

[EBS 온라인클래스 재구조화 사업]

프로그램 개발표준

버전 1.0

2020.11.16

|  |  |  |  |
| --- | --- | --- | --- |
| **승 인 사 항** | | | |
|  | 성명 / 직급 | 일자 | 서명 |
| 작성자 | 박홍수 / 부장 | 2020.11.16 |  |
| 검토자 |  |  |  |
| 승인자 |  |  |  |
| 고객 |  |  |  |

그리기이(가) 표시된 사진

자동 생성된 설명

문 서 이 력

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 버전 | 제/개정 내용 | 작성일 | 작성자 | 승인자 | 시행일 |
| 0.9 | 최초 작성 | 2020.11.16 | 박홍수 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

목 차

[1. 개요 1](#_Toc57213091)

[2. 공통사항 1](#_Toc57213092)

[2.1. Coding 원칙 1](#_Toc57213093)

[2.2. Coding Style 1](#_Toc57213094)

[2.2.1. Indent 1](#_Toc57213095)

[2.2.2. White Space 1](#_Toc57213096)

[2.2.3. Brace 및 줄바꿈 2](#_Toc57213097)

[3. 주석 2](#_Toc57213098)

[3.1. 자바클래스 주석 2](#_Toc57213099)

[3.2. 메서드 주석 2](#_Toc57213100)

[3.3. 클래스 변수 주석 3](#_Toc57213101)

[3.4. 메소드 내 변수 주석 3](#_Toc57213102)

[3.5. 코드 내 주석 3](#_Toc57213103)

[4. 예외처리 3](#_Toc57213104)

[4.1. 예외 분류 3](#_Toc57213105)

[4.2. 예외 처리 4](#_Toc57213106)

[5. 로깅처리 4](#_Toc57213107)

[6. 트랜잭션처리 5](#_Toc57213108)

[7. DataSource 설정 5](#_Toc57213109)

[8. Naming Rule 5](#_Toc57213110)

[8.1. 표기법 5](#_Toc57213111)

[8.2. Package 5](#_Toc57213112)

[8.2.1. 서비스별 패키지 명명 규칙 6](#_Toc57213113)

[8.3. Controller 클래스 작성 규칙 6](#_Toc57213114)

[8.4. Service 클래스 작성 규칙 6](#_Toc57213115)

[8.5. Mapper 클래스 작성 규칙 7](#_Toc57213116)

[8.6. 클래스 변수 7](#_Toc57213117)

[8.7. 지역변수 7](#_Toc57213118)

[8.8. 메소드 8](#_Toc57213119)

[8.9. 상수 8](#_Toc57213120)

[8.10. DTO 클래스 작성 규칙 8](#_Toc57213121)

[9. Coding Naming Rule 8](#_Toc57213122)

[9.1. 명명규칙 8](#_Toc57213123)

[9.2. 연산유형 8](#_Toc57213124)

[10. SQL 작성규칙 10](#_Toc57213125)

[10.1. 명명규칙 10](#_Toc57213126)

[10.2. Mapping XML ID Naming Rule 10](#_Toc57213127)

[10.3. XML Tag: <parameterMap> 10](#_Toc57213128)

[10.4. XML Tag: <resultMap> 10](#_Toc57213129)

# 개요

본 문서는 EBS 온라인클래스 재구조화 프로젝트 개발 부문에 적용할 표준을 정의한다.

# 공통사항

## Coding 원칙

* 주석은 모든 코드에 상세히 기술하는 것을 원칙으로 한다
* 소스코드는 불가피한 내용을 제외하고 원칙적으로 중복을 금지 한다.
* 소스 코드에서 사용하는 모든 단어는 용어사전의 내용을 기준으로 한다.
* 개발 및 테스트를 수행함에 있어 기능과 성능은 물론 보안에도 각별한 주의를 기울인다.

## Coding Style

* 한 line에 하나의 statement만 기술한다.
* Semicolon, comma, 예약어 뒤에는 space를 둔다
* brace {} 는 새로운 라인에 기술하며, 다른 내용과 함께 기술하지 않는다. (주석은 제외)
* ‘{‘ 는 기존의 ‘{‘ 와 비교하여 indent(4-space)를 준다
* brace에 주석을 기입하는 경우 ‘//’ 주석을 사용 한다.

### Indent

* 표준 indent의 크기는 4space이다.
* indent는 tab을 사용하지 않고, space만 사용한다.

### White Space

* 한 줄에는 하나의 statement만 기술한다.
* semicolon, comma, 예약어 뒤에는 space를 둔다.
* unary operation(단항 연산자)은 space를 두지 않는다. (예: i++;)
* binary operation은 양쪽에 space를 둔다. (예: i = i + 1;)
* 괄호 안에 괄호가 있는 경우에는 괄호 사이에 space를 두지 않는다.

### Brace 및 줄바꿈

* 블록이 한 줄로 구성되더라도 항상 중괄호를 사용해야 한다.
* ‘{’ 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.

# 주석

* 자바클래스 주석은 아래 표준 템플릿을 준수하여 작성, javadoc 으로 생성 및 사용한다.
* 클래스명, 클래스설명, 수정이력, 작성자, 최초작성일, 버전, 참조클래스, 권리 (Copyright)를 명시한다.

## 자바클래스 주석

/\*\*

\* 코드 관리 - 코드 관리 항목은 관리자 화면에서 등록/수정/삭제 할 수 있다. \*

\* 여기서는 관리자 화면에서 등록된 기준정보 데이타를 활용하여 JSP 페이지에서

\* <select><option value="val">text</option></select> 의 html 콤보박스 UI 컴포넌트 지원을 위한 데이터 핸들링을 할 수 있도록 한다.

\*

\* **@author** 홍길동

\* **@date** 201X. XX. XX.

\* **@version** 1.0

\*/

## 메서드 주석

/\*\*

\* **@Method Name** :

\* **@작성일** :

\* **@작성자** :

\* **@변경이력** :

\* **@Method 설명** :

\*/

## 클래스 변수 주석

/\*\* 이름 \*/

private String name;

/\*\* 주소 \*/

## 메소드 내 변수 주석

String name; // 이름

String address // 주소

## 코드 내 주석

* 코드 내 주석은 “//”, “/\* ... \*/” 를 권장하며 후자는 블록 단위의 주석에 사용한다.
* 주석만 있는 라인 전에는 빈 줄을 추가한다. 주석이 늘어날 수 있거나 중요한 내용이면 아래에 빈 줄을 삽입한다.
* 주석은 코드와 같은 공간으로 들여쓰기를 한다.
* 주석은 코드의 내용을 명확하게 이해 할 수 있도록 작성하고 “어떻게” 작성했는가가 아닌 “왜” 작성 했는지에 대해 초점을 둔다.
* 라인 끝의 주석은 라인이 바뀌지 않는 범위 내에서 사용하고 라인이 변경 될 경우 코드의 위에 작성 한다.
* Working comments: 코딩 하는 본인이나 다른 사람을 위하여 해결되지 않은 이슈를 표현하기 위해서는 “\*TODO\*”를 사용한다.

# 예외처리

## 예외 분류

* 시스템 내부 에러 (500 번대 에러코드) 예외 사항으로 에러페이지로 forward 되어야 하는 경우
* 일반 방식으로 요청 처리 시 세션 타임 된 경우.
* Ajax 방식으로 들어 온 요청 처리 시 예외가 발생되어 javascript callback 메서드로 에러메시지를 반환해야 하는 경우
* Ajax 방식으로 들어 온 요청 처리 시 세션 타임 아웃 된 경우
* 로그인 인증 후 접속 해야 하는 페이지에 인증 과정 없이 페이지에 접근 하는 경우
* 시스템 수행 시 개발자가 특정 경우를 캐치하여 예외를 발생 시켜야 하는 경우

## 예외 처리

* 4.1 에서 발생 되는 에러에 대해서 공통에서 처리 할 수 있는 방법을 제공하고 biz 개발자는 try, catch 구문으로 예외를 묵시 하는 경우가 생기지 않도록 개발 함.
* Ajax 요청 처리시 예외가 발생 하는 경우를 처리 하기 위해 공통에서 javascript 단 공통처리 모듈과 서버 단 공통 처리 API를 제공하여 이를 활용 하도록 함.
* 예외에 대한 내용을 EndUser 에게 보여 줄 때 다국어 처리가 된 메시지로 전달 될 수 있도록 함.

# 로깅처리

* 기본 적으로 log4j 의 단점을 보완 하고 있는 slf4j을 사용함
* 로깅 시에는 개발 시에 로그 추적을 용이하도록 하는 목적에 운영 시 에 반드시 남겨야 하는 로그를 판단하여 각 에러레벨 (trace, debug, info, warn, error) 에 맞게 로깅을 처리한다.
* 로그 처리에 대한 설정은 log4j.xml에 그 정보를 기록하며 개발/스테이징/운영 환경에 따라 그 레벨을 조정하여 각각 독립성을 유지하여 관리 할 수 있도록 처리한다.  
  스테이징 및 운영 환경으로 갈 경우에는 톰캣 start 스크립트 안에 다음과 같이 설정하여 –Dghc.log4j=/xxx/xxx/log4j.xml으로 관리 될 수 있도록 함.
* 개발 환경 로그 설정 예시



# 트랜잭션처리

* 트랜잭션 처리에 대한 레이어는 Service 단위이다.
* 즉 특정 업무를 하나의 트랜잭션으로 처리를 하기 위해서는 하나의 Service 클래스 내의 메서드에서 여러 개의 Mapper 메서드를 호출 하는 형태가 된다.
* 이 때 트랜잭션으로 묶고자 하는 Service 클래스 내의 메서드 상단에 @Transactional 을 붙여주면 된다.

# DataSource 설정

* DataSource 에 대한 설정은 개발/스테이징/운영 환경에 맞게 달리 설정 되어야 한다.
* 기본적으로 로컬 환경에서 개발 할 경우에는 [프로젝트]/src/main/resources/application.properties 에 설정해서 사용한다.
* 이 때 여기에 세팅 되는 Database 의 driver, url, user, password 정보는 개발/스테이징/운영 환경에 따라 계정 정보를 찾아 갈 수 있도록 application properties에 환경 구분을 하여

작성한다.

# Naming Rule

## 표기법

|  |  |  |
| --- | --- | --- |
| **표기법 명칭** | **설명** | **예제** |
| Pascal Case  (파스칼 표기법) | 첫 문자를 대문자로 시작하며, 의미 구분을 위해 대문자를 사용 | ContentManagement |
| Camel Case  (카멜 표기법) | 첫 문자를 소문자로 시작하며, 의미 구분을 위해 대문자를 사용 | contentManagement |
| Hungarian Case  (헝가리안 표기법) | 사전에 정의된 접두어를 사용 | strContentManagement |
| Lower Case  (소문자 표기법) | 전체를 소문자로 사용 | contentmanagement |
| Upper Case  (대문자 표기법) | 전체를 대문자로 사용 | CONTENTMANAGEMENT |

## Package

* Lower Case를 원칙으로 작성한다.
* Package는 항상 영문소문자를 사용한다.

### 서비스별 패키지 명명 규칙

|  |  |  |
| --- | --- | --- |
| **구분** | **패키지** | **비고** |
| 인증/인가 |  |  |
| 학교/클래스 |  |  |
| 강좌/강의 |  |  |
| 공통 |  |  |
| 통계 |  |  |
| 연계 |  |  |

## Controller 클래스 작성 규칙

* camel Case를 원칙으로 작성한다.
* [메인테이블 명]+Controller.java 로 작성하되 테이블명이 약어일 경우 풀어서 작성한다.
* 컨트롤러 클래스 안에서 메서드 명을 작성 할 때는 아래와 같이 접미사를 붙인다.
* productList() : 목록 조회 유형의 서비스
* productDetails() : 단 건 상세 조회 유형의 서비스
* productSave() : 등록/수정/삭제 가 동시에 일어나는 유형의 서비스
* productAdd() : 등록만 하는 유형의 서비스
* productModify() : 수정만 하는 유형의 서비스
* productRemove() : 삭제만 하는 유형의 서비스

## Service 클래스 작성 규칙

* camel Case를 원칙으로 작성한다.
* [메인테이블 명]+Service.java로 클래스를 작성한다.
* 서비스 클래스 안에서 메서드 명을 작성 할 때는 아래와 같은 접두사를 붙인다.
* findProduct() : 조회 유형의 서비스
* addProduct() : 등록 유형의 서비스
* modifyProduct() : 변경 유형의 서비스
* removeProduct() : 삭제 유형의 서비스
* saveProduct() : 등록/수정/삭제 가 동시에 일어나는 유형의 서비스

## Mapper 클래스 작성 규칙

* camel Case를 원칙으로 작성한다.
* [메인테이블 명]+Mapper.java로 클래스를 작성한다.
* Mapper 클래스 안에서 메서드 명을 작성 할 때는 아래와 같이 접두사를 붙인다.
* selectProduct(); : 조회 유형의 서비스
* insertProduct(); : 등록 유형의 서비스
* updateProduct(); : 변경 유형의 서비스
* deleteProduct(); : 삭제 유형의 서비스

## 클래스 변수

* Camel Case를 원칙으로 작성한다.
* 클래스 변수는 static 멤버 변수이다.
* 클래스 변수는 “the” 와 같이 헝가리안 접두어를 사용하면 안 되고 ClassName.classVarName” 표기법을 사용하여 명시적으로 참조 해야 한다.. 클래스 변수는 소문자로 시작하고 대문자로 단어 단위를 구분해야 한다.

## 지역변수

* Camel Case를 원칙으로 작성한다.
* 지역 변수는 클래스 필드의 명명 규칙을 그대로 따른다.
* 일반 변수는 "\_" 또는 "$" 사용을 하지 않는다.

## 메소드

* Camel Case를 원칙으로 작성한다.
* “[동사][목적어]” 형태로 작성한다.

예) int getRadius() {

return this.radius;

}

## 상수

* Upper Case를 원칙으로 작성한다.
* 의미 구분을 위해 단어 사이에는 ‘\_’를 사용한다.
* Constant는 클래스의 ”static final”이다. Java는 #define constants를 위한 C언어 규칙을 채택했기 때문에 Upper Case가 기본이다.

예) public static final int DEFAULT\_COLOR = Color.black;

## DTO 클래스 작성 규칙

* Camel Case를 원칙으로 작성한다.
* [Dto명]+Dto.java 로 작성하되 테이블명이 약어일 경우 풀어서 작성한다.
* 각 계층 (Controller, Servie 등) 간 데이터 교환을 위해 업무 목적별로 생성한다.

# Coding Naming Rule

## 명명규칙

* 연산 명은 동사로 짓고, 여러 단어를 사용할 경우 단어의 배열은 [동사 + 명사]의 순서를 따른다.
* 첫 단어는 소문자로 시작하고 다음 단어부터는 대문자로 시작하는 Camel Casing을 적용한다.
* 여러 단어를 연결할 경우 단어 사이의 구분을 위하여 각 단어의 첫 글자만을 대문자로 하고, 나머지는 모두 소문자로 쓴다. (기본적으로 Camel Casing을 적용하며, 약어일 경우도 단어 사이의 혼동을 피하기 위하여 각 단어의 첫 글자만을 대문자로 사용한다.)

## 연산유형

|  |  |  |  |
| --- | --- | --- | --- |
| 연산유형 | 접두사 | 설명 | 예시 |
| 조회(단건) | select | 데이터 단건 조회 연산 | selectUserNm() |
| 조회(다건) | selectList | 데이터 다건 조회 연산으로 다건 조회 | selectListUserNm() |
| 등록 | insert | 데이터 등록 연산 | insertUserNm() |
| 수정 | update | 데이터 수정 연산 | updateUserNm() |
| 저장 | save | 등록과 수정 두가지를 함께 처리하는 연산 | saveUser() |
| 삭제 | delete | 데이터 삭제 연산 | deleteUser() |
| 개수 | count | 데이터 레코드 조회(select count(\*) as from table) | countUser() |
| 처리 | process | 반복적으로 사용되는 로직을 연산으로 분리한 경우 | processUser() |
| 검증(여부) | verify | 데이터의 형식 및 유효성 체크등에 해당하는 연산 | verifyZipCode() |
| VALUE SET | set | 데이터 멤버의 value를 지정하는 연산으로 set 뒤에 대상 영문 명을 쓰는 것을 원칙으로 한다. | setArea() |
| VALUE GET | get | 데이터 멤버의 접근연산으로 get 뒤에 대상 영문 명을 쓰는 것을 원칙으로 한다. | getArea() |
| 계산 | calculate | 일반적으로 파라미터 값을 기반으로 금액을 계산하기 위한 함수 | calculatePrice() |
| 체크 | check | 데이터 존재 유무 등을 확인하는 경우 | checkUserInfo() |
| 시작 | start | 프로세스나 작업등을 시작하기 위한 함수 | startProcess() |
| 재시작 | restart | 프로세스나 작업등을 재시작하기 위한 함수 | restartProcess() |
| 취소 | cancel | 프로세스나 작업등을 취소하기 위한 함수 | cancelProcess() |
| 완료 | complete | 프로세스나 작업등을 완료 처리하기 위한 함수 | completeProcess() |
| 승인 | approve | 프로세스의 작업을 승인하기 위한 함수 | approveUser() |
| 반려 | reject | 프로세스의 작업을 반려하기 위한 함수 | rejectUser() |
| 동기화 | sync | 데이터의 동기화 처리를 위한 함수 | syncUserData() |
| 가져오기 | import | 데이터 가져오기 처리를 위한 함수 | importData() |
| 내보내기 | export | 데이터 내보내기 처리를 위한 함수 | exportData() |
| 송신 | send | 데이터 보내기 처리를 위한 함수 | sendUserData() |
| 수신 | receive | 데이터 받기 처리를 위한 함수 | recieveUserData() |
| 임시저장 | saveTemp | 데이터를 임시로 저장하기 위한 함수 | saveTempUser() |
| 올리기 | upload | 파일등을 서버로 올리는 함수 | uploadFile() |
| 내리기 | download | 파일등을 서버로 내리는 함수 | downloadFile() |

# SQL 작성규칙

## 명명규칙

* Mapper 클래스 단위로 하나의 xml 파일을 생성한다.
* Mapper xml 파일명은 Mapper 클래스와 동일하게 사용한다.

## Mapping XML ID Naming Rule

* Mapping XML 이란, SQL-Map XML 파일 내에서 사용하는 <select>, <insert>, <update>, <delete>, <parameterMap>, <resultMap>, <typeAlias> 등의 XML Tag 들을 뜻하며, “ID” 는 이들 구문을 구분하기 위한 unique 한 값을 뜻한다.
* 기본적으로 Mapping XML ID 는 Java 의 메서드 명명 규칙을 이용하여 작성하도록 한다. (<parameterMap>, <resultMap>)
* ID 는 “Package.ID”으로 작성 한다. Example: 업무명이 “주문” 이고, 장바구니기능 중 장바구니 담기에 대한 Mapping XML ID를 작성할 경우.

<insert id=“order.insertShoppingCart” …

## XML Tag: <parameterMap>

* <parameterMap>의 속성으로 정의하는 ID는 실제로 사용하게 될 클래스의 이름을 사용하고, 이름 뒤에 “Param” 이라는 접미사를 붙이도록 한다.  
  예)  
  <parameterMap id=”userVOParam” class=”com.oc.domain.userVO”>

## XML Tag: <resultMap>

* <resultMap>의 속성으로 정의하는 ID 는 실제로 사용하게 될 클래스의 이름을 사용하고, 이름 뒤에 “Result” 이라는 접미사를 붙이도록 한다.

예)<resultMap id=”userVOResult” class=”com.oc.domain.userVO”>

**End Of Document**