springframework.stereotype.Service;

import egovframework.rte.fdl.common.AbstractServiceImpl;
import egovframework.rte.fdl.exception.EgovBizException
import egovframework.rte.fdl.sale.service.EgovCategoryService;
import egovframework.rte.fdl.sale.service.CategoryVO;

@Service("EgovCategoryService")
public class EgovCategoryServiceImpl extends AbstractServiceImpl implements
        EgovCategoryService {

    @Resource(name="categoryDAO")
    private CategoryDAO categoryDAO

    public String insertCategory(CategoryVO vo) throws Exception {
        return categoryDAO.insertCategory(vo);
    }

    public void updateCategory(CategoryVO vo) throws Exception {
        categoryDAO.updateCategory(vo);
    }

    public void deleteCategory(CategoryVO vo) throws Exception {
        categoryDAO.deleteCategory(vo);
    }

    public CategoryVO selectCategory(CategoryVO vo) throws Exception {
        CategoryVO resultVO = categoryDAO.selectCategory(vo);

        if (resultVO == null)
            throw processException("info.nodata.msg");
        return resultVO;
    }

    public List selectCategoryList(Map searchMap) throws Exception {
        return categoryDAO.selectCategoryList(searchMap);
    }

}
```

5.4 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 egovframework.rte.fdl.sale.web;

import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.bind.support.SessionStatus;

import egovframework.rte.fdl.common.DefaultVO;
import egovframework.rte.fdl.sale.service.EgovCategoryService;
import egovframework.rte.fdl.sale.service.CategoryVO


@Controller
@SessionAttributes(types=CategoryVO.class)
public class EgovCategoryController {

    @Resource(name = "EgovCategoryService")
    private EgovCategoryService egovCategoryService

    @RequestMapping("/sale/listCategory.do")
    public @ModelAttribute("resultList")
    List selectCategoryList(@ModelAttribute("searchVO") DefaultVO searchVO)
            throws Exception {
        Map searchMap = PropertyUtils.describe(searchVO);
        return egovCategoryService.selectCategoryList(searchMap);
    }

    @RequestMapping("/sale/insertCategoryView.do")
    public String insertCategoryView(
            @ModelAttribute("searchVO") DefaultVO searchVO, Model model)
            throws Exception {
        model.addAttribute("categoryVO", new CategoryVO());
        return "/sale/registerCategory";
    }
```

```
    @RequestMapping("/sale/updateCategoryView.do")
    public String updateCategoryView(
            @RequestParam("selectedId") String id ,
            @ModelAttribute("searchVO") DefaultVO searchVO, Model model)
            throws Exception {
        CategoryVO categoryVO = new CategoryVO();
        categoryVO.setId(id);
        model.addAttribute(selectCategory(categoryVO, searchVO));
        return "/sale/registerCategory";
    }

    @RequestMapping("/sale/selectCategory.do")
    public @ModelAttribute("categoryVO")
    CategoryVO selectCategory(
            CategoryVO categoryVO,
            @ModelAttribute("searchVO") DefaultVO searchVO) throws Exception {
        return egovCategoryService.selectCategory(categoryVO);
    }

    @RequestMapping("/sale/updateCategory.do")
    public String updateCategory(
            CategoryVO categoryVO,
            @ModelAttribute("searchVO") DefaultVO searchVO, SessionStatus status)
            throws Exception {
    egovCategoryService.updateCategory(categoryVO);
        status.setComplete();
        return "forward:/sale/listCategory.do"
    }

    @RequestMapping("/sale/deleteCategory.do")
    public String deleteCategory(
            CategoryVO categoryVO,
            @ModelAttribute("searchVO") DefaultVO searchVO, SessionStatus status)
            throws Exception {
    egovCategoryService.deleteCategory(categoryVO);
        status.setComplete();
        return "forward:/sale/listCategory.do"
    }

}
```

5.5 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 validator이용)

```
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<!-- head 정보 -->
<head>
<link rel="stylesheet" type="text/css" href="<c:url value='/css/boardTemplate.css'/>" />
<title>Board List Basic</title>
</head>

<!-- body 정보 -->
<body>
<form name="listForm" action="<c:url value='/sale/listCategory.do'/>" method="post"
class="boardListForm">
        <fieldset>
                <legend>List Category</legend>
                <input name="selectedId" type="hidden" />
                <table border="1" cellspacing="0" class="boardList" summary="List of Category">
                        <thead>
                                <tr>
                                        <th scope="col">No.</th>
                                        <th scope="col">카테고리 ID</th>
                                        <th scope="col">카테고리 명</th>
                                </tr>
                        </thead>
                        <tbody>
        <c:forEach var="result" items="${resultList}" varStatus="status">
                        <tr class="<c:out value="${(status.count % 2) == 0 ? 'bg1' : 'bg2' }"/>">
                                <td class="num"><c:out value="${status.count}"/></td>
                                <td class="check">
                                <input name="checkField" type="checkbox" class="inputCheck" />
                                <input name="checkId" type="hidden"
                                  value="<c:out value='${result.name}'/>"/>
                                </td>
                                <td class="category">
                                 <span onclick="javascript:fncSelectCategory('<c:out
value="${result.id}"/>')"><c:out value="${result.id}"/></span></td>
                                <td class="name"><c:out value="${result.name}"/></td>
                                <td class="category"><c:out value="${result.useYn}"/></td>
                                <td class="title"><c:out value="${result.description}"/></td>
                                <td class="author"><c:out value="${result.regUser}"/></td>
                        </tr>
                        </c:forEach>
                    </tbody>
                </table>
        </fieldset>

</form>
</body>
</html>
```

5.6 표준 Sequence Diagram

○ 위의 각 레이어별 처리 sequence는 다음 그림과 같다.