SARA 시스템

(공통, 인증관리, 예약관리, 출입관리)

개발 표준(안)

1. 개요 4

2. 개발환경 표준 4

2.1. 소프트웨어 아키텍처 4

2.1.1. Conceptual Design. 4

2.1.2. 기본 아키텍처 4

2.1.3. 적용 Framework. 5

2.1.4. 적용 SW표준 7

3. J2EE 표준 8

3.1. Naming Convention 8

3.1.1. 목적 8

3.1.2. 업무 Naming Rule 8

3.1.3. 프로젝트 Directory Structure Naming 9

3.1.4. 프로젝트 Program File Naming 11

3.1.5. Class Naming Rule 11

3.1.6. 프로젝트 Method Naming 11

3.2. UML 스테레오 타입 12

3.3. Java Code Convention 14

3.4. File Names 15

3.5. Class and Interface Declarations 15

3.6. File Organization 15

3.7. Package and Import Statement 15

3.8. Beginning Comment 16

3.9. Line Length 16

3.10. Wrapping Lines 16

3.11. Comment 17

3.12. Declarations 17

3.13. Statement 18

3.14. White Space 19

3.15. Miscellaneous Practices 20

3.16. JavaDoc에서 제공하는 Tags 20

3.17. OPTIONS 21

4. nexacro platform 구현표준 21

4.1. Service Package 21

4.2. Form 21

4.3. Component 변수명 22

4.4. Dataset Naming Rule 23

4.5. Comment 23

5. 메시지 표준 23

5.1. 메시지 분류 기준 23

5.2. 분류배경 및 특징 24

5.3. 메시지 처리 24

5.4. 로그 처리 24

6. DBMS 표준 25

6.1. Naming Convention 25

6.1.1. Table 25

6.1.2. XXX : 의미있는 명칭 약어 25

6.1.3. Column 25

6.1.4. DataType 26

6.2. PL/SQL Comment 26

6.2.1. DataType 26

**SARA 시스템 개발 표준(안)**

# **개요**

SARA 시스템의 개발표준을 제정함에 있어 기존의 개발표준을 최대한 반영하고, 미비한 부분은 보완하여 확정, 적용함으로써 Application 전환 개발 및 개발팀에서 작성된 산출물의 품질을 확보하고 가독성(Readibility)을 향상시키고자 한다.

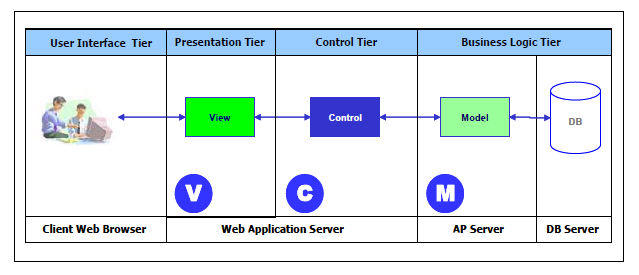
# **개발환경 표준**

## 소프트웨어 아키텍처

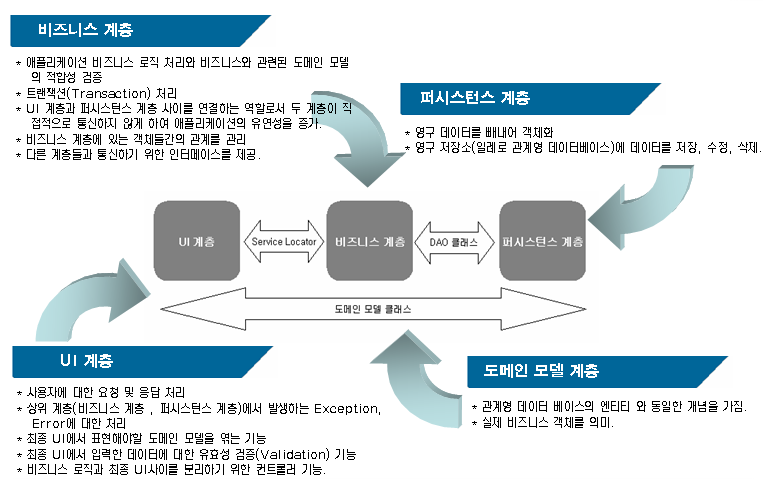
### Conceptual Design.

SARA 시스템의 High Level 관점에서 크게 3가지 Tier로 구성되며, 각각 User Interface Tier,

Presentation Tier (Control Tier), Business Logic Tier로 나누어 볼 수 있다.



### 기본 아키텍처



**UI 계층**

사용자에 대한 요청 및 응답 처리

상위 계층(비즈니스 계층 , 퍼시스턴스 계층)에서 발생하는 Exception, Error에 대한 처리

최종 UI에서 표현해야할 도메인 모델을 엮는 기능

최종 UI에서 입력한 데이터에 대한 유효성 검증(Validation) 기능

비즈니스 로직과 최종 UI사이를 분리하기 위한 컨트롤러 기능.

**비즈니스 계층**

애플리케이션 비즈니스 로직 처리와 비즈니스와 관련된 도메인 모델의 적합성 검증

트랜잭션(Transaction) 처리

UI 계층과 퍼시스턴스 계층 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 애플리케이션의 유연성을 증가시킨다.

비즈니스 계층에 있는 객체들간의 관계를 관리한다.

다른 계층들과 통신하기 위한 인터페이스를 제공한다.

**퍼시스턴스 계층**

영구 데이터를 빼내어 객체화 시킨다.

영구 저장소(일례로 관계형 데이터베이스)에 데이터를 저장, 수정, 삭제한다.

**도메인 모델 계층**

관계형 데이터베이스의 엔티티와 비슷한 개념을 가지는 것으로 실제 비즈니스 객체를 의미한다.

### 적용 Framework.

|  |
| --- |
| *Framework는 일종의 클래스들의 집합 혹은 협력으로서 정의될 수 있으며, 관계된 문제들의 집합을 해결하기 위한 추상화된 Design이다. Framework는 반드시 확장이 가능하거나 적용이 가능한 메커니즘과 서브시스템이 포함되어야 한다. Application Framework는 이러한 문제들을 해결하기 위한 추상화된 클래스와 패턴의 협력으로 나타낼 수 있으며 Application domain을 위한 generic software system이다. ( R. Johnson)* |

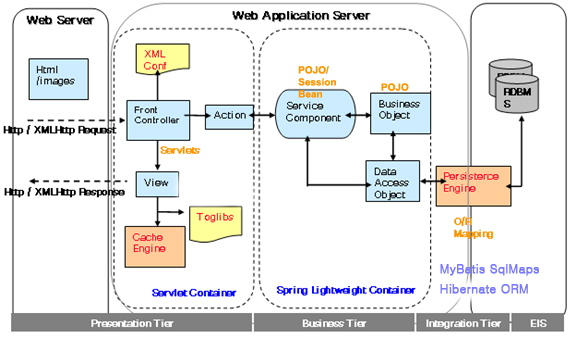
특정상황의 해법을 추상화하고, 해결방법보다는 문제 해결에 집중한다.

디자인 패턴 + 개발 기법 + 개발 가이드를 통해, 이슈를 정의 / 해결하는 목표를 설정, 달성하기 위한 일관성 있는 방법을 제시하기 위함이 목표이다.

Framework = Framework + Framework 의 사상으로 OpenSource + Framework 를 지향한다.

[적용 기술]

|  |  |  |
| --- | --- | --- |
| Spring Framework 3.2x | MyBATIS |  |
| AJAX | XML | AspectJ |
| Javascript | LOG4J |  |



**[ Presentation Tier ]**

**Tobesoft nexacro platform**

NEXACRO PLATFORM Runtime 버전은 기업의 스마트워크 실현을 위해 기능, 성능, 디자인 요소를 기업의 요구 수준에 맞게 조화를 이루도록 구성된 제품

**Spring Framework**

Spring framework는 Enterprise Application에서 필요로 하는 기능을 제공하는 오픈 소스 프레임워크이다. Spring framework는 평범한 자바객체(POJO : Plain Old Java Object)를 이용해서 단순하고, 테스트하기 쉬우며, 객체간의 결합이 느슨한 Enterprise Application 개발이 가능하도록 지원한다.

**[ Business Tier ]**

**TO (Transfer Object)**

네트워크 트랙을 줄이고, 적은 원격 호출로 많은 데이터를 전송하며, 코드 중복을 줄여준다.

**MTO(Multiple Transfer Object)**

하나의 테이블에서 모든 컬럼 정보를 들고 올 필요가 없을 경우, 컬럼들을 여러개의 TO로 나누어 데이터를 접근한다.

**AS (Application Service)**

비즈니스 티어의 컴포넌트와 서비스에 흩어져 있는 비즈니스 로직을 중앙 집중화 한다.

**DispatcherServlet**

사용자의 요청을 받는 단일 진입점의 역할을 수행한다.

**Annotation Based**

스프링 3.0에서 지원되는 @Annotation으로 Bean을 매핑시킴으로서 공통설정 파일에대한 접근으로 파생되는 갈등상황을 최소화 한다.

**MyBatis Framework**

java에서 DB를 편하게 핸들링할 수 있게 해주는 framework로서, SQL 실행 결과를 자바빈즈 혹은 Map 객체에 mapping해주는 퍼시스턴스 솔루션으로 SQL을 소스코드가 아닌 XML로 따로 분리해 관리하여 지겨운 JDBC코드로 부터 해방시켜준다. 또한 XML에서 동적으로 SQL 요소를 사용하여 쿼리문장을 프로그래밍 코딩없이 자유롭게 변환 할 수 있게 해준다. MyBatis를 통하여 개발 생산성을 극대화한다.

### 적용 SW표준

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **소프트웨어** | **제품명 및 버전** | **용도** |
| 응용소프트웨어 | WebServer | Apache HTTP Server | 웹 서버로써 Client 요청을 WAS 에 전송하며 이미지와 같은 정적인 자원에 대한 서비스 제공 |
| WAS | Tomcat 7.0 | 웹 어플리케이션 서버로써 서블릿 컨테이너를 제공하며, 비즈니스 호출 및 제어를 담당. |
| DBMS | Oracle 11g XE | 데이터의 무결성을 보장하며, 사용자의 요청에 의한 데이터 를 가공 및 원하는 데이터를 추출하는 엔진을 제공 |
| 개발도구 | DB Modeling 도구 | ER-Win | 논리적 모델 및 물리적 모델을 설계하는 도구로 활용 |
| IDE(통합개발환경) | Eclipse | JSP 및 어플리케이션 로직을 구현하는 도구로 활용 |
| UML Tool | StarUML | 유스케이스 분석단계부터 설계, 구현을 거쳐 테스트하는 도구로 활용 |
| 형상관리도구 | 문서파일관리 SVN | ToltoiseSVN | Project동기화 및 파일관리 활용 |
| 개발소스 SVN | SVN Team Provider | 버전관리 및 개발 이력 관리 |
| SVN Server | subversion | 개발저장소 관리 |
| Maven | Maven | 자동 빌드 및 lib 관리 활용 |

# **J2EE 표준**

## Naming Convention

### 목적

이 장에서는 “SARA 시스템” 개발을 위하여 적용되는 Naming Rule에 대한 표준을 제시한다.

Naming Rule 표준은 개발자와 Maintenance담당자간의 교량역할을 담당하고 설계, 구현, 유지 보수에 이르는 프로젝트 전반에 걸쳐 기준을 제시하기 위해 다음과 같은 목표를 가지고 설계하였다.

표준화의 인식 확대

개발의 용이성 제공

프로그램의 표준화 및 일관성 유지

프로그램의 품질 향상

운영, 유지, 보수의 편이성 제공

### 업무 Naming Rule

프로젝트 코드

프로젝트 코드는 “SARA System” 프로젝트를 나타내기 위한 가장 큰 개념의 구분이다

SARA는 “SARA System”을 나타내는 이름이다.

시스템 코드

본 프로젝트의 대 분류 성격의 업무를 구분하기 위해 사용한다.

업무 코드 분류

시스템공통: COM (Common System Management)

공통코드: BASE(Base Code Management)

회원: MEMBER(HUMAN RESOURCE SYSTEM)

예약 RES(RESERVATION)

### 프로젝트 Directory Structure Naming

SARA 시스템 프로젝트 Directory Structure Naming

(1) 프로젝트

|  |  |  |
| --- | --- | --- |
| **sara** | **src** | 프로젝트에서 발생되는 모든 소스의 루트. |
| **build** | 원본 build파일 Directory. |
| **WebContent** | Web Application Directory. |
| **target** | 서버배포용 Maven build 파일 Directory. |
| **pom.xml** | 자동빌드툴 설정을 위한 Project Object Model 파일 |
| **src** | **message** | 원본 Framwork 기본 메시지 설정 Java Source Directory.  \* 필수 사항이 아니므로 필요 시 생성 사용. |
| **test** | Test Module Java Source Directory  \* 필수 사항이 아니므로 필요 시 생성 사용. |
| **javadocs** | Javadocs 용 Source Directory  \* 필수사항이 아니므로 필요시 생성 사용 |

(2-1) 프로젝트 -> Java Source

|  |  |  |
| --- | --- | --- |
| **src/com** | **sara.common** | Framwork 공통 Java Source Directory. |
| **sara** | 각 업무에 관련된 application Java Source Directory.  *\* 실제 개발 디렉토리* |

(2-2) 프로젝트 -> Java Source -> 시스템 영역

|  |  |  |
| --- | --- | --- |
| **com.sara.common** | **advice** | aop 관련 Java Source Directory. |
| **~~annotation~~** | ~~DataSet 관련 Java Source Directory.~~ |
| **filter** | 공통 메소드 관련 Java Source Directory.  필요시 생성 사용. |
| **to** | BaseBean 관련 Java Source Directory. |
| **util** | Utilitily Base Java Source Directory  공통 Utilitily 메소드 Java Source Directory. |
| **exception** | 공통 exception 관련 Java Source Directory. |

(2-3) 프로젝트 -> Java Source -> 업무 프로그램 ( => 업무 서브시스템별 프로그램 ) 영역

|  |  |  |
| --- | --- | --- |
| **com.sara** | **com** | 서브시스템중 공통 시스템 관련 프로그램 Java Source Directory. |
| **acc** | 서브시스템중 회계 관련 프로그램 Java Source Directory. |
| **log** | 서브시스템중 물류 관련 프로그램 Java Source Directory. |
| **hrs** | 서브시스템중 인사 관련 프로그램 Java Source Directory. |
| **sup** | 서브시스템중 진료지원 관련 프로그램 Java Source Directory. |
| **med** | 서브시스템중 진료처방 관련 프로그램 Java Source Directory. |
| 서브시스템중 원무 관련 프로그램 Java Source Directory. |
| **won** |

\* 아래 공통 시스템 업무 프로그램 영역의 Source Directory 구조는 나머지 시스템공통, 물류 회계, 인사 프로그램 영역과 동일하다.

(2-4) 프로젝트 -> Java Source -> 공통시스템 업무 프로그램 영역 -> 최종 Source Directory

|  |  |  |
| --- | --- | --- |
| **com.xxx**  **(중분류약자**  **3~6자)** | **controller** | Controller Java Source Directory. |
| **dao** | Dao Java Source Directory. |
| **dao.mapper** | MyBatis SqlMap XML Source Directory. |
| **exception** | Exception Java Source Directory. |
| **service** | Service Java Source Directory. |
| **application Service** | Appliction Service Java Source Directory |
| **to** | Transfer Object Java Source Directory. |

(3-1) 프로젝트 -> WebContent

|  |  |  |
| --- | --- | --- |
| **WebContent** | **WEB-INF** | 최종 Application Output Directory |

(3-2) 프로젝트 -> WebContent-> WEB-INF

|  |  |  |  |
| --- | --- | --- | --- |
| **WEB-INF** | **config** | **context** | Framwork 설정 파일.  contextConfigLocation 과 servlet action 정보의 설정 File   * applicationContext-\*.xml |
| **servlet** | servlet-\*.xml |
| **reports** | IREPORT 관련 파일  업무코드+기능을설명하는단어.jrxml |
| **batis** | MyBatis 설정 파일   * mybatis-config.xml |
| **log4j** | Log4j 설정 파일   * log4j.xml |
| **property** | JDBC 설정파일   * jdbc.properties   resourceBundleView 설정파일   * resourceBundle.properties   사진 경로파일   * imageFileLocation.properties |
| **classes** | 서버 배포 컴파일 2진 파일 | |
| * web.xml * ~~weblogic.xml~~ | | |

### 프로젝트 Program File Naming

일반적으로 웹서비스에서는 단어와 단어 사이를 대소문자로 구분하는 방법인 카멜 표기법 (Camel Notation)과 단어의 첫 번째 글자를 대문자로 사용하는 파스칼 표기법을 사용하며,

모든 이름은 카멜 표기법 (Camel Notation)과 파스칼 표기법(PascalCasing Naming Convention)을 따르는 것을 원칙으로 한다.

1.모든 Java 파일은 첫글자가 대문자로 시작 (xml은 소문자) - 파스칼 표기법

2.복합 단어는 대문자로 구분 (e.g., "itemEdit.xml", “RegEmp.java) - 카멜표기법

3.밑줄("\_")이나 대시("-")는 파일이름에 사용하지 않음.

4.파일이름에 공백을 사용하지 않음

5.예외로 DAO(Data Access Object),AS(Application Service)만 대문자로 표기한다.

ex> empDAO.java , empAS.java.

### Class Naming Rule

|  |  |  |
| --- | --- | --- |
| 분류 | 대상 | 비고 |
| Business Service  (SOURCE) | Model Class, Controller Class Name,  Dao Class Name, Service Class Name | JAVA File에 대해서만 기술. |

(1) Business Service

|  |
| --- |
| **□ □ □ □ □ □ 클래스구분 . java**  기능을 의미하는 단어(~10) + postfix |

* postfix : 클래스 구분 (알파벳 대문자로 시작 )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Controller | DAO | DAOImpl | ServiceFacade | ServiceFacadeImpl | AS | ASImpl | Bean |
| Class | Interface | Class | Interface | Class | Interface | Class | Class |

### 프로젝트 Method Naming

Method는 동사로 시작되며, 첫 글자는 소문자로 시작한다.

“동사+명사”를 원칙으로 하고, 동사의 시작은 소문자로, 명사의 시작은 대문자로 한다

Prefix 동사형의 종류

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **Controller** | **Service/BD/AS** | **Dao** |
| 등록 | register~ | register~ | insert~ |
| 수정 | modify~ | modify~ | update~ |
| 삭제 | remove~ | remove~ | delete~ |
| 상세 조회 | find~Detail | find~Detail | select~Detail |
| 목록 조회 | find~List | find~List | select~List |
| 일괄처리 | batch~Process | batch~Process | batch~Process |

멤버변수(Attribute)

여러 단어가 조합될 경우 조합되는 단어의 첫 번째 문자는 대문자로 시작되어야 한다.

변수명은 짧으면서도 충분히 의미가 느껴지도록 한다.

**집합형**일 경우에는 List로 표현된다.

예) Customer[] mCustomerList = new Customer[MAX\_CUSTOMERLIST];

Method Parameter

생성자(constructor)나 set 메서드에서 멤버 필드에 값을 지정할 때, 파라미터 이름은 멤버 필드 이름과 동일하게 부여한다.

첫문자는 소문자로 시작한다.

Local Variables

라인당 하나의 변수만 선언한다.

로컬 변수마다 "//"로 설명을 달아준다.

로컬 변수는 하나의 목적으로만 사용한다.

Constant

클래스 상수 또는 일반상수로 정의된 변수명은 단어간에 밑줄 문자("\_")로 분리되는 대문자를 사용한다.

static final int MIN\_WIDTH = 4;

static final int MAX\_WIDTH = 999;

static final int PAGE\_LENGTH = 10;

**분석단계**

|  |  |
| --- | --- |
| **스트레오 타입** | **설명** |
| businessworker | 액터가 아닌 Business Domain에서 실제업무를 담담하는 업무담당자를 말한다. |
| usecaseRealization | 유즈케이스가 어떻게 실현 될 것인지 구체화 되는 모델표현이다. |
| boundary | 액터가 시스템에 접근하는 화면UI를 말한다. |
| control | 액터의 요청에 따라 요청을 해당하는 액션으로 구분하여 처리하는 액션이다. |
| entity | Model을 처리하는 영속성 계층의 개체이다. |
| bo | Model과 Control을 연계하여주는 수송자이다. |
| list | 조회를 하였을 때 조회 대상항목정보를 갖고 있다. |
| in | 사용자가 직접 입력하여야 하는 입력정보이다. |
| out | 시스템이 화면에 보여주는 출력정보이다. |
| auto\_in | 시스템이 자동으로 계산하거나 입력하여 주는 입력정보이다. |
| auto | 시스템이 자동으로 계산하여 출력하여 주는 출력정보이다. |
| select | 입력 정보 선택 시 보기에서 선택하는 동작을 나타낸다. |
| search | 검색조건으로 들어가는 입력정보이다. |
| hidden | 화면에 출력 되지 않는 숨겨진 정보이다. |

## UML 스트레오 타입

**설계단계**

|  |  |
| --- | --- |
| **스트레오 타입** | **설명** |
| **Server Programming** | |
| useCaseRealization |  |
| Controller |  |
| ServiceFacade |  |
| ApplicationService |  |
| DataAccessObject |  |
| xml |  |
| to |  |
| **UI(User Intefrace)** | |
| search | 검색할 조건을 속성으로 가지는 Collection |
| [i\_][o\_][io\_]edit | 개발자가 입출력하는 속성을 가진 Edit Component 또는 타입 |
| dialog | Modal Dialog를 띄워서 사용자가 부모 폼의 Component에 사용자가 선택한 값을 전달 |
| popup | 부모 Form과 해당 dialog사이의 **관계**에 사용 |
| button | 마우스의 Click을 통해 지정한 Script를 수행 |
| calendar | 날짜와 시간을 입력받기위한 Component |
| [i\_][o\_][io\_]checkBox | 선택된 상태/선택되지 않은 상태를 Switch 하면서 체크표시 |
| [i\_][o\_][io\_]combo | 여러 후보값중에서 1개의 값을 편리하게 선택하기위한 용도로 사용 |
| div | 하나의 화면에 여러개의 부분화면을 구성할 때 사용 |
| dataset | 데이타를 Table 형태로 저장하는 Component |
| binds | 데이터셋과 다른 컴포넌트간에 바인딩할 때 사용 |
| innerBinds | 바인딩된 데이터셋을 바인딩하는데 사용 |
| fileDownload | FileDownload 를 수행하는 Component |
| imageView | ImageViewer는 Image를 화면에 출력하기위해 사용하는 Component |
| **XP** | 화면을 구성하는 기본 단위로써 내부적으로 각 컴포넌트들을 배치하여 Business 화면을 구성 Form Component |
| grid | Dataset의 내용을 격자 모양으로 표현 |
| groupBox | border 영역에 title을 입력하여 그룹 형태를 나타냄 |
| [i\_][o\_][io\_]listBox | ListBox는 여러개의 후보값중에서 1개 또는 여러개의 값을 선택하기위해 사용 |
| maskEdit | 정해진 규격에 따라 입력될 필요가 있는 자료가 있을 때 사용 |
| menu | 개발자가 MenuBar를 구성하거나 PopupMenu를 구성 |
| radio | 동그란 모양의 여러가지 선택사항 중에서 하나를 선택할 수 있도록 하는 Button의 일종 |
| spin | 일정범위 내의 숫자값을 입력하기 위해 사용 |
| tab | 여러개의 Tab Page를 추가해서 보여주는 Component |
| tabpage | 각각의 Tab을 구성하는 TabPage |
| textArea | 여러줄로된 문자열을 입력받기위해 사용하는 Component |
| tree | 데이터셋의 내용을 레벨값을 이용하여 트리형태로 표현 |
| [i\_][o\_][io\_]cell | grid의 cell을 나타냄 |
| **기 타(etc)** | |
| Client | Front End 사용자를 말한다. |

## Java Code Convention

Java Code Conventions이 필요한 이유는 다음과 같다.

Maintenance의 편리성

소프트웨어 소스코드의 가독성(Readability) 향상

소스코드의 이식성(Portability) 및 재사용성(Reusability)에 도움을 준다.

개발자가 새로운 코드를 더욱 빠르고 완전하게 이해하는데 도움을 준다.

수정이 용이하고(Verifiability) 효율성(Efficiency)이 증대된다.

## File Names

File Suffixes

|  |  |
| --- | --- |
| File Type | **Suffix** |
| Java Source | .java |
| Java Bytecode | .class |
| Property File | .properties |
| JavaScript Source | .js |
| HTML Source | .html |
| Xml Source | .xml |
| CSS Source | .css |

Common File Name

|  |  |
| --- | --- |
| File Name | **Use** |
| Java Source file | Class/Interface 명과 동일하게 하여야 한다.  Class/Interface Naming Rule은 Class Naming Rule에 기술한 내용에 따른다. |
| Xml Source | Class Naming Rule에 기술한 내용에 따른다. |
| Property file | 카멜표기법을 따른다. |

## Class and Interface Declarations

각종 변수 선언은 public , protected , private 순으로 한다. 메소드 선언은 접근 권한에 의한 것보다 기능성 위주로 순서를 정하기로 한다. 이것은 코드를 이해하기 쉽게 하기 위해서다.

## File Organization

File Organization의 내용은 아래와 같다.

Package and Import statements

Beginning comments

Class and interface declarations

## Package and Import Statement

패키지를 선언하거나 import할 때의 문장이다.

package com.smarket.xxx

import javax.servlet.http.xxx;

## Beginning Comment

소스 파일의 처음부분에 기술하여야 한다.

|  |
| --- |
| /\*\*  \* EmpController  \*  \* @Description 사원관리 컨트롤러  \* @Author 홍길동  \* Created on 2015. 8. 11.  \*/ |

Package and Import Statement아래에 위치한다.

Indentation

Tab size를 4, Indent size를 4로 맞춘다.

## Line Length

가급적 한 줄에 80자 이상의 코딩은 하지 않는다.

JAVA Script와 HTML은 예외로 80컬럼 이상을 허용한다.

많은 툴과 터미널에서 Program Coding을 보기 쉽게 하기 위함이다.

## Wrapping Lines

한 줄에 80자 이상의 코딩이 넘어갈 때는 다음의 예시에 준하여 코딩한다.

메소드의 인자로 인하여 80컬럼의 범위를 벗어날 때

|  |
| --- |
| function(longExpression1, longExpression2, longExpression3,  longExpression4, longExpression5); |

메소드의 인자에 메소드가 선언되어서 80컬럼의 범위를 벗어날 때

|  |
| --- |
| var = function1(longExpression1,  function2(longExpression2,  longExpression3)); |

메소드에 관련된 키워드로 인하여80컬럼의 범위를 벗어날 때

|  |
| --- |
| private static synchronized horkingLongMethodName(int anArg,  Object anotherArg, String yetAnotherArg,  Object andStillAnother) {  ... } |

연산 문장이 80컬럼의 범위를 벗어날 때

|  |
| --- |
| longName1 = longName2 \* (longName3 + longName4 - longName5) +  4 \* longname6; |
| if ((condition1 && condition2) || (condition3 && condition4) ||  (condition5 && condition6)) {  doSomethingAboutIt(); } |

조건 비교 문장이 80컬럼의 범위를 벗어날 때

## Comment

Block Comment

comment 를 block으로 작성함

|  |
| --- |
| /\*  \* Here is a block comment with some very special  \* formatting that I want indent(1) to ignore.  \*  \* one  \* two  \* three  \*/ |

Single-Line Comments

한줄의 comment는 C++과 동일하게 ‘//’ 를 이용한다

|  |
| --- |
| if (condition) {  // Handle the condition.  ...  } |

Trailing Comment

문장 뒤에 오는 comment역시 C++과 동일하게 “//”를 사용한다.

|  |
| --- |
| if (a == 2) {  return TRUE; // special case  } else {  return isprime(a); // works only for odd a  } |

## Declarations

Number Per Line

주석처리를 하기 위하여 한 라인당 한번의 선언을 한다.

|  |
| --- |
| int level; // indentation level  int size; // size of table |

Class and Interface Declarations

메소드 이름과 “(“ 문자사이의 공백은 없게 한다.

메소드선언후 공백한칸을 띄운후 “{“ 가 오게 한다.

|  |
| --- |
| class Sample extends Object {  int ivar1;  int ivar2;  Sample(int i, int j) {  ivar1 = i;  ivar2 = j;  }  int emptyMethod() {}  ...  } |

## Statement

Simple Statements

한 라인에는 반드시 한 개의 statement만 허용한다.

|  |
| --- |
| String STATUS\_FILE\_DEV ="/home/weblogic/bea617/wlserver6.1/lib/config/paid\_service\_status"; |
|  |

if, if-else, if-else-if-else Statements

if 조건문

|  |
| --- |
| if (condition) {  statements; }    if (condition) statements; |

Note: {}를 사용함이 원칙, if 블록이 한 문장으로 끝나는 경우에도 {}를 생략하지 않는다.

if-else 조건문

|  |
| --- |
| if (condition) {  statements; } else {  statements; } |

**for Statements**

for statement의 initialization, condition, update는 한 라인에 표현을 하며 한 라인으로 기술하지 못 할 경우 라인 Skip의 Rule를 따른다.

|  |
| --- |
| for (initialization; condition; update) {  statements;  } |

**while Statements**

while statement 는 for statements와 동일한 Rule를 따른다.

|  |
| --- |
| while (condition) {  statements;  }  while (condition); |

단 “{}”안의 내용이 없을 경우는 “{}”를 생략 한다.

**do-while Statements**

|  |
| --- |
| do {  statements;  } while (condition); |

**switch Statements**

switch statement에서 case문과 statements는 반드시 라인을 바꾸어서 기술한다.

|  |
| --- |
| switch (condition) {  case ABC:  statements;  case DEF:  statements;  break;  default:  statements;  break;  } |

**try-catch Statements**

|  |
| --- |
| try {  statements;  } catch (ExceptionClass e) {  statements;  }finally {  statements;  } |

## White Space

Blank Spaces

키워드 다음에 괄호가 나오는 경우 키워드와 괄호 사이에 반드시 공백을 둔다.

|  |
| --- |
| while (true) {  ...  } |

메소드 이름과 그 다음에 오는 괄호사이에는 공백을 두지 않는다. 이것은 메소드 Call과 키워드를 구분하기가 쉽게 한다.

인자 리스트에서 콤마(,) 뒤에는 반드시 공백을 둔다.

연산자와 변수 사이에는 반드시 공백을 둔다.

|  |
| --- |
| a += c + d;  a = (a + b) / (c \* d);  while (d++ = s++) {  n++;  } |

**for 구문 예제**

|  |
| --- |
| for (expr1; expr2; expr3) |
| myMethod((byte) aNum, (Object) x);  myFunc((int) (cp + 5), ((int) (i + 3)) + 1); |

**Casting 예제**

## Miscellaneous Practices

Parentheses

|  |
| --- |
| if (a == b && c == d) // AVOID!  if ((a == b) && (c == d)) // RIGHT |

Returning Values

Return Values를 결정함에 있어서 보다 간결한 코드를 지향한다.

예제1)

|  |
| --- |
| if ( booleanExpression ) {  return TRUE; } else { => return booleanExpression;  return FALSE; } |

예제 2)

|  |
| --- |
| if (condition) {  return x; => return (condition ? x: y); } return y; |

## JavaDoc에서 제공하는 Tags

| Tag | **사용법** | **사용가능한 block** |
| --- | --- | --- |
| @author | @author *name*  ex) @author skcc | class, interface |
| @version | @version *version*  ex)version 1.33, 04/22/98 | class, interface |
| @param | @param *name description*  ex)@param s String | 메쏘드, 생성자 |
| @return | @return *description*  ex)return 참이면 true, 거짓이면 false | 메쏘드 |
| @exception | @exception *classname description*  ex)@exception NumberFormatException 스트링이 분석되지 못하는 integer를 가진 경우 | 메쏘드 |
| @see | @see *name*  ex)@see java.lang.Long#valueOf. | 모두사용 |
| @since | @since *sinec-text*  ex)since JDK1.0 | 모두사용 |
| @deprecated | @deprecated *deprecated-text*  *ex)@* deprecated As of JDK1.1, replaced by serBounds  @see #setBounds(int,int,int,int) | deprecation된 API에 @see tag와 함께 사용 |

javadoc에서 제공하는 tags를 통해서 formated된 문서를 생성한다. 각 tag는 @로 시작 (ex>@see)

| Option | **Description** |
| --- | --- |
| -public | public classes,members를 보여준다. |
| -protected | protected/public classes,members를 보여준다. (default) |
| -package | package/protected/public classes, members를 보여준다. |
| -private | 모든 classes,members를 보여준다. |
| -J *flag* | flag를 runtime system에 전달한다. |
| -encoding *name* | *name* : source file encoding name(ex> EUCJIS\SJIS) |
| -docencoding *name* | *name* : output HTML file encoding name |
| -version | @version tag를 포함한다.(default에서는 생략) |
| -author | @author tag를 포함한다. .(default에서는 생략) |
| -noindex | package index를 생략한다.(AllNames.html이 생성 않됨). |
| -d *directory* | *directory* : output file이 생성될 디렉토리 지정 |
| -sourcepath *path* | *path* : source file의 path  ex) |
| -classpath *path* | *path* : .class files의 path (환경변수<CLASSPATH>로 잡혀있어서 사용할 필요 없음) |
| -nodeprecated | @deprecated tag를 생략한다. |

## OPTIONS

# **nexacro platform 구현표준**

## Service Package

## “prefix ”으로 하여 패키지를 나눈다.

* Prefix업무코드 ( 알파벳 소문자로 시작 )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 공통 | 회계 | 물류 | 인사 | 진료지원 | 진료처방 | 원무 |
| COM | ACC | LOG | HRS | SUP | MED | WON |

## Form

Form ID는 Prefix + 유즈케이스명(소문자시작) + suffix로 통일한다.

Prefix는 유스케이스ID 접두영문 2글자(대문자)

Suffix는 Form이다.

Form의 Script는 개발표준지침의 여타 변수명 규칙보다는 JavaScript 고유의 융통성있는 변수명 규칙을 따르는 것이 훨씬 효율적이다.

## Component 변수명

|  |  |
| --- | --- |
| **컴포넌트별 접미사** | |
| **coomponent Name** | **접미어** |
| Div | Div |
| Button | Btn |
| Combo | Combo |
| Check Box | Check |
| List Box | List |
| Edit | Ed |
| Mask Edit | MaskEd |
| TextArea | Area |
| Menu | Menu |
| Tab | Tab |
| Tab Page | Tp |
| Radio | Radio |
| Static | Stc |
| Grid | Grid |
| Spin | Spin |
| Image Viewer | ImgView |
| Popup Menu | Pmenu |
| virtualFile | Vfile |
| fileDialog | Fdial |
| Calendar | Cal |

컴포넌트를 변수로 지정할 경우 ID속성에 suffix를 다음과 같이 정의하여 변수명에 접미한다

변수명은 사용자 정의 단어들의 \*조합으로 하며 변수 시작은 소문자 그리고 이어지는 단어들은 각각 시작을 대문자로 한다.(카멜표기법)

ex) 사원을 등록하는 버튼의 경우 Component ID: regEmp + Btn

## Dataset Naming Rule

|  |  |  |
| --- | --- | --- |
| 구 분 | DataSetList | DataSet (TO기준) |
| 명명규칙 | dl + 유즈케이스명  Ex) dlCbsReg | ds + 테이블명  Ex) dsAcctngChit |

prefix는 글로벌 데이터셋의 경우 gds을 붙이고 카멜표기법을 따른다.

컬럼명은 자바와 동일하게 카멜표기법을 따른다.

## Comment

* 스크립트별 작성정보 Comment

개발자는 다음의 양식처럼 스크립트별 정보를 주석으로 작성하여야한다.

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

\* \*

\* 종합검진 예약 \*

\* \*

\* @Path 종합검진관리 – 종합검진예약 \*

\* @Description 종합검진 접수에 앞서 예약을 시행. \*

\* @Author 홍길동 \*

\* \*

\* Created on 2015. 11. 11. \*

\* \*

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

Single Line Comment : 두개의 slash (//)를 주석 앞에 사용한다.

Multi Line Comment : /\*과 \*/ 사이에 주석을 기록한다.

# **메시지 표준**

## 메시지 분류 기준

메시지 단계(Message level)

애플리케이션 단계 (Application level)

최종 사용자가 볼 수 있는 메시지로 구성

presentation level

아키텍쳐 단계 (Architecture level)

프로그램 개발 시 발생할 수 있는 에러 메시지로 구성

common library에서 발생

Error handling 영역

DB Vendor Error 영역

Web Application Server 및 OS 영역

## 분류배경 및 특징

에러 메시지를 두 단계로 분리하여 프리젠테이션 단계(Application level)의 메시지는 사용자가 처리 가능한 메시지로 구성하고 아키텍쳐 단계에서 발생하는 (에러) 메시지는 별도의 에러 핸들링을 통해 대처해야 한다.

## 메시지 처리

메시지는 간단 명료하고, 어미는 ~하십시오. ~주십시오. ~없습니다. ~입니다. ~합니다. 등 경어체를 사용한다.

Bussiness Layer 및 Persistance Layer에서의 처리결과를 나타내주는 애플리케이션 단계의 메시지는 아래파일에 ‘메시지코드=메시지설명’의 형태로 등록하여 사용한다.

|  |
| --- |
| src/message/업무구분(3자리소문자)+Message.properties |

메시지코드는 다음과 같이 구성한다.

|  |  |
| --- | --- |
| **구 조** | 업무구분(3자리대문자) + 연번(3) |
| **규 칙** | 연번 : 001~999 사이의 숫자를 동일 명칭으로 하나씩 증가시키면서 부여 |
| **적용예** | WES001=조회된 자료가 없습니다.  WES002=자료가 저장되었습니다. |

## 로그 처리

로그의 처리는 프레임워크에서 제시한 log4j 를 사용하여 로그처리를 하도록 한다.

System.out.println()는 사용하지 않는다.

AOP를 사용해 적용한다.

log4j.properties 예시

|  |
| --- |
| # Root Logger  log4j.rootLogger=DEBUG, Console, File  #log4j.category.gov.paid=ERROR, Console, File  #log4j.category.gov.paid=WARN, Console, File  #log4j.category.gov.paid=INFO, Console, File  #log4j.category.gov.paid=DEBUG, Console, File  # Console  log4j.appender.Console=org.apache.log4j.ConsoleAppender  log4j.appender.Console.layout=org.apache.log4j.PatternLayout  log4j.appender.Console.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss}] %-5p %c{10}.%M(%3L) - %m%n  # File  log4j.appender.File=org.apache.log4j.DailyRollingFileAppender  log4j.appender.File.File=paid.log  log4j.appender.File.layout=org.apache.log4j.PatternLayout  log4j.appender.File.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss}] %-5p %c{10}.%M(%3L) - %m%n |

log수록 예시

|  |
| --- |
| package com.erp.test;  import java.util.TimeZone;  import java.util.Calendar;  import java.text.SimpleDateFormat;  import java.util.Date;  import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;  public class TimeZoneTest{  private static final Log LOG= LogFactory.getLog(TimeZoneTest.class);  public static void main(String args[]){  String[] str = TimeZone.getAvailableIDs();  for(int i=0 ; i<str.length ; i++){  **LOG.debug("debug test");**  TimeZone tz = TimeZone.getTimeZone(str[i]);  **LOG.warn("warning test");**  Calendar cal = Calendar.getInstance();  **LOG.error("error test");**  Date date = cal.getTime();  SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd - HH:mm:ss");  sdf.setTimeZone(tz);  **LOG.info(tz.getDisplayName() + "\t" + str[i] + " time :" + sdf.format(date));**  }  }  } |

# **DBMS 표준**

## Naming Convention

### Table

### XXX : 의미있는 명칭 약어

### Column

추가되는 필드에 한해서도 범용 스펙을 준수한다.

Column명 앞에 중분류를 위한 prefix를 붙이지 않는다.

Column명 앞에 테이블명을 prefix를 붙이지 않는다.

XXX\_XXX : 의미있는 명칭 약어

### DataType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column 종류** | **데이터타입** | **길이** | **예** | **비고** |
| **날짜** | VARCHAR2 | 10 | VARCHAR2(10) | “YYYY-MM-DD” 사용 |
| **개수, 수량, 금액** | NUMBER |  |  |  |
| **설명** | VARCHAR2 | 100 | VARCHAR2(100) |  |
| **코드타입** | VARCHAR2 | 코드길이 |  |  |
| **그 외** | VARCHAR2 | 10, 20, 30  (10단위를 원칙으로 함) | VARCHAR2(10)  VARCHAR2(20) |  |

## PL/SQL

## Comment

* Beginning Comment

개발자는 다음의 양식처럼 스크립트별 정보를 주석으로 작성하여야한다.

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

개체이름 : 개체명

작 성 일 : YYYY-MM-DD

작 성 자 : 작성자명

개 요 : 함수 및 프로시저 목적 및 개요

내 용 : 1. 함수 및 프로시저 내용

2. 함수 및 프로시저 내용

3. 함수 및 프로시저 내용

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

함수 및 프로시저 CREATE OR REPLACE Statement 아랫줄에 위 주석을 추가한다.

ex)

|  |
| --- |
| CREATE OR REPLACE PROCEDURE PC\_SD\_ACCEPT\_INORDER\_PATNO(  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  개체이름 : PC\_SD\_ACCEPT\_INORDER\_PATNO    작 성 일 : 2009-06-03  작 성 자 : 홍길동    개 요 : 환자의 특정 처방만 약처방을 접수한다.    내 용 : 1. 환자정보를 변경 및 추가.  2. 마스터정보를 변경 및 추가.  3. 환자의 특정 처방만 약처방 추가.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  i\_dosdate in varchar2, --투약일자  i\_keyin in varchar2, --환자번호  i\_procstat in varchar2, --처방구분 (N : 정규처방, A: 추가처방, E: 응급처방)  i\_editid in varchar2, --작업자  o\_logmsg out varchar2, --결과코드  o\_successyn out varchar2 --결과메세지  )  is ... |

Single Line Comment : 두개의 대시(-)를 주석 앞에 사용한다.

Multi Line Comment : /\*과 \*/ 사이에 주석을 기록한다.

## 오라클 Function & Procedure Naming Role

## FN\_업무구분2자\_함수명

## PC\_업무구분2자\_함수명