**개발표준 가이드**

**개발표준 가이드**

**Ver. 0.1**

**2017. 05.**

**개정이력**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **버전** | **작성일** | **변경내용** | **작성자** | **승인자** |
| V0.1 | 2017-05 | 최초작성 | 임상건 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목 차

[1. 개요 3](#_Toc418668701)

[2. 어플리케이션 구조도 3](#_Toc418668702)

[3. 프로젝트 구조 4](#_Toc418668703)

[4. 뷰 페이지 관련 6](#_Toc418668709)

[4.1 공통 6](#_Toc418668710)

[4.2 JSP 7](#_Toc418668715)

[4.3 JAVASCRIPT 9](#_Toc418668723)

[5. 서버 사이드 관련 11](#_Toc418668735)

[5.1 Java Source File 규칙 11](#_Toc418668736)

[5.2 Java Class / Interface 명명 규칙 11](#_Toc418668740)

[5.3 Java Method 12](#_Toc418668743)

[5.4 Java 변수 및 상수 14](#_Toc418668753)

[5.5 Java 주석 15](#_Toc418668764)

[5.6 Java 소스 작성 16](#_Toc418668770)

[5.7 log 작성 18](#_Toc418668786)

[5.8 Java Transaction 19](#_Toc418668793)

[5.9 예외처리 19](#_Toc418668797)

[6. SQL 관련 20](#_Toc418668799)

[7. 시큐어 코딩 가이드 21](#_Toc418668810)

# 개요

본 문서는 SKC인프라서비스\_선로운용 프로젝트의 개발 표준 절차를 가이드 함으로써, 개발의 생산성과 효율성을 향상하고, 개발된 결과물의 통일성을 유지하여 향후 유지보수의 편의성, 높은 가독성을 제공하기 위함이다.

# 어플리케이션 구조도

DB

데이터처리

사용자

화면

처리요청

업무처리

연계/배치처리

외부

시스템

공통기반

|  |  |
| --- | --- |
| 처리 요청 | 업무처리 서비스와 사용자 간의 인터페이스를 담당하는 서비스로 사용자 화면 구성 및 사용자 입력 정보 검증 등의 기능을 지원 |
| 업무 처리 | 업무 프로그램의 로직을 담당하며 흐름제어, 트랜잭션 관리, 에러 처리 등의 기능을 지원 |
| 데이터 처리 | 업무 처리 서비스의 데이터 처리에 대한 CRUD 기능 지원 |
| 연계 / 배치 처리 | 타 시스템 연동 및 일괄 데이터 처리를 위한 설정 및 실행 기능 지원 |
| 공통기반 | 개발의 생산성과 효율성을 위한 공통 서비스 지원 |

# 프로젝트 구조

## SKC인프라서비스\_선로운용Domain\_V3

SKC인프라서비스\_선로운용 공통 및 로직이 들어있는 프로젝트

java project로서 다른 프로젝트들이 참조하는 프로젝트

### 프레임워크 패키지

|  |  |  |  |
| --- | --- | --- | --- |
| src | config | | 프레임워크 관련 설정 파일과 DB 제품군별 sql 파일 관리 |
| SKC인프라서비스\_선로운용.domain |  | java 소스를 관리하는 패키지 |
| construct | 공사관리 관련 패키지 |
| Manage | 운영관리 관련 패키지 |
| bp | 협력사 관련 패키지 |
| calculate | 정산관리 관련 패키지 |
| common | 공통 관련 패키지 |
| community | 업무커뮤니티 관련 패키지 |
| eApproval | 전자결재 관련 패키지 |
| safe | 안전관리 관련 패키지 |
| schedule | 배치 프로그램 관련 패키지 |
| system | 시스템 관련 패키지 |
| doc | | | 프로젝트 관련 문서를 관리하는 패키지 |
| lib | | | 프로젝트 관련 라이브러리를 관리하는 패키지 |

## SKC인프라서비스\_선로운용Front\_V3 : SKC인프라서비스\_선로운용 일반 사용자들이 사용하는 프로젝트

웹 프로젝트, 미구현

## SKC인프라서비스\_선로운용Backoffice\_V3 : SKC인프라서비스\_선로운용 공급사 및 관리자가 사용하는 프로젝트

SKC인프라서비스\_선로운용Domain\_V3 참조하는 웹 프로젝트로서 작업 진행 중

### 프레임워크 패키지

|  |  |  |  |
| --- | --- | --- | --- |
| src | config | 프레임워크 관련 설정 파일 | |
| SKC인프라서비스\_선로운용.backoffice |  | java 소스를 관리하는 패키지 |
| construct | 공사관리 관련 패키지 |
| Manage | 운영관리 관련 패키지 |
| bp | 협력사 관련 패키지 |
| calculate | 정산관리 관련 패키지 |
| common | 공통 관련 패키지 |
| community | 업무커뮤니티 관련 패키지 |
| eApproval | 전자결재 관련 패키지 |
| safe | 안전관리 관련 패키지 |
| system | 시스템 관련 패키지 |
| www | daumeditor | 뷰 페이지에서 사용할 에디터 폴더(다음 에디터) | |
|  | META-INF | 프레임워크 환경 설정 관련 폴더 | |
|  | css |  | 웹페이지에서 사용하는 각종 자원 관련 폴더 |
|  |  | fonts | 웹페이지에서 제공하는 폰트 관련 폴더 |
|  | Jq |  | 웹페이지에서 사용하는 자바스크립트 관련 폴더 |
|  |  | js | 웹페이지에서 사용하는 자바스크립트 관련 폴더 |
|  | img |  | 웹페이지에서 사용하는 이미지 관련 폴더 |
|  | WEB-INF |  | 웹페이지 구동과 관련된 폴더 |
|  |  | jsp | 각 서비스별 jsp 파일이 들어있는 폴더 |
|  |  | lib | 프레임워크 구동에 필요한 jar 파일이 들어있는 폴더 |

# 뷰 페이지 관련

## 공통

### 뷰 페이지 내 경로는 절대 path 사용을 원칙으로 한다.

### Web root 하위로 확장자 별로 directory를 구성하고, 하위로 system 명과 sub-system명을 directory로 한다. (사용자의 직접 접근을 허용하지 않는 경우 WEB-INF 폴더 하위에 작성하도록 한다.)

### directory 명은 영문 소문자만 사용하도록 한다.

### 파일 인코딩은 UTF-8을 기본으로 한다. 단 파일의 특성상 다른 인코딩 방식이 필요한 경우 PM(Project Manager)의 허가를 받아 처리하도록 한다.

## JSP

### 명명규칙

#### jsp 파일은 기본적으로 Web root/WEB-INF/ jsp/ 하위 directory에 위치하는 것을 기본으로 한다. 예외가 필요한 경우 PM의 허가를 받아 위치하도록 한다.

#### jsp 파일의 명명 규칙은 role + [업무용어] + postfix.jsp로 한다. 표기법은 카멜 표기법(Camel Notation)[[1]](#footnote-1)을 사용하도록 한다.

##### role

|  |  |  |
| --- | --- | --- |
| role | 설명 | 적용예 |
| adm | 관리자 | admRoleList.jsp |
| ven | 공급사 | venProductDetail.jsp |

##### postfix

|  |  |  |
| --- | --- | --- |
| postfix | 설명 | 적용예 |
| List | 목록 | admRoleList.jsp |
| Reg | 등록 | admRoleReg.jsp |
| Detail | 상세(+ 수정) | admRoleDetail.jsp |
| Modify | (Only)수정 | admRoleModify.jsp |
| Pop | 팝업 | admRolePop.jsp |

### 코딩 스타일

#### jsp 내에서 공통 및 공통 팝업을 제외하고 jsp include는 원칙적으로 사용하지 않는다.

#### 기본 스크립트릿 사용을 지양하며, EL, JSTL, Spring Custom tag library를 사용하도록 한다.

### 스타일은 css을 사용하도록 한다.

### 해당 jsp 파일에서만 사용하는 javascript를 제외하고는 jsp에 작성하지 않는다.

### 웹 화면은 웹표준 및 접근성 지침을 준수하도록 한다.(크로스브라우징 지원)

크로스 브라우징 지원 범위는 프로젝트마다 상이하므로 PM이 사전에 고객과의 협의를 통해 범위를 정의하도록 한다.(default : IE 9 이상, firefox, chrome, safari, opera)

### 주석은 <%-- --%>만 사용하도록 한다. <!-- --> 사용은 지양하도록 한다.

### 텍스트 정렬

|  |  |
| --- | --- |
| 숫자 | 오른쪽 정렬 |
| 데이터 길이가 동일한 경우 | 가운데 정렬 |
| 그 외 | 왼쪽 정렬 |
| 버튼 | 오른쪽 정렬 |

### BACK OFICEE JSP 순서는 다음과 같이 구성된다.

|  |  |
| --- | --- |
| page 선언부 | <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%> |
| Page import | 페이지 script 작성시 필요한 경우 사용  Ex) <%@ page import=*"SKC인프라서비스\_선로운용.domain.common.dto.CodesDto"*%> |
| Page include | 페이지 공통 부분 include  1. 일반 페이지  <%@ include file=*"/WEB-INF/jsp/backoffice/common/includeStart.jsp"* %>  2. 팝업 페이지  <%@ include file=*"/WEB-INF/jsp/backoffice/common/includePopStart.jsp"* %> |
| Page script 작성 | 페이지에 작성에 필요한 script 작성  Ex) String srcEndDt = CommonUtils.getCurrentDate(); |
| 검색조건 HTML 작성 | <div class=*"col-xs-12"*>  <div class=*"row"*>  <div class=*"widget-box collapsed"*>  <div class=*"widget-header"*>  <div class=*"widget-title"*>  <%-- 검색 테이블 시작 --%>  <table width=*"100%"* border=*"0"* cellspacing=*"0"* cellpadding=*"0"*>  <tr>  <%-- 검색 조건명 start!--%>  <td class=*"td\_sch\_title7"*>주문번호</td>  <%-- 검색 조건명 end!--%>  <%-- 검색 조건 값 start!--%>  <td class=*"td\_sch\_content20"*>  <input id=*"srcOiNo"* name=*"srcOiNo"* type=*"text"* value=*""* maxlength=*"21"* size=*"15"*/>  </td>  <%-- 검색 조건 값 end!--%>  ...  <td class=*"td\_sch\_title6"* rowspan=*"3"*>  <div class=*"widget-toolbar"*>  <small>  <%-- 검색 조건 이벤트 버튼 start --%>  <button class=*"btn btn-sm btn-white btn-info btn-bold"* id=*"btn\_search"*>  <i class=*"ace-icon fa fa-search bigger-120 blue"*></i>  <b>조회</b>  </button>  <%-- 검색 조건 이벤트 버튼 end --%>  </small>  </div>  </td>  </tr>  <tr>  ...  </tr>  </table>  <%-- 검색 테이블 종료 --%>  </div>  </div>  </div>  </div>  </div> |
| 그리드 HTML 작성 | <table>  <tr>  <td align=*"right"* valign=*"bottom"*>  <%-- 그리드 이벤트 버튼 start! --%>  <img id=*"regButton"* src=*"/resource/img/backoffice/icon/Add.gif"* class=*'gridBtnRole'* style='cursor: *pointer*;' title=*"등록"* />  <img id=*"modButton"* src=*"/resource/img/backoffice/icon/Modify.gif"* class=*'gridBtnRole'* style='cursor: *pointer*;' title=*"수정"* />  <img id=*"delButton"* src=*"/resource/img/backoffice/icon/Trash.gif"* class=*'gridBtnRole'* style='cursor: *pointer*;' title=*"삭제"* />  <img id=*"excelButton"* src=*"/resource/img/backoffice/icon/Table.gif"* class=*'gridBtnRole'* style='cursor: *pointer*;' title=*"엑셀다운로드"* />  <img id=*"uploadButton2"* src=*"/resource/img/backoffice/icon/Upload.gif"* class=*'gridBtnRole'* style='cursor: *pointer*;' title=*"코드일괄등록"* />  <%-- 그리드 이벤트 버튼 start! --%>  </td>  </tr>  <tr>  <td valign=*"top"*>  <%-- 그리드 start! --%>  <div id=*"jqgrid"*>  <table id=*"list"*></table>  <div id=*"pager"*></div>  </div>  <%-- 그리드 end! --%>  </td>  </tr>  </table> |
| Javascript 작성 | <script type=*"text/javascript"*>  $(document).ready(**function**() {  });  ...  </script> |

## JAVASCRIPT

### 공통 파일 담당자가 공통 자바스크립트를 담당하며 필요한 기능 추가 및 수정은 공통 파일 담당자를 통해 작업이 되도록 한다.

### 크로스 브라우징 기능을 지원하기 위해 JQuery 사용을 권장하며 JQuery 사용하지 않을 때는 크로스 브라우징 여부를 반드시 확인한다.

### 특정 기능 제공이 아닌 경우 javascript는 </head> 태그 상단에 하나만 위치하도록 하며 소스는 일관성 있는 형식으로 작성하도록 한다.

### 전역 변수는 <script> 태그 바로 아래에 위치하도록 한다.

### body 태그의 onload 이벤트 또는 jquery의 document.ready 이벤트는 전역 변수 아래에 위치하도록 하며 되도록 여러 곳에 나누어 생성하지 않고 한곳에 작성하도록 한다.

### 하나의 function에 여러 기능이 기술되지 않도록 노력하며 function의 길이도 너무 길지 않도록 작성한다.(단 유효성 검사 function의 길이는 무방하다)

### 변수와 function명은 카멜 표기법(Camel Notation)을 사용하도록 한다. function 명은 fn으로 시작하도록 한다.

### function 선언부는 들여쓰기를 하지 않으며 구현부는 tab키를 사용하여 들여쓰기로 구현시작하며 구현 소스 중에는 스페이스 만을 사용하도록 한다.

ex)

<script type="text/javascript">

function fnCloseButtonOnClick(){

var tempApp = 10;

$("#close").click();

}

</script>

### if 나 else, for, while, try를 사용할 때는 괄호를 사용하며 여러 줄로 나누어 사용한다.

ex)

if(condition){

// 코드

}

while(condition){

// 코드

}

var i = 0;

for(i = 0; i < 100; i++){

// 코드

}

### 변수 선언은 함수의 시작하는 부분에 위치하도록 한다.

ex)

function fnTest(){

var bar = "";

// 코드

}

### history.back() 메소드는 사용하지 않도록 한다.

### 주요 공통 자바스크립트 펑션

#### 엑셀다운로드 펑션

##### fnExportExcel : 페이징 처리가 없는 JQGRID의 정보를 엑셀 다운로드

##### fnExportExcelToSvc : 페이징 처리가 있는 JQGRID의 정보를 엑셀 다운로드

호출 URL 정보는 General Controller 호출 부분을 참조한다.

##### fnAllSheetExportExcelToSvc : 여러 sheet에 걸친 엑셀 다운로드

#### 문자열의 바이트 수를 반환하는 펑션 : getByteLength

#### 특정 url로 post 방식으로 이동하는 펑션 : fnDynamicForm

#### 이벤트 객체의 키코드 값을 반환하는 펑션 : getKeyCode

#### 숫자 입력 여부를 반환하는 펑션 : fnOnlyNumber

#### 이메일 유효성을 검사하는 펑션 : fnEmailChk

# 서버 사이드 관련

## Java Source File 규칙

### 각각의 자바 소스 파일은 하나의 public 클래스 또는 인터페이스를 가진다.

### 자바소스 파일은 package 문장, import 문장, 클래스와 인터페이스 선언 순서를 가진다.

#### Java Package 명명 규칙

|  |  |
| --- | --- |
| 설명 | 규칙 |
| 도메인 영역 시스템 | SKC인프라서비스\_선로운용.domain.[시스템명].[서브시스템명] |
| 벡오피스 영역 시스템 | SKC인프라서비스\_선로운용.backoffice.[시스템명] .[서브시스템명] |
| 프론트 영역 시스템 | SKC인프라서비스\_선로운용.front.[시스템명] .[서브시스템명] |

### 소스 한 줄의 길이는 80자 이내를 권장한다.

#### 소스 한 줄의 길이가 너무 길 경우 아래 원칙에 의해 분리할 것을 권장한다.

|  |
| --- |
| 콤마 후에 분리한다. |
| 연산자(Operation) 전에 분리한다. |
| 레벨이 낮은 원칙 보다는 레벨이 높은 원칙에 따라서 분리한다. |
| 앞줄과 같은 expression이 시작되는 새로운 줄은 앞줄과 들여쓰기를 일치시킨다. |

## Java Class / Interface 명명 규칙

### Java Class / Interface 명명 규칙은 [업무용어] + postfix.java로 하며 카멜 표기법(Camel Notation)을 사용하되 맨처음은 영문 대문자를 사용한다.

|  |  |  |
| --- | --- | --- |
| postfix | 설명 | 활용예 |
| Ctl | 컨트롤러 | BbsCtl |
| Svc | 서비스 | AccountSvc |
| Dto | 데이터 전달 객체 | DisplayDto |

### 클래스는 다음 순서로 구성된다.

|  |  |
| --- | --- |
| 클래스 / 인터페이스 문서 주석 | 생략 가능 |
| 클래스 / 인터페이스 문장 |  |
| 클래스 / 인터페이스 구현 주석 | 생략 가능 |
| 클래스 Static 변수 | 생략 가능, 권장하지 않음 |
| 클래스 일반 변수 | 생략가능, 권장하지 않음 |
| 생성자 | 생략 가능 |
| 메소드 | 접근자별 정렬 보다는 로직별 정렬 추천 |

## Java Method

### java method의 명명규칙은 prefix + [업무용어] + postfix로 하며 카멜 표기법(Camel Notation)을 사용하도록 한다.

|  |  |  |  |
| --- | --- | --- | --- |
| 유형 | prefix | postfix | 활용예 |
| 등록 | insert |  | insertAccount |
| 수정 | update |  | updateAccount |
| 삭제 | delete |  | deleteAccount |
| 리스트 조회 | select | List | selectAccountList |
| 단건 조회 | select | Info | selectAccountInfo |
| 카운트 조회 | select | \_count | selectAccount\_count |

### 메소드 이름과 그 메서드의 파라미터 리스트의 시작인 괄호 '(' 사이에는 스페이스가 없어야 한다.

### 여는 중괄호 '{'는 선언문과 같은 줄의 끝에 나타난다.

### 닫는 중괄호 '}'는 '{' 후에 즉시 나타나야 하는 null 문장일 경우를 제외하고는 여는 문장과 일치하는 들여쓰기를 하는 새로운 줄에서 시작해야 한다.

### 메소드를 구분하기 위해서 각 메서드를 사이에 비어있는 한 줄을 사용한다.

ex)

class Sample extends Object{

int ivar = 1;

int ivar = 2;

Sample(int i, int j){

ivar1 = i;

ivar2 = j;

}

}

### 메소든 명은 재미나거나 기발한 명보다는 명료하게 작성하도록 한다.

### 메소드의 길이는 최대한 짧게 작성하도록 한다.(모니터에 메소드 전체가 보일수 있도록 권장)

### 메소드의 인수는 3개를 넘지 않도록 권장하며 여러 개의 파라미터를 넘겨야 하는 경우에는 DTO[[2]](#footnote-2)나 Map을 이용하여 인자를 넘기도록 한다.

### 밀접한 관련 이있는 메소드는 서로 가까이 위치하도록 한다.

## Java 변수 및 상수

### java 변수의 명명규칙은 업무용어를 조합하여 30자 이내로 명명하고 카멜 표기법(Camel Notation)을 사용하도록 한다.

### 변수 하나당 하나의 선언문을 갖도록 한다.

### 변수가 선언될 때 값 초기화를 하도록 한다. 어떤 계산에 의해 변수의 값이 정해진다면 초기화를 하지 않을 수 있다.

### 변수 선언은 블록의 시작 위치에 있어야 한다.(변수 선언 위치가 다를 경우 소스 이해에 어려움을 겪거나 범위 내에서만 존재하는 변수와 혼동될 수 있다. ) 단 for 문장 내에서의 변수 선언은 허용한다.

ex)

for(int i = 0; i < 100; i++){

// 코드

}

### loop index에서 사용하는 변수는 i, j, k, x, y, z 등을 (관용적으로) 사용할 수 있다.(소문자 l은 1과 혼동 가능성이 있다)

### 일반변수는 '\_'(Underline), '$' 사용을 하지 않는다. (단, DB의 속성명을 그대로 사용하는 경우에는 '\_' 사용을 허용)

### static, final 변수는 영문 대문자로만 구성하며 단어 사이는 '\_'를 사용하여 구분한다.

### 변수명 작성시 서로 흡사한 이름을 사용하지 않도록 주의한다.

### 소문자 l과 1, 대문자 O와 0 같이 유사한 문자로 변수명을 짓지 않도록 한다.

### 변수에는 접두어를 사용하지 않는다.

## Java 주석

### 주석은 코드에 대한 개요와 코드 자체만 가지고 이용할 수 없는 추가적인 정보들을 제공하기 위해 사용되어져야 한다.

### 주석은 프로그램을 읽는 것과 이해하는 것에 관계된 정보만을 포함하고 있어야 한다.

### 주석의 빈번한 발생은 때때로 코드의 질을 떨어트릴 수 있음을 인지하도록 한다.

### 주석을 별표 또는 다른 문자를 이용하여 그려진 큰 사각형에 넣거나 특수 문자를 포함하지 않도록 한다.

### 특별한 경우가 아니라면 소스를 주석처리 하지 않는다.(필요한 경우 주석 사유와 주석 시작점 종료점을 명확하게 나타내도록 한다.)

## Java 소스 작성

### 각각의 줄에는 최대한 하나의 문장만 사용하도록 한다.

### 복합문장은 중괄호 "{ 문장들 }"로 둘러싸여진 문장들의 리스트를 포함하는 문장이다.

|  |
| --- |
| 둘러쌓여진 문장들은 복합 문장보다 한 레벨 더 들여쓰기를 해야 한다. |
| 여는 중괄호는 복합 문장을 시작하는 줄의 마지막에 위치해야 한다. |
| 닫는 중괄호는 새로운 줄에 써야 하고, 복합 문장의 시작과 같은 들여쓰기를 해야 한다. |

### 값을 가지는 return 문장은 어떤 방법으로 더 확실한 return 값을 가지는 경우를 제외하고는 괄호를 사용해서는 안된다.

### if, if-else, if-else if-else 문장

#### 항상 중괄호를 사용하도록 한다.

#### 아래와 같은 예제의 형식으로 사용하도록 한다.

ex)

|  |  |  |
| --- | --- | --- |
| if(condition){  // code  } | if(condition){  // code  }  else{  // code  } | if(condition){  // code  }  else if(condition2){  // code  }  else{  // code  } |

### for 문장

#### for 문장내 initialization 또는 update 절에서 여러 변수를 사용하여 연산하지 않도록 한다.

#### 항상 중괄호를 사용하도록 한다.

### while, do-while 문장은 항상 중괄호를 사용하도록 한다.

### 빈 줄과 빈 스페이스는 코드의 가독성을 향상시키기 위해 적절하게 사용하도록 한다. (아예 없거나 너무 많은 경우 소스 수정에 어려움이 발생한다.)

### 어떤 인스턴스 또는 클래스 변수를 합단한 이유없이 public으로 선언하지 않는다.

### 한 문장에서 같은 값을 여러 개의 변수들에게 할당하지 않는다.

### 여러 연산자를 한 문장에서 사용하는 경우 괄호를 넣어주도록 한다.

ex)

|  |  |
| --- | --- |
| 권장하지 않음 | 권장 |
| if(a == b && c == d){  // 권장하지 않음  } | if((a == b) && (c == d)){  // 권장  } |

### 3항 연산자 보다는 if-else 문을 사용하도록 한다.

ex)

|  |  |
| --- | --- |
| 권장하지 않음 | 권장 |
| condition?x:-x; | if(condition){  x;  }  else{  -x;  } |

### 임의의 가비지 콜렉션은 사용하지 않는다.

#### System.gc(); 사용금지

#### Runtime,getRuntime().gc(); 사용금지

### System.exit(0); 사용하지 않는다.

### 문자열 작성시 '+' 연산자 사용하지 않고 StringBuffer or StringBuild 객체를 사용한다.(객체 사용하여 문자열 작성시 내부에 '+' 가 들어가 있으면 의미 객체 사용 의미 없음)

### 창발적 설계로 깔끔한 코드를 구현하도록 한다.

|  |
| --- |
| 모든 테스트를 수행한다. |
| 중복을 없앤다. |
| 프로그래머의 의도를 표현한다. |
| 클래스와 메소드 수를 최소로 줄인다. |

## log 작성

### System.out.println()과 e.printstacktrace() 메소드는 사용하지 않는다.

### 시스템 차원에서는 서비스가 요청되는 시점부터 모든 처리 내용을 자동으로 로깅하도록 처리한다.

### 로그처리를 위해 log4J를 사용하며, 상세설정에 대한 부분은log4J.properties 파일을 참고한다.

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

### 로그에 주요 개인정보가 기록되지 않도록 한다.

### 로그는 가급적 필요한 부분에만 작성하도록 한다.

## Java Transaction

### Application 개발자가 직접 commit/rollback을 처리하지 않고 Framework에서 처리하도록 한다.

### 업무처리상 rollback을 수행해야 하는 경우에는, Exception을 발생시켜서 rollback 처리를 한다

### 어플리케이션 개발자가 Service에서 Transaction 처리를 하기 위한 설정을 @Transactional Annotation 을 사용하여 설정한다

## 예외처리

### 예외 발생시 처리해야 하는 항목

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

#### CUD transaction이 수행중인 경우에는 트랜잭션 서비스와 연계되어 rollback을 수행한다.

#### 해당 Exception에 대해서 사용자에게 message를 제공하되 시스템 정보나 로그 정보가 담긴 메세지가 출력되지 않도록 한다.

# SQL 관련

## sql 쿼리 파일 위치는 /SKC인프라서비스\_선로운용Domain\_V3/src/config/sqlMap 하위 폴더에 [DB상품별].[업무별] 로 위치하도록 한다.

## 각 쿼리 파일은 Mybatis에서 정의한 xml 형식을 유지하여야 한다.

## namespace + 쿼리 아이디는 DB 상품별로 unique 하여야 한다.

## 쿼리 파일이 추가되는 경우에는 DB 상품별 config 파일에 경로가 추가 되어야 한다.

## SQL 표준 작성 규칙

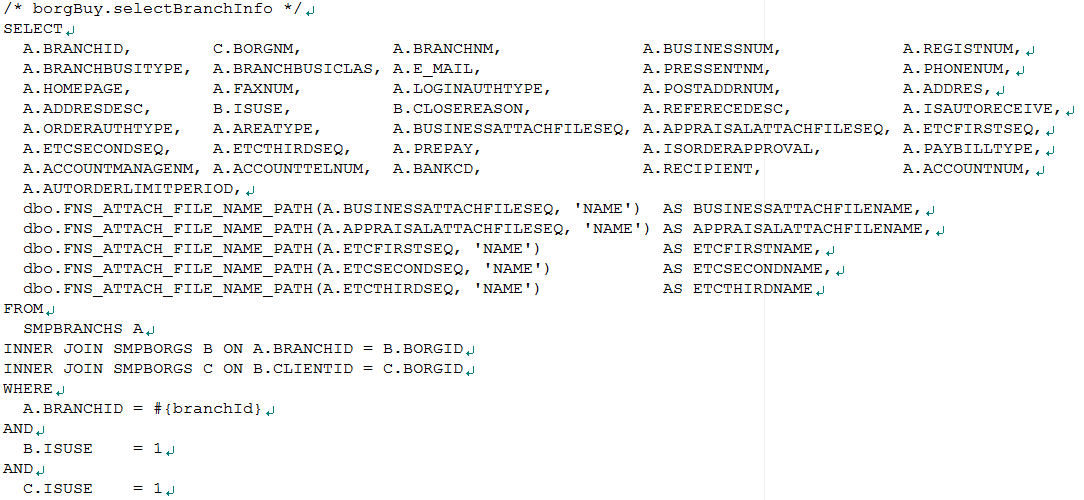
### 단어와 단어 사이 COMMA다음 OPERATOR ( +, -, \*, /, | |).앞뒤, COMPARISON CONDITIONS (=, !=, <, >, <=, >=, <> )앞뒤에는 한 칸을 띄운다.

### 절 (SELECT, FROM, WHERE, ORDER BY 등)은 읽고 편집하기 쉽게 줄을 나눈다.

### 문제 SQL에 대한 추적관리를 위해 쿼리 namespace + ID를 쿼리 맨 앞에 기술한다. HINT와 동시 사용시에는 반드시 HINT를 먼저 기술하고 HINT.와의 구분을 위해 “ ”안에 기입한다.

### 키워드 (SELECT, FROM, WHERE)부분은 들여쓰기 한다.

### 여러 테이블을 조인할 경우 테이블의 Alias명을 알파벳 대문자 한자리인 A, B, C, D… 순으로 부여한다.



# 시큐어 코딩 가이드

## 주요 개인정보는 아래와 같이 개인을 식별할 수 있는 정보이며, 반드시 암호화 되어야 한다.

|  |  |
| --- | --- |
| 카드번호 | 16 자리 중 중간 8자리와 끝의 한 자리를 '\*'로 표시한다. (15자리는 중간 8자리만 암호화 한다.) |
| 주민등록번호 | 13자리 중 마지막 7자리를 '\*'로 표시한다. |
| 유선 및 모바일 번호 | 마지막 4자리 또는 3자리를 '\*'로 표시한다. |
| 계좌번호 | 계좌번호의 1,2,4 자리를 제외한 나머지를 '\*'로 표시한다. |
| 이메일 주소 | 처음 3 자리를 제외하고 나머지를 '\*'로 표시한다. |

### 주요 개인정보 DB 필드에 대해 양방향 암호화 알고리즘으로 암호화 시켜야 한다.

양방향 암호화 알고리즘은 SEED, ARIA 알고리즘 이상의 보안성을 가지는 알고리즘을 사용한다.

### 패스워드 관련 필드는 Hash 알고리즘으로 암호화 시켜야 한다.

단방향 알고리즘은 SHA-256 이상의 보안성을 가지는 알고리즘을 사용한다.

### 주요 개인정보에 대해 HTML에 Hidden 값 설정을 지양한다. 불가피하게 Hidden 값으로 설정할 경우, 해당 필드에 대해 암호화된 값으로 설정 및 타임아웃 기능을 사용한다.

#### 암호화 형식은 시간 + 값을 암호화하여 매번 호출시 변경되도록 한다.

#### 서버에서는 복호화하여 시간 값을 추출하여 현재시간에서 30분이상 경과된 시간이면 거절처리한다.

## 유효성 검사

### HttpRequest의 parameter 체크는 javascript 및 서버사이드에서 중복 체크하도록 함.

## 소스내 하드코딩 지양

### 작업 소스내 jdbc 연결 또는 ip 연결시 하드코딩이 아닌 설정 파일에서 제공받아 작업한다.

## 파일 업로드 정책

### 파일 업로드 기능 제공시 다음 확장자만을 사용하도록 한다.

txt, hwp, xls, ppt, doc, xlsx, pptx, docx, jpg, gif, png, bmp

### 파일 업로드 디렉토리는 웹에서 직접 접근 할 수 없어야 한다. 상대경로를 참조하여 임의로 다운로드 할 수 없어야 한다.

## 관리자 페이지 관리

### 외부/내부 관리자 페이지 접근 Url의 명칭에 admin, administrator 등의 유추 가능한 명칭을 사용하지 않는다.

### 외부제공 관리자 페이지 접근시 ID/PW 검증, 공인인증서 검증, 접근 IP 검증을 수행할 것을 권장한다.

### 외부공중망을 통해 내부 관리자 페이지에 접근하지 못하도록 한다.

## 웹어플리케이션 취약점 체크리스트

### 공통

#### 공격방법

##### 코딩 프로그램이 Client Side에서 수정이 가능한 프로그램은 해킹의 대상이 됨 (javascript, vbscript, ajax, activeX, cooke)

##### 서버로 업로드하는 데이터도 수정이 가능함(웹프락시 툴을 이용하여 서버로 업로드하는 데이터를 수정가능함)

#### 방어방법

##### 최종 통제 메카니즘은 Server Side에서 수행해야 함.

##### 인증이 필요한 프로그램은 반드시 Server Side에서 코딩할 것(화면 콘트롤이 필요한 부분은 자바스크립트 사용하고 반드시 Server Side에서도 다시 한번 코딩할 것)

##### 게시판의 경우 수정, 삭제 요청시 다시 한번 게시자가 수정, 삭제 요청하는지 Server Side에서 코딩할 것.

##### 스크립트 페이지에 로그인정보 저장말것

##### HTML, 자바 스크립트 페이지에 주석처리 하지 말것(Client Side 프로그램에는 주석 처리 하지 말것)

##### Client Side에 중요정보 저장하지 말것

request 사이에 중요 data를 유지하기 위한 hidden 필드 의존으로 인해 발생

구매 폼 같은 경우 제품 가격을 hidden으로 입력하였을 경우 브라우져에 보이지는 않으나 사용자가 제품 구매 등의 행위를 선택시 서버로 전송

##### 웹 서비스의 에러정보 즉 JAVA, DB의 에러정보는 그대로 보여주지 말고 공통 에러 페이지 처리할 것

##### 입력창이 있는 부분은 반드시 Max Length로 길이제한 할 것

##### 중요 정보를 보여주는 화면에 no-cache 설정할 것(안하면 뒤로가기가 수행됨)

##### 부적절한 파라미터에 대한 보안 해결 방법은 입력값 검증을 통해 수행

###### 데이터 유형(문자열, 정수형, 실수형)을 검증, 허용된 문자셋(character set) 검증

###### 최대, 최소길이 제한, NULL 값의 허용여부, 반드시 필요한 인자와 그렇지 않은 인자

###### 중복 허용여부 검증, 숫자의 범위 검증, 타장한 것으로 지정된 값인지 검증

##### DB 쿼리문은 반드시 prepared statement문으로 처리할것

### 공격도구 업로드

#### 공격방법

##### 게시판과 관리자화면 등에서 파일 업로드의 취약점을 이용한 공격

##### 자바스크립트는 해커가 소스 수정을 통한 공격가능

#### 방어방법

##### 파일 확장명을 철저히 검증(Server Side 프로그램으로 수행)

###### 파일 네임의 뒷부분에서 확장명 검증해야함

###### 확장명 필터링시 업로드가 허용된 파일만 가능하도록 코딩

##### 업로드된 파일은 웹에서 열람 불가능한 디렉토리에 이름을 변경하여 저장하고 업로드된 파일명은 DB에 저장하고, 사용자 요구시 해당 파일을 반환하며, DB 내용을 토대로 이름을 복구

##### 파일, 디렉토리 퍼미션 Other에 Write 권한 주면 안됨

### 관리자 페이지 강제처리 공격

#### 공격방법

##### 인증우회하여 관리자 페이지 접근

##### Url Guessing을 통한 공격

#### 방어방법

##### 관리자 페이지 디렉토리는 admin 디렉토리를 사용하면 안되며, 디렉토리명은 알수 없는 문자을 이용하여 관리

##### 외부관리자는 반드시 공인인증서를 사용하여 사용자를 인증하여야 한다.

##### 내부 관리자는 IP Blocking 하여 인증을 하여야 한다.

##### 인증을 거치지 않고 직접 URL 접근이 되지 않도록 세션처리한다.

##### 관리자 화면은 반드시 로그인 과정을 거쳐서 서비스가 되어야 한다.

##### 테스트 소스는 반드시 삭제한다.

### File download 취약점

#### 공격방법 : Url 인자로 부적절한 대상 지정

#### 방어방법

##### 웹상에서 받는 외부변수들은 보안에 문제가 될 수 있는 문자열에 대하여 필터링 처리

##### 절대 경로를 통한 인증회피 기능 점검

### SQL 구문 삽입 공격

#### 공격 방법

##### 웹어플리케이션의 GET / POST 인자로 SQL 구문 공격

##### 내부 DB 서버의 데이터 유출, 변조 가능

##### 사용자 및 관리자 인증 우회 공격

#### 방어방법

##### 사용자의 입력값을 철저히 검증(특수문자가 처리되지 않게 처리)

##### 중요한 DB 데이터에 대한 암호화 처리

##### 에러 메세지를 사용자에게 최소로 반환(해커는 에러정보를 보고 Query 정보 파악)

##### DB 쿼리문은 반드시 prepared statement문으로 처리할것

### Blind SQL 구문삽입 공격

#### 방어 방법

##### 웹어플리케이션 코드를 사용자 입력이 SQL 문장으로 사용되지 않도록 수정함

##### 사용자 입력으로 사용이 불가능한 스트링과 사용자 입력으로 사용이 가능한 최소의 문자 집합을 결정, 최소 사용 가능 문자 집합에는 가급적 숫자만을 포함하도록 하고, 그것이 어렵다면 숫자와 무자만을 포함하도록 함.

##### SQL 서버의 에러 메세지를 외부에 제공하기 않도록 함. 에러가 발생한 경우 에러 발생 사실을 보여주기 보다 메인 페이지로 포워딩 되도록 함.

##### 웹 어플리케이션이 사용하는 데이터베이스 사용자 권한을 제한

##### SQL과 관련된 문자열 필터링을 통한 특수 문자 입력을 금지함.

### 검증되지 않은 파라미터 공격

#### 공격방법

##### 클라이언트로부터 웹어플리케이션이 요청을 받았을 때 그 요청이 적절한 값인지 여부를 검증하지 않음으로 인해 백엔드에 존재하는 허가되지 않은 자원에 접근할 수 있는 취약성입니다.

##### 웹 요청들을 강제로 브라우징 한다거나 명령어 삽입등을 강제로 브라우징 한다거나 명령어 삽입등을 통해서 보안 매커니즘을 우회할 수 있게 됩니다.

#### 방어방법

##### 데이터 형식(문자, 정수, 실수 등)

##### 허용되는 문자셋

##### 최소, 최대 허용 길이

##### NULL 값의 사용여부

##### 파라미터의 허용여부

##### 허용되는 숫자 범위

##### 허용되는 인코딩 방식

##### 안전하지 않은 ID 점검

##### URL 강제 접속, 이용인증 회피 가능성 점검

##### 웹컨텐츠 퍼미션 점검

##### 클라이언트의 캐쉬점검 - session reply 공격

##### DB 접속 환경 설정 파일은 웹에서 열람 불가능한 위치로 이동

##### 사용자 세션 관리 점검

### 크로스 사이트 (XSS) 공격

#### 공격 방법 : 쿠키 / 세션 도용, 바이러스 / 해킹도구 설치

#### 방어방법 : 웹상에서 입력되는 문자열에 대하여 특수문자가 처리되지 않도록 처리

### 패스워드 Crack 공격

#### 공격방법 : 패스워드 Guessing을 통한 공격

#### 방어방법

##### 6~8 영어, 숫자 반드시 혼합

##### 단순 비밀번호 사용불가(ID와 4자 이상 중복 / 연속된 글, 숫자 4개 이상)

##### 패스워드 3회 이상 오류시 당일 로그인 불가(온라인 패스워드는 다음날 자동 로그인 횟수 해제)

### Buffer Overflow 공격 / 시스템 명령어 삽입 공격

#### 공격방법

##### 메모리에 할당된 버퍼의 양을 초과하는 데이터를 초과하는 데이터를 입력하여 프로그램의 복귀 주소를 조작하여 해커가 원하는 코드를 실행하는 것

##### 웹 어플리케이션에서 html 쿠키, url 파라미터 형식으로 시스템 명령어 삽입허용함으로써 웹상에서 시스템 명령어를 실행 할 수 있는 취약점

#### 방어방법

##### 웹서버 최신버젼으로 업그레이드

##### 사용자의 입력값을 검증하거나 strncpy와 같이 길이가 정의된 api를 사용

##### 시스템 명령어를 사용하는 모든 사용자의 입력값을 검증

##### 쉘스크립트를 어플리케이션 내에서 사용하지 않는다.

##### 운영체제의 시스템 명령이 사용자 입력에 존재하는지 점검한다.

### 부적절한 오류 처리

#### 공격 강법 : 오류를 아무 고려없이 사용자에게 그대로 반환하여 발생 - 공격에 유용한 정보 제공

#### 방어방법 : 공통 에러 메세지로 처리바람

### 웹프락시를 이용 고객 PC로 다운로드 정보 가로채기

#### 공격방법 : 웹 프락시 툴을 이용하여 클라이언트 PC로 다운로드 되는 쿠키정보 취득

#### 방어방법 : 고객 PC로 다운되는 쿠키정보 및 우회할 수 있는 정보는 반드시 암호화 처리할 것(쿠키 사용 지양)

### GET/POST 취약점

#### 공격방법 : GET 사용시 url 뒷부분에 data 보임

#### 방어방법

##### POST 방식 사용하고 GET 방식 사용금지

##### POST 사용에도 중요 데이터는 암호 처리할 것

# 큐브몰 General Controller URI 안내

## 로그인 필요없는 단순 페이지 이동

### 1 depth 페이지 이동 : /home/페이지명

### 2 depth 페이지 이동 : /home/폴더명/페이지명

## 로그인 체크를 하는 단순 페이지 이동

### 1 depth 페이지 이동 : /menu/페이지명

### 2depth 페이지 이동 : /menu/폴더명/페이지명

### 3depth 페이지 이동 : /menu/폴더명/폴더명/페이지명

## 쿼리 호출을 통한 JSON Object 조회

### 1 depth 조회 : /sql namespace/쿼리아이디/JSON명/object

ex) /bitcube/slectOne/BIT/object를 호출할 경우

bitcube.selectOne 에 해당하는 쿼리문을 수행한 후 BIT라는 객체에 결과를 담아 반환한다.

### 2depth조회 : /sql namespace앞/sql namespace뒤/쿼리아이디/JSON명/object

ex) /oracle/bitcube/slectOne/BIT/object를 호출할 경우

oracle 폴더에 bitcube.selectOne 에 해당하는 쿼리문을 수행한 후 BIT라는 객체에 결과를 담아 반환한다.

## 쿼리 호출을 통한 JQGRID LIST JSON 조회 : JQGRID에서 사용하는 리스트 형식에 맞게 JSON으로 리스트 결과 반환

### 1 depth 조회 : /sql namespace/쿼리아이디/JSON명/list

### 2 depth 조회 : /sql namespace 앞/sql namespace 뒤/쿼리아이디/JSON명/list

## 쿼리 호출을 통한 EXCEL 정보 조회

### 1 depth 조회 : /sql namespace/쿼리아이디/excel

### 2 depth 조회 : /sql namesapce앞/sql namespace 뒤/쿼리아이디/excel

## 트랜잭션 구문 처리

/서비스명/메소드명/save을 호출하여 처리하며 이때 어떤 동작을 처리하는지 여부가 저장된 "oper" 파라미터를 필수를 넘겨주어야 함

/sql namespace 앞/sql namespace 뒤/쿼리아이디/save를 통한 호출은 사용하지 않음

1. 맨 앞의 단어에는 전체 소문자 뒤에 따라오는 단어들은 앞의 1글자만 대문자 나머지는 소문자로 표기한다. [↑](#footnote-ref-1)
2. Data Transfer Object : 데이터가 포함된 객체를 한 시스템에서 다른 시스템으로 전달하는 작업을 처리하는 객체. 흔히 VO(Value Object)와 혼용하여 부르기도 한다. [↑](#footnote-ref-2)