**개발 표준(Development Standard)**

* Author: 이상욱
* Created: 2014-04-28
* Modified: 2014-06-02
* File Name: DIG-BULD-02\_Development Standard\_R0.04\_140530\_HJZ.docx

# Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Reviser | Revised contents |
| 2014-04-28 | 0.01 | SU Lee | Initial revision |
| 2014-05-28 | 0.03 | SU Lee | Revised |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 Change History 2](#_Toc389471704)

[2 Naming Rule 6](#_Toc389471705)

[2.1 Project trigram 6](#_Toc389471706)

[2.2 Common naming rule 6](#_Toc389471707)

[2.3 ENOVIA Reserved Naming Convention 6](#_Toc389471708)

[2.3.1 File Naming Convention 6](#_Toc389471709)

[2.3.2 Naming Conventions for UI Administrative Objects 7](#_Toc389471710)

[2.4 ENOVIA Prohibited Naming Convention 7](#_Toc389471711)

[2.5 ENOVIA Administrative Object 7](#_Toc389471712)

[2.5.1 Data Schema Object 8](#_Toc389471713)

[2.5.1.1 Type 8](#_Toc389471714)

[2.5.1.2 Attribute 8](#_Toc389471715)

[2.5.1.3 Policy 8](#_Toc389471716)

[2.5.1.4 Relationship 8](#_Toc389471717)

[2.5.2 Dynamic UI Object 9](#_Toc389471718)

[2.5.2.1 Web Form 9](#_Toc389471719)

[2.5.2.2 Web Form Field Name 9](#_Toc389471720)

[2.5.2.3 Table 9](#_Toc389471721)

[2.5.2.4 Table Column Name 9](#_Toc389471722)

[2.5.2.5 Menu 10](#_Toc389471723)

[2.5.2.6 Command 10](#_Toc389471724)

[2.5.2.7 Portal 10](#_Toc389471725)

[2.5.2.8 Channel 11](#_Toc389471726)

[2.5.2.9 Inquiry 11](#_Toc389471727)

[2.5.2.10 Java Program Object (JPO) 11](#_Toc389471728)

[2.5.2.11 그 외 Administrative Objects 11](#_Toc389471729)

[2.5.2.12 Symbolic Name 11](#_Toc389471730)

[2.6 Administrative Business Object 12](#_Toc389471731)

[2.6.1 Number Generator Object 12](#_Toc389471732)

[2.6.2 Trigger Parameter Object 12](#_Toc389471733)

[2.7 Java Package Standard 13](#_Toc389471734)

[2.8 JSP 13](#_Toc389471735)

[2.8.1 기존 JSP 13](#_Toc389471736)

[2.8.2 신규JSP 13](#_Toc389471737)

[2.9 기타 14](#_Toc389471738)

[2.9.1 Config.xml 14](#_Toc389471739)

[3 Coding Convention 14](#_Toc389471740)

[3.1 General Coding Convention 14](#_Toc389471741)

[3.2 주석 15](#_Toc389471742)

[3.2.1 파일 주석 15](#_Toc389471743)

[3.2.2 Class주석 15](#_Toc389471744)

[3.2.3 변수 주석 15](#_Toc389471745)

[3.2.4 method 주석 16](#_Toc389471746)

[3.2.5 그 외 주석 16](#_Toc389471747)

[3.2.6 OOTB 코드 수정 표시 주석 16](#_Toc389471748)

[4 ENOVIA 확장 / 변경 18](#_Toc389471749)

[4.1 Administrative Object 18](#_Toc389471750)

[4.1.1 Data Schema Administrative object 18](#_Toc389471751)

[4.1.2 Dynamic UI Object 18](#_Toc389471752)

[4.1.3 Web form Field & Table Column 18](#_Toc389471753)

[4.2 JSP 파일 18](#_Toc389471754)

[4.3 JPO 19](#_Toc389471755)

[4.3.1 Base class와 Derived class 19](#_Toc389471756)

[4.3.2 JPO method overriding 19](#_Toc389471757)

[4.4 Java Beans 20](#_Toc389471758)

[4.4.1 기존 Beans로부터 파생 20](#_Toc389471759)

[4.4.2 파일과 newInstance method 맵핑 (AEF) 20](#_Toc389471760)

[4.4.3 검색 경로 21](#_Toc389471761)

[4.4.4 맵핑 파일 수정 21](#_Toc389471762)

[4.5 Javascript 파일 22](#_Toc389471763)

[4.6 Property 파일 23](#_Toc389471764)

[4.7 String Resource 파일 23](#_Toc389471765)

[4.7.1 Data Schema Object 24](#_Toc389471766)

[4.7.2 Label 및 Message 24](#_Toc389471767)

[4.8 Config.xml 26](#_Toc389471768)

[4.9 기타 26](#_Toc389471769)

[4.9.1 3rd Part Library 사용 26](#_Toc389471770)

[4.9.2 Html 태그 26](#_Toc389471771)

[4.10 Customization 유형 Summary 27](#_Toc389471772)

[4.11 성능 최적화 27](#_Toc389471773)

[4.11.1 Vault 또는 OID로 Business Objects 또는 Relationships 접근 27](#_Toc389471774)

[4.11.2 Relationship 과 Business Object 데이터 선택 27](#_Toc389471775)

[4.11.3 여러 Object Type의 쿼리 28](#_Toc389471776)

[4.11.4 잘못된 쿼리 사용 28](#_Toc389471777)

[4.11.5 중복된 Business Object 이름의 검사 29](#_Toc389471778)

[4.12 트랜잭션 처리 30](#_Toc389471779)

[5 개발 환경 구성 30](#_Toc389471780)

[5.1 개발 서버 구성 30](#_Toc389471781)

[5.2 개발자 개발 환경 구성 31](#_Toc389471782)

[5.3 개발 소스 관리 32](#_Toc389471783)

[6 Deploy 33](#_Toc389471784)

[6.1 서버 코드 배포 34](#_Toc389471785)

[6.2 Client 코드 배포 34](#_Toc389471786)

[7 Appendix 35](#_Toc389471787)

[7.1 자바 코딩 방법 35](#_Toc389471788)

[7.1.1 JSP 35](#_Toc389471789)

[7.1.1.1 “Include” 대 “JSP:Include” 35](#_Toc389471790)

[7.1.1.2 “JSP:Forward” 대 “Redirect” 35](#_Toc389471791)

[7.1.1.3 페이지 세션 35](#_Toc389471792)

[7.1.1.4 JSP 파일 컴파일 35](#_Toc389471793)

[7.1.2 Java 36](#_Toc389471794)

[7.1.2.1 “List” 와 “Vector” 의 예 36](#_Toc389471795)

[7.1.2.2 “Pattern” 객체 사용 36](#_Toc389471796)

[7.1.2.3 “Hashtable” 객체 사용 37](#_Toc389471797)

[7.1.2.4 “StringList” 와 “SelectList” 사용 37](#_Toc389471798)

[7.1.2.5 Business Object의 TNR(Type, Name, Revision) 가져오기 37](#_Toc389471799)

[7.1.2.6 공백 문자열 비교 38](#_Toc389471800)

[7.1.2.7 String (java.lang.String) 38](#_Toc389471801)

[7.1.2.8 반복문 39](#_Toc389471802)

[7.1.2.9 조건 비교 39](#_Toc389471803)

[7.1.2.10 StringTokenizer (java.util.StringTokenizer) 39](#_Toc389471804)

[7.2 JPO 코드 41](#_Toc389471805)

[7.2.1 JPO 이름 매크로 41](#_Toc389471806)

[7.2.1.1 class 선언 41](#_Toc389471807)

[7.2.1.2 class 참조 41](#_Toc389471808)

[7.2.2 필수 method 42](#_Toc389471809)

[7.2.2.1 생성자 42](#_Toc389471810)

[7.2.2.2 Main Entry Point (mxMain method) 42](#_Toc389471811)

[7.2.3 JPO 호출 43](#_Toc389471812)

[7.2.3.1 외부에서 JPO method 호출 43](#_Toc389471813)

[7.2.3.2 내부에서 JPO method 호출 43](#_Toc389471814)

[7.2.3.3 ADK 접근 43](#_Toc389471815)

[7.2.3.4 JPO method에서 자바 개체를 매개 변수로 전달 44](#_Toc389471816)

[7.2.4 주석 44](#_Toc389471817)

[7.2.5 Table을 위한 JPO 프로그램 45](#_Toc389471818)

[7.2.5.1 JPO for Column Value 45](#_Toc389471819)

[7.2.5.2 JPO for Column Update 46](#_Toc389471820)

[7.2.5.3 JPO for Column Sort 47](#_Toc389471821)

[7.2.5.3.1 ENOVIA Framework의 정렬 JPO 프로그램 48](#_Toc389471822)

[7.2.6 Web Form을 위한 JPO 프로그램 48](#_Toc389471823)

[7.2.6.1 JPO for Field Value 48](#_Toc389471824)

[7.2.6.2 JPO for Field Range Value 49](#_Toc389471825)

[7.2.6.3 JPO for Field Update 50](#_Toc389471826)

[7.3 응용 프로그램 보안 점검 체크 리스트 52](#_Toc389471827)

# Naming Rule

이 장은 ENOVIA 기반의 프로젝트 수행에 있어서 사용되는 프로그램 및 Data Schema (Administrative Object)에 대한 Naming 표준을 기술함으로써 개발 소스의 품질 및 가독성을 높이고 향후 유지보수를 용이하게 함을 목적으로 한다. 모든 Java 프로그램(j, Java Bean, Servlet 등), CAA 프로그램 및 ENOVIA 관련 파일 (properties, javascript, xml, xsl, css 등)들은 본 문서의 Naming Rule을 준수하여야 한다.

## Project trigram

ENOVIA기반으로 개발할 때 시스템 전체에서 사용할 프로젝트 약어를 정의하여 사용한다.

본 프로젝트에서 사용되는 프로젝트 약어는 아래와 같으며 대문자를 사용하도록 한다.

|  |  |
| --- | --- |
| **Full Name** | **Abbreviation** |
| Doosan Infracore Global PDM | DIG |

## Common naming rule

다음은 ENOVIA에서 지원하는 Naming Rule에 대한 기본적인 내용으로 만약 Administrative Object에 별도의 Naming Rule이 정의되어 있지 않으면 아래 기술한 내용에 기반하여 Naming을 하도록 한다.

1. 127자 이내에서 영문으로 해당 Administrative Object의 의미를 전달할 수 있도록 정의한다.
2. 대/소문자를 구분하여 정의한다.
3. 여러 단어로 오브젝트 이름을 구성할 경우 이름 중간에 공백(Space) 문자를 사용하지 않으며 단어 구분을 위하여 단어의 첫 글자는 대문자를 사용한다.
4. ENOVIA에서 사용하는 예약어 및 XML / MQL에서 사용되는 Keyword는 사용하지 않는다.
5. 특수 문자는 사용하지 않는다.
6. 새로운 오브젝트를 만드는 경우 프로젝트 약어를 Prefix 로 사용하도록 한다.

## ENOVIA Reserved Naming Convention

아래에 제공된 예약어는 신규 프로그램 및 ENOVIA Administrative Object를 정의할 때 Prefix로 사용할 경우 문제를 발생시킬 수 있으므로 사용하지 않는다.

### File Naming Convention

|  |  |
| --- | --- |
| **Prefix** | **Description** |
| eService  emx  common  emxcommon  eServicecommon | Prefix for jsp, jpo, tcl.   * + eServiceInstallAdmin.tcl   + emxTeamAddBusinessUnit.jsp   + emxcommonPushPopShadowAgent ( JPO ) |
| banner  bar  button  datapopulate  logo  menu  populate  softmgmt  splash  UserInterface  Util | Other reserved prefix for file naming convention |

### Naming Conventions for UI Administrative Objects

|  |  |
| --- | --- |
| **Prefix** | **Descriptions** |
| AEF | Application Exchange Framework |
| APP | Common Components |
| CFC | Configuration Central |
| DMC | Document Management Central |
| ENC | Engineering Central |
| LIB | Library Central |
| PRC | Product Central |
| PMC | Program Management Central |
| SCS | Sourcing Central |
| SPC | Specification Central |
| SUP | Supplier Central |
| TMC | Team Central |

## ENOVIA Prohibited Naming Convention

ENOVIA Object 및 프로그램 이름을 정의 할 때 아래 표시된 문자는 사용을 금지하며, 이외에도 특수 문자 사용은 피하도록 한다.

|  |
| --- |
| **" # $ @ % , \* ? \ < > [ ] |** |

또한, Underscore(“\_”)는 ENOVIA에서 Symbolic Name을 구성하는 중요 인자로 사용되므로 Data Schema의 이름에는 사용하지 않는다.

단, Web Form 및 Menu 등에서 해당 Administrative Object의 Symbolic Name을 이용하여 정의되는 Object는 예외로 한다.

## ENOVIA Administrative Object

이 절에서는 ENOVIA Administrative Object을 정의하는데 필요한 Naming Rule을 정의하며 그 대상은 아래와 같다.

|  |  |
| --- | --- |
| **구분** | **종류** |
| Data Schema Object | Type, Attribute, Policy, Relationship, Interface |
| Dynamic UI Object | Web Form, Table, Portal, Channel, Menu, Command |
| Program Object | Inquiry, Program, Page |

### Data Schema Object

Data Schema Object의 Name을 정의할 때, 약어를 사용하기 보다는 직관적으로 의미를 알 수 있는 용어를 사용하여 이름을 정의하도록 한다.

#### Type

1. Abstract Type의 경우 이름은 모두 대문자로 정의한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Type Name> | DIGPart, DIGRawMaterial  DOCUMENTS (Abstract Type) |

#### Attribute

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Attribute Name> | DIGWeight, DIGUnitOfMeasure |

#### Policy

1. 제어하는 Type의 이름과 동일하게 정의한다.
2. 만약 동일 Type에 여러 개의 Policy가 필요한 경우 Policy명 뒤에 의미를 전달할 수 있는 단어를 조합하여 사용한다.
3. 만약 하나의 Policy가 여러 개의 Type을 제어하는 경우 범용 의미로 사용 될 수 있는 용어를 Policy 명으로 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Governed Type Name> | DIGECO |
| <Project Abbreviation><Governed Type Name><Meaning> | DIGECOForOption |

#### Relationship

1. Relationship의 경우 From/To Type 명의 조합으로 이름을 정의한다.
2. 동일 From/To Type간에 여러 Relationship이 존재할 경우 From/To Type 명 조합 뒤에 직관적으로 사용자에게 의미를 전달할 수 있는 단어를 기술한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><From Type Name>To<To Type Name> | DIGPartToDocument |
| <Project Abbreviation><From Type Name><Meaning><ToTypeName> | DIGPartReferenceDocument, |

### Dynamic UI Object

#### Web Form

1. Type의 Symbolic Name을 기반으로 프로젝트Prefix와 Web Form의 용도에 해당하는Suffix를 함께 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Symbolic Name of Type>\_<Purpose of Web Form> | DIGtype\_Part\_CreateForm DIGtype\_Part\_EditForm  DIGtype\_Part\_SearchForm |

|  |  |
| --- | --- |
| **Suffix** | **Description** |
| CreateForm | 생성 화면 |
| EditForm | 조회/수정 화면 |
| SearchForm | 검색 화면 |

1. 수정과 조회 화면을 위한 web form은 동일한 것을 사용하는 것을 원칙으로 한다.

#### Web Form Field Name

1. Web Form Field의 경우 해당 Web Form안에서 Unique하게 정의되어야 하며, Field 목적을 전달할 수 있는 용어를 사용한다.
2. 특수 Field (Section Header/Separator, Table Holder) 처럼 의미가 있는 이름을 정의하기 어려운 경우 아래와 같이 Field Type의 약어에 Sequence Number를 부여한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Field Type> + 2Digit Number | SectionSeparator01, SectionSeparator02 |

#### Table

1. Indented Table로 사용되는 경우 “Indented”를 Suffix로 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaning>[<Indented>] | DIGEBOMSummary  DIGEBOMSummaryIndented |

#### Table Column Name

1. Table Column의 경우 해당 Table안에서 Unique하게 정의되어야 하며, Column 목적을 전달할 수 있는 용어를 사용한다.
2. 특수 Column (Separator, Group Header) 처럼 유의미한 이름을 정의하기 어려운 경우 아래와 같이 Column Type에 Sequence를 부여 한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Column Type> + 2Digit Number | Separator01,Separator02 |

#### Menu

1. 기본적인 Naming 규칙과 함께Menu의 용도에 따라 해당하는 Suffix를 함께 사용한다.
2. Category Menu (Tree Menu)의 경우 반드시 해당 Type의 Symbolic Name을 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaning><Suffix> | DIGPartMyDesk  DIGPartSearchMenu  DIGSpecToolbar  DIGEBOMActionsToolBar  DIGEBOMToolBar |
| <Symbolic Name of Type> | type\_Part, type\_ProductLine |

|  |  |
| --- | --- |
| **Suffix** | **Description** |
| MyDesk | ENOVIA의 My Desk Menu에 포함되는 Menu |
| Actions | ENOVIA의 Actions에 포함되는 Menu |
| SearchMenu | ENOVIA의 Global Search에 포함되는 Menu |
| ToolBar | ENOVIA의 Page Action Toolbar로 사용되는 Top-Level Menu |
| ActionsToolBar |  |

#### Command

1. 기본적인 Naming 규칙과 함께 Command의 용도에 해당하는Suffix를 함께 사용한다.
2. Sequence Number를 붙이는 경우 2 Digit Number를 사용하고 Sequence Number 앞에 “\_”를 사용한다

|  |  |
| --- | --- |
| **ormat** | **Example** |
| <Project Abbreviation><Meaning><Suffix>[\_<2 Digit Number>] | DIGPartRelatedSpecLink  DIGRemoveSelectedDocumentCmd |

|  |  |
| --- | --- |
| **Suffix** | **Description** |
| Cmd | Toolbar의 Actions 메뉴를 구성하는데 사용하는 경우 |
| Link | Category Menu (Tree Menu), Channel, MyDesk Menu를 구성하는 데 사용하는 경우 |

#### Portal

1. 기본적인 Naming 규칙과 함께 Suffix로 “Portal”을 함께 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaning>Portal | DIGProjectIssuePortal  DIGEBOMPortal |

#### Channel

1. 기본적인 Naming 규칙과 함께 Suffix로 “Channel”을 사용한다.
2. Sequence Number를 붙이는 경우 Sequence Number 앞에 “\_”를 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaing>Channel[\_<2 Digit Number>] | DIGProjectIssueChannel\_01  DIGProjectIssueChannel\_02 |

#### Inquiry

1. 기본적인 Naming 규칙과 함께 Suffix로 “Inquiry”을 사용한다.
2. Sequence Number를 붙이는 경우 Sequence Number 앞에 “\_”를 사용한다

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaning>Inquiry[\_<2 Digit Number>] | DIGReferenceDocumentInquiry  DIGEBOMSummaryInquiry\_01 |

#### Java Program Object (JPO)

신규 JPO의 경우 프로젝트 약어를 Prefix로 사용한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Project Abbreviation><Meaning> | DIGECUtil  DIGPartUtil |

### 그 외 Administrative Objects

그 외 아래 기술된 Administrative Object들은 기본 Naming 규칙외에 특별한 제한을 두지 않는다.

* Format
* Association
* Role
* Process
* Group

### Symbolic Name

1. Administrative Object Type Name과 Administrative Object Name의 조합으로 구성하며Underscore (”\_”)로 구분한다.
2. Administrative Object Name에 공백(Space)이 포함된 경우 Symbolic Name에서는 이를 제거한다.
3. Administrative Object Name에 부득이 하게 Underscore가 사용된 경우 Symbolic Name에서는 이를 제거 한다.
4. Data Schema에 해당하는 Administrative Object는 반드시 Symbolic Name을 등록하고 program에서 이를 사용하도록 한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Admin Type Name><Underscore><Administrative Object Name> | type\_Part  attribute\_UnitOfMeasure |

## Administrative Business Object

### Number Generator Object

1. Object Name 은 자동 채번 기능을 사용하려는Type의 Symbolic Name을 사용한다.
2. Revision은 동일 Type에 여러 형태의 채번 규칙이 필요한 경우에 사용한다.

|  |  |
| --- | --- |
| **Type** | **Example** |
| Type | eService Object Generator  eService Number Generator |
| Object | type\_Part  type\_DOCUMENTS |
| Revision | 1, 2, 3…  A, B, C…  a, b, c… |

### Trigger Parameter Object

1. Object Name 은 Trigger가 발생하는 Administrative Object Name (Policy의 경우에는 State Name 포함), Trigger Event 그리고 Trigger Type의 조합으로 정의한다. 단, 공통으로 사용되는 Trigger를 정의하는 경우 기능을 표현할 수 있는 용어와 Trigger Event 및 Trigger Type의 조합을 사용할 수 있다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <AdminObjectName>[<StateName>]<TriggerEvent><Trigger Type> | PartDeleteAction  ECPartWorkingPromoteCheck |
| <Meaning>[<StateName>]<TriggerEvent><Trigger Type> | TypeAllCreateAction |

|  |  |
| --- | --- |
| **Naming Rule** | **Descriptions** |
| Administrative Object Name | Trigger가 발생되는 Administrative Object Name  e.g.) Part, ECPart, DOCUMENTS… |
| State Name | Policy의 경우 Trigger 발생 되는 State Name ( only for Policy )  e.g.) Preliminary, Review |
| Trigger Event | Trigger Event 명  e.g.) Promote, Create, Delete, Connect, ModifyAttribute |
| Type of Trigger | Trigger Type 명  e.g.) Check, Override, Action |

1. Revision은 특별한 규칙을 두지 않고 해당 Trigger를 표현할 수 있는 용어를 사용한다.

Ex) CheckPartDuplicate, RouteToReviewer

## Java Package Standard

Java class를 생성해야 하는 경우, 아래와 같은 package 구조를 가지도록 한다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Format** | **Example** |
| BEAN | com.di.<TrackName>.<DomainName>.bean | com.di.pdm.product.bean |
| UTIL | com.di.<TrackName>.<DomainName>.util | com.di.pdm.product.util |
| Servlet | com.di.<TrackName>.<DomainName>.servlet | com.di.pdm.product.servlet |
| Taglib | com.di.<TrackName>.<DomainName>.taglib | com.di.pdm.product.taglib |

|  |  |
| --- | --- |
| **Track Name** | **Descriptions** |
| pdm | GPDM |
| cat | CATIA |
| dm | DELMIA |

|  |  |
| --- | --- |
| **Module Name** | **Descriptions** |
| product | Product Configuration |
| cad | CAD/DMU |
| bom | BOM |
| common | Common |
| document | Document |
| Interface | Interface |
| migration | Migration |

## JSP

### 기존 JSP

기존 OOTB 파일을 수정해야 하는 경우, web.xml 파일의 “CustomFilter”에 정의된 CustomFilter Prefix를 붙여 수정하도록 한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Custom Filter Prefix><OOTB JSP File Name> | DIG\_emxpartCreatePart.jsp  DIG\_emxProductCentralProductUtil.jsp |

### 신규JSP

신규 JSP 파일은 프로젝트 약어를 prefix로 사용하고 도메인 약어와 JSP파일의 용도 등을 조합하여 Naming을 하도록 한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| < Project Abbreviation><Domain Name><JSP Name><Suffix>.jsp | DIGCadCreateNewPartDialogFS.jsp  DIGCadCreateNewPartDialog.jsp  DIGCadCreateNewPartProcess.jsp |

|  |  |
| --- | --- |
| **Domain Abbreviation** | **Domain Full Name** |
| Prd | Product Configuration |
| Cad | CAD/DMU |
| Bom | BOM |
| Com | Common |
| Doc | Document |
| Int | Interface |
| Mig | Migration |

|  |  |
| --- | --- |
| **Suffix** | **Descriptions** |
| FS | Frameset JSP Page |
| Process | Business Logic 처리용 JSP |
| intermediate | 실행되는 JSP를 구성하기 위한 중간 처리용 JSP |

## 기타

### Config.xml

검색 조건 field 정의를 위해 사용되는config.xml 파일에 추가되는 ELEMENT의 이름은 프로젝트 약어와 함께 underscore(“\_”)를 prefix로 사용하도록 한다.

|  |
| --- |
| <BOTYPEFIELDS name="DIG\_FIELDS" application="DIG">  <FIELD name="DIG\_FIELDNAME" ... />  </BOTYPTFIELDS> |

# Coding Convention

두산 인프라코어 GPDM프로젝트에 적용되는 모든 소스 코드(Java, JSP, javascript, ENOVIA JPO, ENOVIA Inquiry 등)는 아래에 정의되는 Coding convention을 준수해야 한다.

## General Coding Convention

1. Tab 키 대신 들여쓰기 4칸을 한다. (IDE 툴에서 Tab을 들여쓰기 4칸으로 설정한다)
2. 모든 주석은 가급적 영문으로 기술하는 것을 권장하며, 한글을 사용할 수 있다. 단. 한글 주석 사용을 위하여 소스 파일의 content type은 UTF-8를 사용해야 한다.
3. Deprecated된 method, class, interface를 사용하지 않는다.
4. JPO method에서 JPO.invoke() method를 사용하여 다른 JPO의 method 호출하지 말고 일반 class 처럼 JPO class를 Instantiate 한 후 method를 호출한다.
5. Data Schema 관련 상수는 Symbolic Name 정의를 이용하고 FrameworkProperties. getSchemaProperty(context, symbolic\_name) method를 사용하여 실제 상수 값을 가져온다.
6. JPO 또는 class에 가급적 public 및 static method를 정의하여 사용하지 않는다.

## 주석

새롭게 생성되는 모든 jsp, jpo, javascript의 class, method에는 그 class/method의 내용을 기술하는 주석이 주어져야 한다. 이 주석을 이용하여 유사한 class 또는 method를 식별할 수 있어야 한다.

### 파일 주석

새로운 Java, JSP, Javascript 파일을 설명하기 위해 파일명, 파일 정보, 수정 내역 등의 설명을 기술한다.

파일 주석의 예는 다음과 같다.

|  |
| --- |
| /\*------------------------------------------------------------------------------  \* Name : TestClass.class  \* DESC : File description  \* VER. : 1.0  \* AUTHOR : Gil-Dong, Hong  \* PROJ : KHR PLM  \*  \* Copyright 2007 by ENOVIA All rights reserved.  \*------------------------------------------------------------------------------  \* Revision history  \*------------------------------------------------------------------------------  \* DATE Author Description  \* ---------- --------------------- ---------------------------  \* 2007/02/09 Gil-Dong, Hong First Creation  ------------------------------------------------------------------------------\*/ |

### Class주석

class 에 대한 주석은 java doc 생성을 위한 표준을 따르며, 아래 예제와 같은 최소한 정보는 기술하도록 한다

<Example>

|  |
| --- |
| /\*\*  \*<PRE>  \* *Description*  \* </PRE>  \* @author Gil-Dong, Hong  \* @version 1.0 2007/07/30  \* @see MX\_FILE\_SAMPLE  \*/ |

### 변수 주석

Java 변수에 대한 주석은 아래 예제와 같이 표시한다.

<Example>

|  |
| --- |
| protected String bookID = null; // Document number  protected String title = null; // Document name  protected String author = null; // Author  protected String publisher = null; // Publisher |

### method 주석

Java method에 대한 주석은 java doc생성을 위한 표준을 따르며 아래 예제와 같이 기술한다. 만약 상속받은 class의 method를 override 할 경우, @see를 사용하여 override된 method 정보를 반드시 기술한다.

<Example>

|  |
| --- |
| /\*\*  \* <PRE>  \* Descriptions : *Describe ……*  \* </PRE>  \* @param int param1 – XXXXX  \* @param String param2 – YYYYY  \* @return boolean openSuccess – *Describe ……*  \*  \* @throws SQLException – *Describe ……*  *\* @see java.math.XXXX.yyyy*  \*/ |

### 그 외 주석

|  |  |
| --- | --- |
| **Case** | **Example** |
| 한 줄 주석인 경우 | // *Describe …….* |
| 여러 줄 주석인 경우 | /\*  *Describe …….*  *Describe …….*  \*/ |

### OOTB 코드 수정 표시 주석

수정되는 OOTB 코드 (java, javascript, jsp, css 파일)는 파일 유형별로 주석을 위한 문자를 사용하도록 하고 수정 시작 부분과 끝 부분을 아래와 예제와 같이 반드시 표시한다

<Example – java, jsp, javascript >.

|  |
| --- |
| //[B] – By Author on YYYY/MM/DD  // OOTB CODE #1  CUSTOM CODE #1  …  /\*  OOTB CODE #2  OOTB CODE #3  \*/  CUSTOM CODE #3  …  //[E] – By Author on YYYY/MM/DD |

<Example – xml, xsl >.

|  |
| --- |
| <!-- [B] – By Author on YYYY/MM/DD -->  <!-- OOTB CODE #1 -->  CUSTOM CODE #1  …  <!--  OOTB CODE #2  OOTB CODE #3  -->  CUSTOM CODE #3  …  <!-- [E] – By Author on YYYY/MM/DD --> |

# ENOVIA 확장 / 변경

## Administrative Object

모든 Administrative Object의 신규 생성 및 수정은 반드시 MQL스크립트를 작성하여 DB에 반영하는 것을 원칙으로 한다. 편의상 Business Modeler에서 직접 수정하는 경우라도 수정된 결과를 MQL 스크립트로 작성하도록 한다. 즉, 수정/생성에 사용된 MQL 스크립트는 Administrative Object를 오류 없이 만들 수 있도록 완전한 스크립트로 구성되어야 한다.

그리고, 신규 추가되는 Administrative Object 의 MQL스크립트를 만들 때 system property로 다음의 내용을 반드시 추가한다.

property = application value ‘Framework’

property = installer value ‘<Project Abbreviation>’

property = original name value ’<Administrative Object>’

property = version value ‘R214x’

### Data Schema Administrative object

기존 기능을 확장하는 경우, Type 및 Relationship은 상속을 받아서 사용을 하고 Policy의 경우는 직접 수정을 하도록 한다.

신규 기능의 경우, Type, Relationship 그리고 Policy 모두를 새롭게 정의한다. 새롭게 정의하는 Data Schema Administrative Object의 Naming Rule은 2 절에 있는 Naming Rule을 준수하도록 한다.

### Dynamic UI Object

ENOVIA에서 OOTB로 제공되는 Dynamic UI Object의 내용을 수정해야 할 경우 원본에 “\_OOTB” 라는 Suffix를 붙여 복사해 둔 후 수정을 한다.

|  |  |
| --- | --- |
| **Format** | **Example** |
| <Original Name>\_OOTB | Type\_Part\_OOTB |

### Web form Field & Table Column

OOTB Web Form의 Field 또는 Table의 Column을 삭제해야 하는 경우, 해당Field 또는 Column을 삭제하지 말고 “Access Expression” setting을 이용해 화면에서 보이지 않도록 처리한다.

## JSP 파일

OOTB로 제공되는 JSP 파일을 수정해야 할 경우 web.xml 파일에 정의된 Custom Filter Prefix를 활용하여 OOTB JSP파일을 복사하고 복사한 JSP 파일을 수정하도록 한다. 예외적으로 JSP 파일에 “<include ../>” 구문으로 include되는 파일들은 Filter를 적용할 수 없으므로 직접 해당 파일을 수정하도록 한다. 이 경우는 3절에 정의한 주석 처리 방법을 참고하여 수정하는 부분에 반드시 주석을 추가하도록 한다.

OOTB JSP를 수정하는 경우가 아닌 신규로 JSP 파일을 만들어야 하는 경우, 데이터 입/출력을 위한 Business Logic은 JPO 에 구현하고 JSP파일에서는 해당 JPO method를 호출하도록 구현한다.

## JPO

대부분의 OOTB Business logic은 JPO로 구현되어 있고, 이를 수정하기 위해서는 java bean가 동일하게 파생 JPO에 method를 overriding하여 변경된 logic을 구현한다.

그리고, 일반적인 Business logic구현은 JPO를 통해서 구현하도록 한다. (JSP 또는 java bean의 사용을 지양한다.)

### Base class와 Derived class

대부분 JPO들은 Base class와 파생한 class의 조합으로 구성된다. 예를 들면, Part 관련 Business Logic을 담고 있는 JPO는 Base class가 “emxPartBase”이고 파생 class가 “emxPart” 이다.

OOTB Business Logic은 기본적으로 JPO로 구현된다. OOTB에서 제공하는 Business Logic을 변경하는 경우는 향후 버전 업그레이드를 위해 파생된 JPO에 변경된 Logic을 구현한다.

method를 상속받기 위하여 시스템(Bean, 트리거, 동적 UI 컴포넌트)에 의해 만들어진 JPO method들의 모든 참조는 파생된 JPO에 있어야 한다. 만약 파생한 JPO에 존재하지 않는다면, Base JPO class method가 호출된다.

### JPO method overriding

JPO method를 상속받는 기본 방법은 Base JPO에 존재하는 method의 method argument와 정확히 일치하는 method를 파생된 JPO에 기술하는 것이다. 만약 OOTB JPO에 Custom Logic을 추가하는 경우라면 파생된 method는 “super”를 사용하여 Base JPO의 method들을 참조할 수 있다. 만약 파생된 method가 상위 method와 완전히 틀릴 경우, 파생된 method를 완전히 바꾸면 된다.

**EXAMPLES**



## Java Beans

OOTB로 제공되는 Java Beans의 Logic의 수정이 필요하거나 새로운 method가 필요하면, 새로운 Java Bean을 생성하고 AEF class로부터 상속을 받아 새로운 class를 만들어 사용한다.

### 기존 Beans로부터 파생

Bean을 상속한 새로운 class를 작성한다 (즉, jar파일 중 하나의 OOTB Bean으로부터 상속받아 만든다.). Business Type 또는 Relationship Type을 사용하기 위한 Bean을 찾을 수 있도록 맵핑 파일을 업데이트한다.

**EXAMPLES**



### 파일과 newInstance method 맵핑 (AEF)

각 jar 파일은 주어진 OID, 주어진 Type에 따라 Bean instance를 반환하는 newInstance method에 의하여 사용되는 맵핑 파일을 가지고 있다.

맵핑 파일 이름은 “emx” 접두어와 “MappingFile.properties” 접미어를 사용합니다. 두 단어 사이에 “Common”이나 어플리케이션 명이 들어간다. 예를 들어, engineering.jar 파일은 “emxEngineeringMappingFile.properties”이름의 맵핑 파일을 찾는다. common.jar 파일은 “emxCommonMappingFile.properties” 이름의 맵핑 파일을 찾는다.

파생된 Bean class를 정의한 후 newInstance() method에서 OOTB 대신에 정의한 class의 instance가 사용되도록 지정하기 위해 알맞은 맵핑 파일의 내용을 변경하라. 예를 들어, 만약 engineering.jar의 Part Bean을 확장하여 com.abc.apps.engineering.abcPart 라는 새로운 class를 생성하면

type\_Part=com.matrixone.apps.engineering.Part

에서

type\_Part=com.abc.apps.engineering.abcPart

로 맵핑 파일의 내용을 변경한다.

단일 JSP 파일이나 요청 범위(Request Scope)이 필요한 경우, “useBean” 태그보다 newInstance() method를 사용하는 것을 추천한다.

|  |
| --- |
| Part partObj = (Part) DomainObject.newInstance(context, objectId, “engineering“); |

### 검색 경로

DomainObject.newInstance() method에서 적합한 Bean을 찾는 방법은 주어진 Type이나 OID를 기본으로 한다.

1. 만약 어플리케이션이 주어지면, 맵핑 파일을 찾는다. 적합한 파일을 찾으면, 스키마에서 찾은 Type에 가장 근접한 것을 반환하는데 사용한다.
2. 만약 common 맵핑 파일에서 찾지 못하면, DomainObject를 반환한다.

### 맵핑 파일 수정

emxSystem.properties 설정 파일은 처음으로 찾는 외부 맵핑 파일들을 나열하는데 사용된다. 만약 맵핑 파일을 처음으로 찾길 원할 경우 목록의 처음에 있어야 한다. 아래 예제는 항상 customMappingFile.properties이 처음으로 검색된다.

|  |
| --- |
| emxFramework.BeanMapping.PreliminarySearchFiles =  customMappingFile.properties,emxIntegrationsBeanMappingFile.properties |

예를 들어 아래와 같은 맵핑 파일들이 존재할 경우,

|  |
| --- |
| customMappingFile.properties 🡪 type\_Part= com.matrixone.apps.custom.CustomPart  emxIntegrationsBeanMappingFile.properties 🡪 type\_Part= com.matrixone.apps.integrations.IntegrationsPart  emxEngineeringMappingFile.properties 🡪 type\_Part= com.matrixone.apps.engineering.Part  emxCommonMappingFile.properties 🡪 type\_Part= com.matrixone.apps.common.Part |

newInstance() method를 사용하여 Part 타입을 호출할 경우,

항상 com.matrixone.apps.custom.CustomPart를 반환한다.

심지어 세번째 아규먼트에 “engineering”을 주어도 com.matrixone.apps.custom.CustomPart가 반환된다.

## Javascript 파일

Javascript는 JPO 나 JSP 처럼 OOTB 코드를 유지할 수 있는 방법이 제공되지 않는다. 하지만 한 html 페이지에 동일 이름의 function이 여러 개 정의하였을 경우 마지막에 로드된 function만 유효하게 인식하는 javascript 언어의 특성을 이용하여 기존 function을 overriding (?) 한다.

* 기존 function의 수정을 하거나, 새로운 function을 추가해야 하는 경우 해당 javascript파일의 맨 아래 부분에 아래 예제와 같은 주석을 추가하고 수정된 function 또는 신규 function을 주석 사이에 추가한다.

<Example>

|  |
| --- |
| //--------------------------------------------------------------------------------------------------------------  //  // [B] – CUSTOM CODE FOR DOOSAN INFORCORE GPDM PROJECT  //  //--------------------------------------------------------------------------------------------------------------  …  **수정 또는 추가되는 내용을 입력합니다**.  …  //--------------------------------------------------------------------------------------------------------------  //  // [E] – CUSTOM FOR DOOSAN INFORCORE GPDM PROJECT  //  //-------------------------------------------------------------------------------------------------------------- |

* 특정 브라우저에서만 지원하는 script 언어(Ex, VBScript 또는 Jscript)는 사용하지 않는다.
* Webform Validation에 사용되는 javascript function의 수정 및 추가는 다음의 방법을 사용한다.
* Common Custom JS 파일 정의: emxSystem.properties의 “***eServiceSuiteFramework.UIForm.ValidationFile***” 에 Custom js 또는 jsp 파일을 정의하고 정의한 파일에 필요한 javascript function을 정의한다.
* Application Specific Custom JS 파일 정의: emxSystem.properties에서 각각의 Application 별로 정의되는 “***eServiceSuite<Application-Name>UIForm.ValidationFile***”에 Custom js 또는 jsp 파일을 정의하고 정의한 파일에 필요한 javascript function을 정의한다.

<Example – emxSystem.properties 파일 내용>

|  |
| --- |
| …  eServiceSuiteFramework.UIForm.ValidationFile= scripts/emxUIFormValidation.js,\  scripts/CustomUIFromValidation.js  …  …  eServiceSuiteEngineeringCentral.UIForm.ValidationFile= emxEngrValidation.jsp, \ CustomAppsUIFormValidation.js  … |

* Indented Table Validation에 사용되는 javascript function의 수정 및 추가는 다음의 방법을 사용한다.

emxSystem.properties에 각각의 Application별로 정의되는 “***eServiceSuite<Application-Name >.UIFreezePane.ValidationFile***” 파일에 Custom js 또는 jsp 파일을 정의하고 정의한 파일에 필요한 javascript function을 정의한다.

<Example – emxSystem.properties 파일 내용>

|  |
| --- |
| …  eServiceSuiteEngineeringCentral.UIFreezePane.ValidationFile = emxEngineeringCentralFormValidation.jsp, \ CustomValidation.jsp  … |

## Property 파일

ENOVIA는 시스템 일부 기능을 property설정을 통해서 변경할 수 있다. emxSystem.properties 파일에 각 모듈(central)에서 사용하는 property 파일을 정의하는 것이 원칙이다.

Ex)

eServiceSuiteEngineeringCentral.ApplicationPropertyFile=emxEngineeringCentral.Properties

eServiceSuiteEngineeringCentral.PropertyFileAlias=emxEngineeringCentral.Properties

각 모듈(central)에서 사용하는 property는 아래와 같이 3가지 방법으로 수정할 수 있다.

1. WEB-INF\classes 폴더에 있는 central별 property를 직접 수정
2. emxSystem.properity 파일 내 emxFramework.CustomProperties.Filename 에 정의된 파일 (Ex, emxFrameworkCustom.properties)을 이용한 수정
3. Page object를 이용한 수정

위의 (3) 번은 R2014x 버전부터는 지원되는 방식이며 이 방식은 WAS의 재 구동 없이 변경 내용을 반영할 수 있다.

**본 프로젝트에서는 위의 (3) 방식을 이용해서 property를 수정하거나 추가하는 것을 원칙으로 한다.** Page object를 활용하는 자세한 방법은 On-line 문서를 참고하도록 한다.

## String Resource 파일

OOTB로 제공되는 Label 또는 Message 변경은 OOTB string resource 파일의 맨 아래 부분에 아래 예제와 같이 주석을 추가하고 수정하려는 string resource key를 주석 사이에 복사하여 수정하도록 한다. 새로 추가하는 resource key도 수정하는 경우와 동일한 방법으로 추가한다.

본 프로젝트에서는 기본 화면 표시 언어는 영어로 한다. (Wintop / Webtop 동일함)

<Example>

|  |
| --- |
| //--------------------------------------------------------------------------------------------------------------  //  // [B] – CUSTOM CODE FOR DOOSAN INFORCORE GPDM PROJECT  //  //--------------------------------------------------------------------------------------------------------------  …  **수정 또는 추가되는 내용을 입력합니다**.  …  //--------------------------------------------------------------------------------------------------------------  //  // [E] – CUSTOM FOR DOOSAN INFORCORE GPDM PROJECT  //  //-------------------------------------------------------------------------------------------------------------- |

### Data Schema Object

ENOVIA는 Data Schema Object (Type, Attribute, Relationship, Policy 및 Policy State)의 이름을 화면에 표시할 때 브라우저의 언어 설정에 기반하여 다국어로 표시하는 기능을 지원한다. 화면에 표시되는 이름을 변경하거나, 새롭게 추가된 Data Schema Object의 이름을 Alias name으로 표시하려고 하면 OOTB에서 제공되는 emxFrameworkStringResource.properties 파일의 내용을 수정한다.

|  |  |
| --- | --- |
| **유형** | **형태** |
| Type | emxFramework.Type.<Type Name> |
| Attribute | emxFramework.Attribute.<Attribute Name> |
| Policy | emxFramework.Policy.<policy Name> |
| State | emxFramework.State.<Policy Name>.<State Name> |
| Relationship | emxFramework.Relationship.<Relationship Name> |
| Format | emxFramework.Format.<format Name> |
| Store | emxFramework.Store.<Store Name> |

단, Attribute의 Lange 값을 화면에 표시하는 경우에 한하여 한글 사용을 허용한다. 이 경우 emxFrameworkStringResrouce\_ko.properties 파일을 만들고 필요한 StringResource 키를 정의한다.

### Label 및 Message

ENOVIA는 화면에 표시되는 Label 및 Message에 대해 브라우저의 언어 설정에 기반하여 다국어로 표시하는 기능을 지원한다. 다국어 지원을 위해서는 emxSystem.properties 파일에 각 모듈(central)에서 사용하는 string resource 파일을 정의하는 것이 원칙이다.

Ex) Engineering Central에서 사용하는 String Resource 파일 정의

eServiceSuiteEngineeringCentral.StringResourceFileId = emxEngineeringCentralStringResource

* string resource 파일에 화면에 표시되는 text를 언어별로 정의함.

emxEngineeringCentralStringResource.properties : 영어

새로 추가되는 Label과 Message를 위한 resource key는 다음과 같은 format으로 정의하도록 한다.

|  |  |
| --- | --- |
| **유형** | **형태** |
| Webform Field Label | digFramework.Field.Label.<resource key> |
| Table Column Header | digFramework.Column.Label.<resource key> |
| Alert Message | digFramework.Alert.<resource key> |
| Warning Message | digFramework.Worning.<resource key> |
| Error Message | digFramework.Error.<resource key> |

## Config.xml

검색 조건 및 Indexed 검색을 위한 Index field 정의를 하는 config.xml 파일의 수정은 OOTB에서 제공하는 config.xml 파일에 본 프로젝트를 위한 Application을 정의하고, 별도의 config\_<project abbreivation>.xml 파일에 추가로 검색에 사용할 Type 및 Field등을 도메인 별로 구분하여 정의하도록 한다.

<Example – config.xml 파일 내용>

|  |
| --- |
| <INDEXER>  <CHUNK size="10000"/>  <THREADPOOL size="4" />  <DATABASE name="AdvancedSearchDB0" />  <PROVIDER name="XL"/>  <SERVER host="<INDEXSERVER IP>" baseport="19000"/>  ...  ...  <APPLICATION name="DIG" active="true"/>  ...  <!— [B] – Include the custom index field definition -->  <INCLUDE xmlsrc="config\_dig.xml" />  <!—[E] – Include the custom index field definition -->  ..  ..  </INDEXED-VAULTS>  </INDEXER> |

<Example – config\_dig.xml 파일 내용>

|  |
| --- |
| <BOTYPE name="DIG\_BOTYPE" includes="basic,DIG\_BOFIELD" indexfile="false" active="true" />  <BOTYPEFIELDS name="DIG\_BOFIELD" application="DIG">  <FIELD name=”ATTRIBUTE\_NAME” />  ...  </BOTYPEFIELDS> |

## 기타

### 3rd Part Library 사용

Dassualt Systems에서 제공하지 않는 Open Source Library를 포함한 모든 3rd Party Library(Ex, jquery javascript library, POI java library)의 사용은 원칙적으로 금지한다. 만약 기능 구현을 위해 꼭 써야 하는 경우에는 Domain Leader, Technical Leader에게 해당 Library 또는 Tool의 사용 당위성을 설명한 후 사용 여부에 대한 인가를 받은 후 개발에 사용을 하도록 한다.

### Html 태그

Html 표준을 준수하도록 한다. 개발은 Fireforx 기반으로 하고, 테스트는 IE10으로 하도록 한다. 특정 브라우저만 지원하는 html tag는 사용하지 않는다.

## Customization 유형 Summary

아래는 Customization 대상(유형)별로 Customization 여부를 구분할 수 있는 요소에 대한 설명이다.

|  |  |  |  |
| --- | --- | --- | --- |
| **유형** | **명칭 구분** | **Content구분** | **비고** |
| Data Schema | O |  |  |
| Dynamic UI Schema | O |  |  |
| JPO | O | O |  |
| java 파일 | O |  |  |
| jsp 파일 | O | O |  |
| CSS 파일 |  | O |  |
| xls, xml 파일 |  | O |  |
| javascript 파일 | O | O |  |
| String Resource |  | O |  |
| Properties | O |  |  |
|  |  |  |  |

## 성능 최적화

### Vault 또는 OID로 Business Objects 또는 Relationships 접근

Business Object를 검색할 때, 가능한 Vault를 검색 조건에 사용하여야 한다. Vault를 사용하지 않을 경우, ENOVIA Core는 데이터베이스에 있는 모든 Vault를 대상으로 SQL을 실행시키기 때문에 시스템의 성능에 나쁜 영향을 미친다.

MQL Command를 사용할 때 Object의 ID를 알고 있다면, 그 Object ID를 사용하는 것이 Object의 **T**ype, **N**ame, **R**evision(TNR)을 사용하는 것보다 좋다.

print bus Part A100 1 select attribute[Quantity];

보다

print bus Part A100 1 in ‘eService Production’ select attribute[Quantity];

보다

print bus ***OID*** select attribute[Quantity];

를 사용하는 것이 좋다.

### Relationship 과 Business Object 데이터 선택

Object에서 정보를 가져올 때 한번에 하나씩 가져오는 것 보다는 SelectList를 사용하여 여러 개의 정보를 동시에 가져오는 것이 좋다.

**BAD**:

*DomainObject dObj = DomainObject.newInstance(context, objectId);*

*String strOne = dObj.getInfo(context,"attribute[One]");*

*String strTwo = dObj.getInfo(context,"attribute[Two]");*

**BETTER**:

*DomainObject dObj = DomainObject.newInstance(context, objectId);*

*StringList strlist = new StringList(2);*

*strlist.add("attribute[One]");*

*strlist.add("attribute[Two]");*

*Map map = dObj.getInfo(context, strlist);*

특정 Object와 연관된 Object(Related Object)의 정보를 가져오기 위하여 MQL Selectable Item을 사용하여야 할 때, 만약 Relation 방향(From 혹은 To)을 알고 있다면, “relationship” Item 보다는 “from” 혹은 “to” Item을 사용한다.(검색 대상 데이터베이스 테이블의의 수를 줄일 수 있기 때문)

*Instead of …* relationship.[BOM].attribute[Qty]

*Use …* from.[BOM].attribute[Qty]

예를 들면,

print bus Part A100 1 select relationship[BOM].attribute[Qty];

보다

print bus Part A100 1 select ***from***[BOM].attribute[Qty].***value;***

을 사용한다.

그리고, relationship으로 연결되어 있는 Object의 정보를 가져올 경우, “print bus” 보다는 “expand bus” MQL문을 사용하는 것이 보다 빠르다.

*Instead of …* print bus OID select from.name from.businessobject.type from.businessobject.name from.businessobject.revision;

*Use …* expand bus OID from;

Object에 연관된 Object의 Type, Name, Revision을 가져오는 MQL문이지만, “expand bus” MQL 문이 보다 효율적이다.

### 여러 Object Type의 쿼리

여러 Object Type에서 값을 가져올 경우, where 절에 각 Type별로 질의를 만드는 것 보다, comma 구분자를 사용하여 Type Field에 넣는다.

**BAD:**

*temp query bus \* 900 \* vault V1 where " type == 'Query Test 1' || type == 'Query Test 2' ";*

**BETTER:**

*temp query bus "Query Test 1,Query Test 2" 900 \* vault V1;*

### 잘못된 쿼리 사용

Business Object의 TNR을 알고 있는 경우, “temp query bus” 보다 “print bus” MQL문을 사용하는 것이 좋다.

**BAD:**

*temp query bus “type” “name” “revision” select current dump ;*

**BETTER:**

*Print bus “type” “name” “revision” select current dump ;*

### 중복된 Business Object 이름의 검사

기본적으로 ENOVIA는 시스템에 새로운 Business Object를 추가하기 전에 모든 Vault에서 Business Object의 Type, Name, Revision(TNR)의 값이 중복되어 있는지 검사한다. ini 파일의 설정을 바꿈으로써 바꿀 수 있다.

MX\_CHECK\_DUPLICATE\_NAMES=false

설정을 바꾸더라도, TRN의 중복을 검사는 유효하다.

만약 다음 중 하나가 사실이면, 모든 Vault에서 검사를 행하는 것에 대한 성능 부하에 의미를 두지 않는다.

* 새로운 Business Object 생성은 사용자 정의 코드나 유일함이 확보된 이름이 사용된다.
* 사용자 데이터는 Vault에 의하여 구성되고 만약 중복된 TNR이 존재하더라고, 사용자는 Vault가 다르기 때문에 서로 다르다고 이해한다.

## 트랜잭션 처리

try-catch문의 시작에 start 트랜잭션, 마지막에 commit 트랜잭션을 기술한다. 이것은 트랜잭션 동안에 어떠한 에러를 잡아낼 수 있다. 에러가 발생하였을 경우, catch문에서 abort 트랜잭션을 가장 먼저 기술한 후 사용자 통지 및 error 메시지 출력을 기술한다.

|  |
| --- |
| Boolean bTransactionStarted = false  try {  ContextUtil.startTransaction(context, true);  bTransactionStarted = true;  ...  businessobject = new BusinessObject(s7);  businessobject.open(context);  ……  ContextUtil.commitTransaction(context);  } catch (Exception ex) {  if(bTransactionStarted)  ContextUtil.abortTransaction(context);  ...  } finally {  businessobject.close(context);  } |
|  |

# 개발 환경 구성

## 개발 서버 구성

개발 환경은 구성하는 H/W는 DB 서버, AP 서버, License 서버 그리고 SVN 서버로 구성되며, AP 서버에는 3D Experience Platform을 비롯하여 WebLogic, FTS(Full Text Search) 서버인 Exalead, FCS(File Collaboration Server), 3D Index Server가 설치되어 있다. 개발 소스의 형상 관리를 위한 툴은 별도의 Workstation 장비에 구성했으며 구성 및 기본 정보는 아래 그림과 같다.



주요 서버의 IP 및 사용되는 Port 정보는 다음과 같다.

|  |  |
| --- | --- |
| **서버** | **IP (Port)** |
| DSLS | 10.231.108.139 (4085) |
| DB | 10.231.108.72 (1521) |
| WAS | 10.231.108.73 (7080) |
| FCS | 10.231.108.73 (7085) |
| SVN | 10.38.49.80 (3690) |

개발 서버에 설치된 S/W의 설치 버전은 아래와 같다.

|  |  |
| --- | --- |
| **항목** | **설치 버전** |
| OS | AIX 7.1 TL1 SP2 |
| DSLS | R2014x HF1418 |
| 3D Experience Platform | R2014x HF1418 |
| DB | Oracle 11gR2 (11.2.0.1.0) Enterprise |
| WAS | WebLogic 12c (12.1.1) |
| JAVA | Java 7 SR4FP2 64bit |

## 개발자 개발 환경 구성

개발자들은 개인별 개발 장비 (노트북, W/S)에 개발 툴과 WAS를 설치하고, 개발 DB Server에 접속하여 개발을 진행한다.

개발 장비에 설치해야 할 S/W 및 버전은 아래와 같다.

자세한 설치 방법 및 환경 구성은 “DIG-BUIL-xx\_개발환경설정 가이드\_CIL\_R1.00\_140423.pptx”를 참조하도록 한다.

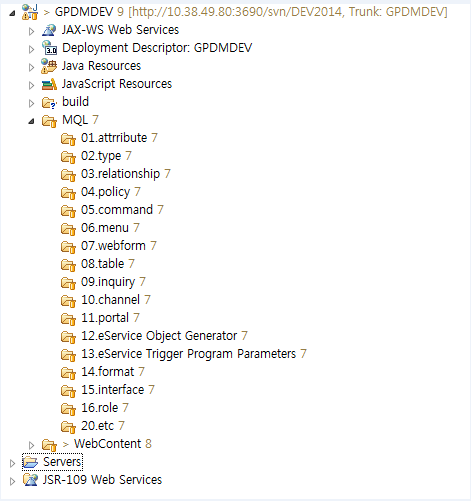
|  |  |
| --- | --- |
| **항목** | **설치 버전** |
| 3D Experience Platform  - ENOIVALiveCollaborationServer | R2014x HF1418 |
| 3D Experience Platform  - ENOVIAStudioModelingPlatformRichClient | R2014x HF1418 |
| OS | Windows 7 Enterprise Edition 64bit |
| DB | Oracle Client 11gR2 (11.2.0.1.0) |
| WAS | TomEE+ 1.6.0 |
| JDK | Java 7 Update 45 64bit |
| JRE | JRE 7 update 45 32bit 주2) |
| Eclipse | Kepler Release (4.3.2) |
| Web browser | FireFox 24 ESR, Internet Explorer 10 |
| CATIA V6 + ENOVIA 주1) | R2014x HF1418 |

주1) CATIA V6 와 ENOVIA는 필요한 개발자만 설치하도록 한다.

주2) 웹 브라우저의 버전에 따라 32bit 또는 64bit 설치가 필요하다.

## 개발 소스 관리

ENOVIA에서 OOTB로 제공하는 코드 (JSP, JPO 및 Tcl) 및 개발 프로그램 소스의 형상 관리를 위해 SVN 툴을 사용한다. SVN 툴을 통해 관리하는 소스 및 SVN Repository 구조는 아래 그림과 같다.



|  |  |
| --- | --- |
| **폴더 명** | **내용** |
| MQL | ENOVIA Data Schema 및 Business Administrative Object를 생성/수정할 때 사용한 MQL script를 Administrative Object Type별로 별도 폴더를 구성하여 관리함.  Ex) MQL script 파일 |
| WebContent | ENOVIA web application deploy 폴더 전체를 관리하기 위한 폴더  Ex) jar, jsp, javascript, xml, css, xsl, xslt, gif, jpg 등. |
| Java Resources | JPO, java bean 소스 코드를 관리하기 위한 폴더  Ex) java 파일 |

# Deploy

개발된 코드는 서버 코드와 클라이언트 코드로 구분하여 배포 방식을 정의한다.

## 서버 코드 배포

서버 코드는 아래와 같은 용도별 서버에 특정 시점에 배포하도록 한다. 서버 코드 배포를 위한 툴은 HUDSON과 SVN 및 ANT 툴을 사용한다.

|  |  |  |
| --- | --- | --- |
| **용도** | **용도** | **배포 시점 및 주기** |
| 개발서버 | 개발코드의 검증을 위한 서버 | 기본은 2주 단위. 그 외 필요한 시점. |
| 테스트서버 | 단위 테스트, 통합 테스트를 서버 | 테스트 기간 전. / 테스트 기간 동안은 필요한 시점. |
| Pilot 서버 | PILOT 시스템 테스트를 위한 서버 | PILOT 시스템 오픈 전 / 매주 금요일 오후 |
| 운영서버 | 운영 환경을 위한 서버 | 운영 시스템 오픈 전 / 별도 계획을 따름 |

## Client 코드 배포

Client 코드는 공용 파일 서버를 통해 빌드된 binary 코드와 관련 resource 파일을 배포하도록 한다. 자세한 방법은 아래를 참고한다. (TO-BE UPDATE)

# Appendix

## 자바 코딩 방법

### JSP

JSP 파일의 성능 향상을 위한 구체적인 코딩 팁을 제시한다.

#### “Include” 대 “JSP:Include”

가능하면, <jsp:include page=”filename.jsp” /> 대신 <%@ include file=”filename.jsp” %>를 사용하라.

<jsp:include page=”filename.jsp” />는 실행 시점까지 실행되지 않고, <%@ include file=”filename.jsp” %>에서 지정된 파일은 컴파일시 같이 컴파일 되어, 컴파일 된 서블릿의 일부가 된다. 페이지가 실행된 후 페이지의 데이터에 의존하는 내용이 있을 경우에 한하여 <jsp:include..>를 사용한다.

.

#### “JSP:Forward” 대 “Redirect”

가능하면, *response.sendRedirect(URL)* 대신 <JSP:forward page=”filename.jsp” />를 사용하라.

<JSP:forward page=”filename.jsp” />가 빠르고 리다이렉트 실행이 좋아진다. 그 이유는 JSP:forward는 동일한 요청을 처리하고, 브라우저에서 새로운 JSP 파일 처리를 알리지 않는다(웹 컨테이너 내에서 처리된다.) 그러나, response.sendRedirect(URL)은 브라우저에서 서버로부터 새로운 요청을 만들어 처리하게 되므로 느리다.

#### 페이지 세션

기본적으로 HttpSession 모든 JSP 파일에 전달된다. 만약 코드에 HttpSession이 필요하지 않다면, session을 사용하지 않게 함으로써 오버해드를 줄이고 성능을 향상시킬 수 있다. session을 사용하지 않기 위해서는 <%@ page session=“false”%> 설정하면 된다.

JSP 파일 내에 useBean을 사용하면, useBean의 사용 범위를 최소화하여야 한다. 필요가 없을 경우, 페이지 범위를 항상 “scope=page”로 설정한다.

#### JSP 파일 컴파일

웹 서버는 사용자에게 보여질 때까지 JSP 파일을 servlet으로 컴파일 하지 않는다. 이럴 경우 두 가지 성능 문제를 일으킨다.

(1) JSP 파일이 컴파일 되는 동안 기다려야 한다.

(2) JSP 파일을 컴파일 하기 위하여 CPU를 사용하면 다른 사용자에게 영향을 미칠 수 있다.

이러한 영향을 줄이기 위하여, 웹 서버는 JSP 파일을 미리 컴파일 하는 기능을 제공한다.

### Java

JSP 파일, Java Beans, Servlets, JPO에서 자바 코딩 방법은 다음과 같다.

#### “List” 와 “Vector” 의 예

List의 사이즈를 알고 있는 경우, 생성자에 그것을 지정할 수 있다. List의 생성자 사이즈를 지정하지 않으면 기본 사이즈가 10이다. 만약 리스트 또는 벡터에 추가된 엘리먼트들의 수가 10을 초과하면, 용량은 두 배가 된다. 그 용량이 초과될 때, 그것은 다시 배가 된다. 크기를 지정함으로써 객체에 의해 요구된 메모리의 량을 줄여라.

*Bad:*

SelectList SelectStmts = new SelectList();

SelectStmts.addId();

SelectStmts.addCurrent();

*Better:*

SelectList SelectStmts = new SelectList(2);

SelectStmts.addId();

SelectStmts.addCurrent();

#### “Pattern” 객체 사용

패턴 객체를 생성하여 필요 없는 메모리에 추가되지 않도록 한다. 객체의 무분별한 생성은 단지 Garbage Collection을 유발시킨다.

*Bad:*

Pattern relPattern = new Pattern(memberTaskStr);

ExpansionWithSelect memberSelect =

BO.expandSelect(context, relPattern.getPattern()….);

*Better:*

ExpansionWithSelect memberSelect =

BO.expandSelect(context, memberTaskStr….);

#### “Hashtable” 객체 사용

HashTable은 인스턴스를 초기화할 필요가 없다. 초기화는 Null로 하고, 루푸에서 초기화를 한다. 객체의 무분별한 생성은 단지 Garbage Collection을 유발시킨다.

*Bad:*

Hashtable attributesTable = new Hashtable();

While (relItr.next() {

attributesTable = relItr.obj().getTargetData();

*Better:*

Hashtable attributesTable = null;

While (relItr.next() {

attributesTable = relItr.obj().getTargetData();

#### “StringList” 와 “SelectList” 사용

BusinessObject.select 와 BusinessObject.expandSelect에 변수를 넘기기 위해 비어있는 StringList와 SelectList를 객체를 생성하지 마라. 그 BussinessObject method는 JSP 파일에서 생성한 StringList 객체 대신에 Null을 받아들일 수 있다. 이렇게 함으로써, Garbage Collection할 객체의 수를 줄일 수 있다.

*Bad:*

ExpansionWithSelect projectSelect = bo.expandSelect(context, relString, typeString,

RelSelect, new SelectList(), true, true, (short) 1);

*Better:*

Public static SelectList EMPTY\_SELECT\_LIST = new SelectList(0);

ExpansionWithSelect projectSelect = bo.expandSelect(context, relString, typeString,

RelSelect, EMPTY\_SELECT\_LIST, true, true, (short) 1);

#### Business Object의 TNR(Type, Name, Revision) 가져오기

TNR의 정보를 가져오기 위하여 BusinessObject를 open하지 않는다. 이미 그 정보를 객체는 포함하고 있다.

*Bad:*

while (relItr.next()) {

BusinessObject routeObject = new BsinessObject(relItr.obj().getTo());

routeObject.open(context);

String name = routeObject.getName();

routeObject.close(context);

*Better:*

While (relItr.next()) {

BusinessObject routeObject = new BsinessObject(relItr.obj().getTo());

String name = routeObject.getName();

#### 공백 문자열 비교

length method를 사용하여 공백 문자열을 비교한다. 사소하게 보일지 몰라도 Web 개발 시 개선하는데 도움을 준다.

*Bad:*

If (relString != null && relString.equals(“”)) { …

*Better:*

If (relString != null && relString.length() == 0) { …

#### String (java.lang.String)

개발자가 문자열을 오버라이드하거나 단순히 + 연산자를 사용하여 정보를 덧붙인 것은 일반적인 방식이다.

그러나, 이 방식은 당신의 애플리케이션의 성능에 부정적인 영향을 줄 수 있습니다. 만약 당신이 문자열의 값을 조작하는 것이 필요한다면, 당신은 대신에 StringBuffer Class를 사용하여야만 합니다. 단순히 String은 변경할 수 없는 class이기 때문입니다. 만약 당신이 +연산자를 수행하면 다음 단계들이 수행되다

1. StringBuffer 생성
2. StringBuffer에 원본 문자열을 붙인다.
3. StringBuffer에 덧붙이고자 하는 문자열을 붙인다.
4. StringBuffer를 String으로 변환한다.
5. 원래 문자열에 새로운 문자열을 값을 업데이트한다.
6. 기존 문자열은 Garbage Collection 대상이 된다.

위에서 보는 것과 같이, 프로그램에 많은 불필요한 오버해드가 발생하고 코드는 매우 비능률적입니다.

#### 반복문

반복문에서 여러 번 동일한 function을 실행시키는 방법은 매우 일반적으로 잘못된 방법입니다.

BAD:

*for (int i=0; i<mapList.size(); i++) { ... }*

BETTER:

*int upperLimit = mapList.size();  
 for (int i=0; i<upperLimit; i++) { ... }*

#### 조건 비교

조건부 문장을 수행할 때, 처리 시간 별로 조건을 사용하는 것을 적극 추천한다. 불필요한 처리시간을 줄이기 위하여 빠른 명령어를 넣는다. “or” 조건에서, 자바에서 조건이 일치하면 다음 조건은 비교하지 않는다.

BAD:

*if (String1.equals(String2) || boolean1) { }*

BETTER:

*if (boolean1 || String1.equals(String2))*

*{ }*

#### StringTokenizer (java.util.StringTokenizer)

StringTokenizer은 String을 구분 위해 사용하는 가장 일반적인 method 중 하나입니다. 그러나, 일련의 구분 기호가 속한 String의 각 캐릭터를 비교하기 때문에 오버해드를 발생시킬 수 있습니다.

StringTokenizer을 사용하지 않고 String을 구분하는 여러 가지 방법이 있습니다. StringTokenizer 대신 사용하는 방법은 다음과 같다.

public static void myTokenizer()String s, String delimiter)

{

String sub = null;

int i =0;

int j =s.indexOf(delimiter); // First substring

while( j >= 0) {

sub = s.substring(i,j);

i = j + 1;

j = s.indexOf(delimiter, i); // Rest of substrings

}

sub = s.substring(i); // Last substring

}

그러나, StringTokenizer class 대신 위의 예제를 사용할 때 조심하시기 바랍니다. 프로그램의 유지보수를 어렵게 할 수 있습니다.

## JPO 코드

자바 프로그램 객체(JPO)는 자바 언어로 쓰여진 코드를 포함한 프로그램 객체의 또 다른 타입입니다. 일반적으로, ENOVIA의 어느 프로그램에서도 JPO를 사용할 수 있습니다.

JPO들은 최상위 ENOVIA ADK 프로그램 interface를 자바 프로그램에 사용하고 그 프로그램이 동적으로 호출되도록 합니다.

ENOVIA Collaboration Kernel 내에서 프로그램이 실행될 때의 스레드에서 동일한 Context 및 트랜잭션을 공유합니다. 사실상, 자바 프로그램은 Kernel과 같은 자바 가상 머신(JVM)내에 실행됩니다.

### JPO 이름 매크로

JPO class를 위해 스키마 객체 이름 사용에 대한 2개의 문제가 있다. 첫 번째 문제는 ENOVIA 의 JPO 스키마 객체 이름이 자바 컴파일러가 실행되지 않는 원인이 될 non-alpha 문자와 공백을 포함하는 것이 허용된다. 두 번째 문제는 JPO 스키마 객체 이름이 JPO 내의 코드에 의해 자동적으로 만들어지지 않고 동적으로 변경된다.

ENOVIA에서 제공하는 특별한 매크로 명령은 JPO 스키마 객체 이름을 자바 호환 이름으로 나타내기 위하여 이용된다. 이러한 매크로가 적절하게 JPO 스키마 개체의 이름은 변경할 때 자동적으로 바뀌게 됩니다.

#### class 선언

첫 번째 매크로는 자바 class명 정의 시 JPO 이름을 참고하도록 사용됩니다.

|  |
| --- |
| **Pattern : ${CLASSNAME}** |
| public class ${CLASSNAME}  {  …  } |

#### class 참조

두 번째 매크로는 JPO 코드 내에서 다른 JPO class를 참조할 때 사용됩니다. 예를 들어 다른 JPO를 상속받거나 메소스를 호출할 때 사용됩니다.

|  |
| --- |
| **Pattern : ${CLASS:jpo\_name}** |
| public class ${CLASSNAME} extends ${CLASS:emxDomainObjectBase}  {  …  ${CLASS:emxContextUtil} contextUtil = new ${CLASS:emxContextUtil}(context, null);  …  ${CLASS:emxContextUtil}.startTransaction(context,true);  …  } |

### 필수 method

각 JPO는 규정된 몇 가지 method를 가져야 합니다. 일반적으로 모든 method의 처음 변수는 ENOVIA.db.Context 타입을 가집니다.

#### 생성자

각 JPO는 Context 객체와 String[]을 가지는 생성자를 생성합니다. String[]는 ObjectId와 같은 생성자에게 전달하는 필요한 변수를 가진다.

|  |
| --- |
| public ${CLASSNAME} (Context context, String[] args) throws Exception  {  …  } |

#### Main Entry Point (mxMain method)

각 JPO는 mxMain이라고 불리는 public method를 가지고, int형을 반환한다. 이 method는 method 이름이 지정이 되지 않았을 경우 기본적으로 호출이 된다. 다른 프로그램에서 method 없이 JPO를 호출하는 것을 막기 위해 아래와 같이 mxMain을 기술하여 예외를 발생시킨다.

|  |
| --- |
| public int mxMain(Context context, String[] args) throws Exception  {  if (true)  {  throw new Exception("must specify method on this JPO invocation");  }  return 0;  } |

### JPO 호출

#### 외부에서 JPO method 호출

JPO.invoke()나 MQL을 통하여 method가 호출되는 경우, Context와 String[] 두 가지 인수만 허용된다.

|  |
| --- |
| public void **floatEBOMToEnd**(Context context, String[] args) throws Exception  {  ...  } |

#### 내부에서 JPO method 호출

JPO.invoke()나 MQL을 통하여 호출되는 JPO method가 아닌 경우, 다양한 인수를 가질 수 있다.

|  |
| --- |
| public int updateModel(Context context, String[] args) throws Exception  {  HashMap programMap = (HashMap) JPO.unpackArgs(args);  ……  updateConnection(context, strObjectId, …);  return 0;  }  protected void **updateConnection**(Context context, String strObjectId, …) throws Exception  {  ……  } |

#### ADK 접근

JPO class는 아래와 같이 일반 java class 나 JSP 코드상에서 JPO의 public method에 접근하는 2가지 method를 제공한다. 이때 호출하려는 public method는 인수로 matrix.db.Context 와 java.lang.String[] 만을 가질 수 있다.

|  |
| --- |
| int ret = JPO.invoke(Context context, String className, String[] initargs, String methodName, String[] methodargs);  Object ret = JPO.invoke(Context context, String className, String[] initargs, String methodName, String[] methodargs, java.lang.Class retType); |

#### JPO method에서 자바 개체를 매개 변수로 전달

JPO class는 String에서 Object로 serializing하거나, Object에서 String으로 de-serializing하는 두 가지 method가 있다. 각각 packArgs와 unpackArgs method이다. packArgs method는 Java Object를 String[]로 변환하고, unpackArgs method는 String[]을 Java Object로 변환한다.

|  |
| --- |
| public static void sendMessage(Context context, …) throws FrameworkException  {  ……  Map note = new HashMap();  note.put("toList", toList);  ……  String[] args = JPO.packArgs(note);  JPO.invoke(context, "emxMailUtil", null, "sendMessage", args);  ……  }  public static int sendMessage(Context context, String[] args) throws Exception  {  if (args == null || args.length < 1)  {  throw (new IllegalArgumentException());  }  Map map = (Map) JPO.unpackArgs(args);  ……  } |

### 주석

일반적으로 JPO 주석은 3.1.1주석에 기술된 룰에 따라 주석을 입력하고, ENOVIA 버전 정

보와 Trigger에 의해 호출되는 JPO method의 경우, 다음과 같이 주석을 입력한다.

|  |
| --- |
| /\*\*  \* ……  \* @since Engineering Central 10.7.SP1  \* @trigger PolicyECPartStateReviewPromoteAction.  \*/ |

### Table을 위한 JPO 프로그램

#### JPO for Column Value

Table 컬럼 설정이 program 또는 programHTMLOutput일 경우, JPO를 통하여 Table 각 열의 정보를 가져올 수 있다. 이 경우 성능을 고려하여 DB 호출을 최소화하기 위한 ADK method를 사용하도록 한다.

|  |  |  |
| --- | --- | --- |
| **BusinessObject.getSelectBusinessObjectData** | | |
| **Parameters** | context | the context for this request |
| oidList | a String List of OIDs |
| selectStmts | A StringList of select statements (name, owner, etc.) |
| **Returns** | a complete set of data of type BusinessObjectWithSelectlist, for all the  business objects you passed in as a list | |

|  |  |  |
| --- | --- | --- |
| **Relationship.getSelectRelationshipData** | | |
| **Parameters** | context | the context for this request |
| oidList | a String List of OIDs |
| selectStmts | a StringList of select statements (name, owner, etc.) |
| **Returns** | a complete set of data of type RelationshipWithSelectlist, for all the  business objects you passed in as a list | |

<Example>

|  |
| --- |
| public static Vector getVault(Context context, String[]args)throws Exception  {  // get parameter  HashMap programMap = (HashMap) JPO.unpackArgs(args);  MapList relBusObjPageList = (MapList)programMap.get("objectList");  HashMap paramMap = (HashMap)programMap.get("paramList");  // initialize return variable  Vector vaultList = new Vector();  // setting select lists  StringList listSelect = new StringList(1);  listSelect.addElement("vault");  // to store object ids for each row  String objIdArray[] = new String[relBusObjPageList.size()];    if ( relBusObjPageList != null)  {  // setting object id  for (int i = 0; i < relBusObjPageList.size(); i++)  {  objIdArray[i] = (String)((HashMap)relBusObjPageList.get(i)).get("id");  }    // get the vault for the object  BusinessObjectWithSelectList vaultSelectList = null;  vaultSelectList = BusinessObject.getSelectBusinessObjectData(objIdArray, listSelect);  for (int i = 0; i < relBusObjPageList.size(); i++)  {  String vaultName = vaultSelectList.getElement(i).getSelectData("vault");  vaultList.add(vaultName);  }  }  return vaultList;  } |

#### JPO for Column Update

아래와 같이 Table 컬럼이 설정되어 있는 경우, JPO를 사용하여 해당 컬럼의 값을 업데이트 할 수 있다.

• Update Program=JPO name

• Update Function=JPO method name

만약 업데이트가 성공되었을 경우, 이 method의 반환형으로 Boolean(또는 int), true(또는 0) 이다.

|  |
| --- |
| Public static int methodName(Context context, String[] args) throws Exception  {  HashMap programMap = (HashMap) JPO.unpackArgs(args);  HashMap paramMap = (String) programMap.get("paramMap");  HashMap requestMap = (String) programMap.get("requestMap");  // Get the required parameter values from "paramMap" - as required  String objectId = (String) paramMap.get("objectId ");  String relId = (String) paramMap.get("relId ");  String newValue = (String) paramMap.get("New Value");  String oldValue = (String) paramMap.get("Old Value");  // get languagestr from requestmap  String languageStr = (String) requestMap.get("languageStr");  // Define and add selects if required  // Process the information to set the field values for the current object  ......  return 0;  } |

#### JPO for Column Sort

컬럼 값으로 테이블 정렬을 위한 사용자 정의 알고리즘을 JPO로 정의하여 사용할 수 있다. JPO를 이용한 정렬을 위해선, 테이블 컬럼 설정으로 Sort Type과 Sort Program이 설정이 되어야 한다.

Sort Program을 JPO에서 사용하기 위하여 다음의 규칙을 따라야 한다.

• JPO class는 emxCommonBaseComparator JPO class를 상속받아야 한다.

• 정렬을 위한 JPO class는 compare() method를 정의하여야 한다. 아래는 compare() method를 정의하기 위한 템플릿을 제공한다.

|  |  |  |
| --- | --- | --- |
| **public int compare(Object object1, Object object2)** | | |
| **Parameters** | object1 | the first object to compare |
| object2 | the second object to compare. |
| **Returns** | A negative integer if first argument is less than second argument  A zero if the arguments are equal.  A positive integer if the first argument is greater than second argument. The object1 and object2 is of type Map | |

**<**Example>

|  |
| --- |
| public int compare(Object object1, Object object2)  {  // Logic to get column values to compare from object1 and object2  // Get sort info keys  Map sortKeys = getSortKeys ( );  // Get column key name defined to get column values from object1 and object2  String columnKey = (String) sortKeys.get("columnKey");  // Get column values from object1 and object2  Map m1 = (Map)object1;  Map m2 = (Map)object2;  String columnValue1 = (String)m1.get(columnKey);  String columnValue2 = (String)m2.get(columnKey);  // Code to custom compare columnValue1 and 2 goes here.  ......  } |

#### ENOVIA Framework의 정렬 JPO 프로그램

OOTB로 사용자 정의 정렬 JPO 프로그램을 제공하고 있다. Engineering Central의 EBOM relationship에서 Find Number를 정렬하는데 사용할 수 있다.

|  |  |
| --- | --- |
| JPO Name | Description |
| emxSortNumericAlphaLarger | The program first considers pure alphabetics as well as alphanumeric strings as alpha for sorting, then looks for pure numerics and sorts them.  • If both objects are numbers, does a numeric compare.  • If both objects are alphanumeric, does a string compare.  • If one object is a number and the other is alphanumeric, then Alphanumeric > Number. |
| emxSortNumericAlphaSmaller | • If both objects are numbers, does a numeric compare.  • If both objects are alphanumeric, does a string compare.  • If one object is a number and other alphanumeric, then Number > Alphanumeric. |

### Web Form을 위한 JPO 프로그램

#### JPO for Field Value

JPO를 사용하여 필드 값을 가져올 때, 다음과 같이 필드를 설정한다.

• Field Type=program / programHtmlOutput

• program=JPO name

• function=JPO method name

<Example>

|  |
| --- |
| public static Object methodName(Context context, String[] args) throws Exception  {  HashMap programMap = (HashMap) JPO.unpackArgs(args);  HashMap requestMap = (HashMap) programMap.get("requestMap");  HashMap paramMap = (HashMap) programMap.get("paramMap");    String objectId = (String) requestMap.get("objectId");  String relId = (String) paramMap.get("relId");  String languageStr = (String) requestMap.get("languageStr");    // initialize the return variable  StringList fieldValues = new StringList();    // define and add selects if required  // Process the information to obtain the values  if ( objectId != null)  {  //add the field values  fieldValues.addElement(......);  }  return fieldValues;  } |

#### JPO for Field Range Value

JPO를 사용하여 필드의 범위를 가져올 경우, 다음과 같이 필드를 설정한다.

• Input Type=combobox / radiobutton / checkbox

• Range Program=JPO name

• Range Function=JPO method name

<Example>

|  |
| --- |
| public static Object methodName(Context context, String[] args) throws Exception  {  HashMap programMap = (HashMap) JPO.unpackArgs(args);  HashMap requestMap = (HashMap) programMap.get("requestMap");  HashMap paramMap = (HashMap) programMap.get("paramMap");    // Get the required parameter values from "programMap" - as required  String objectId = (String) paramMap.get("objectId ");  String relId = (String) paramMap.get("relId ");  String languageStr = (String) paramMap.get("languageStr");    // initialize the return variable HashMap tempMap = new HashMap();  HashMap tempMap = new HashMap();    // initialize the Stringlists fieldRangeValues, fieldDisplayRangeValues  StringList fieldRangeValues = new StringList();  StringList fieldDisplayRangeValues = new StringList();    // Process information to obtain the range values and add them to fieldRangeValues  // Get the internationlized value of the range values and add them to fieldDisplayRangeValues  fieldRangeValues.addElement(xxx);  fieldDisplayRangeValues.addElement(internationlised xxx);  tempMap.put("field\_choices", fieldRangeValues);  tempMap.put("field\_display\_choices", fieldDisplayRangeValues);  return tempMap;  } |

#### JPO for Field Update

필드명을 사용하여 필드의 변경된 값을 얻기 위하여 requestMap을 사용하는 JPO프로그램의 일부이다.

String[] revisionValues=(String[]) requestMap.get("Revision");

String attributeValue=(String) requestMap.get(attrName);

만약 테이블이나 structure 테이블의 수정 모드에서 사용되는 method를 생각하고 있다면, paramMap을 사용하여 서로 다른 값을 가져올 수 있다.

String newValue = (String) paramMap.get("New Value");

or

String[] newValues = (String[]) paramMap.get("New Values");

Web Form의 필드를 업데이트 하는 JPO 프로그램에서 requestMap으로부터 timeZone, charSet, localObj, languageStr 값들을 얻을 수 있다.

String timeZone = (String) requestMap.get("timeZone");

String charSet = (String) requestMap.get("charSet");

Locale localeObj = (Locale) requestMap.get("localeObj");

String languageStr = (String) requestMap.get("languageStr");

## 응용 프로그램 보안 점검 체크 리스트

다음 표는 두산 그룹에서 시스템 구축 프로젝트를 수행할 때, 응용프로그램 보안 점검을 위한 체크리스트이다. 프로그램 개발을 할 때 아래 내용을 숙지하여 보안 규정을 준수하도록 노력한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NO** | **점검항목** | **코드** | **점검 내용** | **증적** | **대상** | **비고** |
| 1 | 로그인 및  사용자 인증 | A-01 | 로그인 전 비인가자 접근경고 및 주의를 알리는 보안메세지 출력(게시)하고 있습니까?  설명) 사내 임직원 용도로 사용되는 응용프로그램에는 비 인가자의 불법적인 접근을 제한하기 위한 경고 메시지를 출력(게시)하여야 합니다.  조치방법) 임직원 로그인 페이지에 아래와 같은 표준 경고 문구를 출력(게시)하도록 구현하여야 합니다.   예시 1) Warning Banner  "This Site is for the use of aututhorized Doosan Group Personnel only and by accessing this   site you hereby consent to the site being monitored.  Any unauthorized user will be considered a breach of Doosan Group Information Security   Policies and may also be unlawful under low"   예시 2) 경고 문고  "본 사이트는 두산 임직원 및 인가된 사용자만 사용할 수 있으며, 사용자 접속 모니터링 및 로그  를 기록하고 있습니다.   불법 사용시에는 법령에 의해 민∙형사상 책임을 질 수 있습니다."  증적제출) 응용프로그램 구현 시 로그인 페이지에 해당 경고 문구가 출력되는지 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | 사내 임직원 용도 | 로그인 페이지 캡쳐 화면 |
| 2 | 로그인 및  사용자 인증 | A-02 | 로그인 인증완료 후 사용자 최종 접속정보 출력(게시)하고 있습니까?  설명) 로그인 인증완료 후 사용자 최종 접속정보를 사용자에게 알려줌으로써 자신이 모르는 사이에 해당 아이디가 도용되었는지를 판단하고 대응할 수 있습니다.  조치방법) 로그인 인증완료 후 팝업 또는 게시를 통해 사용자가 최종 접속정보를 인식하고 계정도용 여부를 확인할 수 있도록 합니다.. 팝업 또는 게시를 통해 사용자에게 알려줄 경우 마지막 접속시간(년:월:일:시:분) 또는 마지막 접속시간 + 접속주소(IP Address) 를 알려주어야 합니다.  ※ 관리자 계정의 경우, 로그인 접속 이력을 DBMS에 기록 관리하여야 한다. (접속일시, 로그인 IP , 로그인 계정)   증적제출) 로그인 인증 후 사용자 최종 접속정보(마지막 로그인 일시 및 접속 IP) 출력여부를 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | 사내 임직원 용도 | 로그인 후 사용자 접속정보 출력페이지 화면캡쳐 |
| 3 | 로그인 및  사용자 인증 | A-03 | “계정, 암호” 입력방식 인증 이상의 인증절차를 수행하고 있습니까?  설명) 계정∙암호 입력방식의 인증만 사용하게 된다면, 계정정보 노출 시 또는 무작위 공격을 통해서도 계정을 도용달할 가능성이 있다. 그러므로 계정∙암호 인증방식 외 다른 방식의 인증절차를 추가로 수행함으로써 보안을 강화해야 합니다.  조치방법) 다음과 같은 5가지 방식의 인증 절차 중 어플리케이션에 맞는 방법을 선택하여 적용하도록 합니다.  Case1) 계정∙암호 입력방식 인증 : 자체 Database 계정암호 관리  Case2) 계정∙암호 통합 인증 : Active Directory 도메인계정(SSO)  Case3) 계정∙암호 + IP Address 접속대역 인증  Case4) 계정∙암호 + MAC Address 인증  Case5) 계정∙암호 + 2차인증(인증서, OTP(One Time Password))  증적제출) 별도의 증적은 없으며, PM의견란에 사용자 인증방식에 대해 자세히 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 4 | 로그인 및  사용자 인증 | A-04 | 로그인 실패 시, 최소한의 실패 원인만 출력하도록 구현하였습니까?  설명) 로그인 실패 시 알려주는 각각의 메시지 정보는 공격자에게 좋은 정보가 될 수 있습니다. 그러므로 로그인 실패 시 최소한의 정보만을 노출할 수 있도록 구현되어야 합니다.  조치방법) 로그인 실패 시 “아이디가 존재하지 않습니다.”, “패스워드가 일치하지 않습니다.” 처럼 공격자가 유용한 정보를 얻을 수 있는 오류사항이 아닌 다음과 같이 어느 항목이 틀렸는지 확인이 어려운 오류문구를 사용자에게 알려주어야 합니다.   예시 1) 로그인 오류문구  "현재 입력하신 아이디가 등록되어 있지 않거나, 아이디 또는 비밀번호를 잘못 입력하였습니다"  증적제출) 등록되지 않은 계정, 패스워드를 입력하여 출력되는 오류메시지 화면을 캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | ALL | 잘못된 로그인정보 입력 시 출력되는 오류메시지 화면캡쳐 |
| 5 | 로그인 및  사용자 인증 | A-05 | 개인정보 전송(로그인 정보 포함)시 보안성 유지를 위한 보안서버 설치하였습니까?  설명) 인터넷 상에서 암호화되지 않고 전송되는 개인정보는 가로채기 등의 해킹기법 등을 통해 해커에게 쉽게 노출될 수 있으므로, 웹 서버에 보안서버 설치하여 암호화 채널을 통해 개인정보가 전송될 수 있도록 구축하여 운영하여야 합니다.   조치방법) 개인정보를 입력 또는 전송되는 모든 웹 페이지에 대해서 보안서버가 적용되어야 전송될 수 있도록 구축하여야 합니다. (응용프로그램 방식 / SSL 인증서 방식)   ① 주민번호, 여권번호, 운전면허, 금융거래정보  ② 계정ㆍ암호 전송, 계정ㆍ암호 찾기, 회원가입, 회원정보 변경  ③ 그외 개인 식별이 가능한 정보를 입력전송하는 웹페이지 (FAQ, Q&A)   증적제출) 로그인 페이지 등 보안서버가 적용된 웹 페이지를 인터넷브라우저에서 확인 시 적용된 인증서 정보를 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | 개인정보 전송 응용프로그램 | 인터넷 브라우저에서 적용된 인증서 정보 캡쳐 화면 |
| 6 | 로그인 및  사용자 인증 | A-06 | 로그인 후 사용자가 일정시간 동안 미사용 시 자동 Logoff 하도록 설정하여 구축하였습니까?  설명) 로그인 후 작업 중 자리에서 장시간 이석에 대비하여 키보드 또는 마우스 입력이 장시간 없을 시에 세션을 강제로 Logoff 하여야 합니다. 이를 통해 불법적인 사용자가 로그온된 화면을 도용하여 불법 행위를 하는 것을 막는 역할을 수행할 수 있습니다.  조치방법) 일정시간동안 미사용 시 사용자 계정을 Logoff 시키도록 해야 한다. 세션 타임아웃 시간(20분 권장)은 응용프로그램의 특성에 맞게 설정하여야 한다.  증적제출) 응용프로그램 세션 타임아웃(Time Out)이 설정된 부분을 화면캡처하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | ALL | 응용프로그램 내 세션타임아웃 설정 부분 화면 캡쳐 |
| 7 | 로그인 및  사용자 인증 | A-07 | 계정 생성 및 관리 기준에 따라 권한 관리가 이루어지고 있습니까?  설명) 시스템에 접근 가능한 사용자 계정 목록을 관리하여 상시적으로 이루어지는 계정 생성 및 삭제에 대한 사용자 계정에 대한 접근권한 관리를 하여야 합니다. 삭제되어야 할 계정이 삭제되지 않아 발생하는 비인가 접근 및 침해사고를 미연에 방지하여야 합니다.  조치방법) 접근 가능한 사용자 계정 목록을 수시로 관리하고 계정 삭제 및 부서 이동에 따른 권한 설정도 즉시 이루어져야 합니다. 시스템에 대한 접근권한은 인가된 사용자만 로그인 가능하도록 개발해야 합니다.  ① 접근 가능한 사용자 계정 접근 권한 목록 관리  (통합인증 후 인가된 사용자만 로그인 가능하도록 개발)   증적제출) 사용자 계정에 대한 접근 권한 관리 페이지를 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | Active Directory  통합인증  사용 응용프로그램 | 사용자 계정 접근 권한 관리 페이지 화면캡쳐 |
| 8 | 로그인 및  사용자 인증 | A-08 | 암호화 저장되어야 하는 정보는 (File 혹은 DataBase) 암호화하여 저장하고 있습니까?   설명) 주민등록번호, 금융거래정보 등 사용자의 개인정보와 사용자 및 관리자 패스워드 등 중요한 정보에 대해서는 반드시 암호화 저장을 수행하여, DB 노출 시에도 정보가 쉽게 노출되지 않도록 해야합니다.  조치방법) 다음의 정보들에 대해 File 혹은 DataBase 저장 시 암호화를 수행하여야 한다   ① 사용자 및 관리자 패스워드(일방향 암호화)   ② 주민등록번호, 금융거래정보 등 개인정보(안전한 암호알고리즘으로 암호화)   증적제출) 별도의 증적은 없으며, 암호화 대상정보에 적용된 암호화 알고리즘은 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 9 | 로그인 및  사용자 인증 | A-09 | 암호생성 및 관리는 그룹 암호정책 기준에 준수하고 있습니까?  설명) 패스워드 추측공격에 대한 대응으로 패스워드 최소길이, 패스워드 복잡성, 계정명과 동일한 암호 사용 금지 항목을 준수하여야 합니다. 사내 암호 관리 기준으로 정의하여 사용자들이 설정하는 패스워드가 그룹 암호정책 기준을 만족하도록 강제화되어야 합니다.  조치방법) 다음의 항목처럼 패스워드 관리기준을 수립하고 운영하여야 합니다.  Case 1) 사내 Active Directory 도메인계정 통합인증 사용  - 도메인계정 암호정책 유지  Case 2) 자체 계정∙암호를 수집/보유하여 운영  - 사용자 계정 암호는 최소 8자리 이상  - 관리자 계정 암호는 최소 8자리 이상  - 대문자/소문자/숫자/특수문자 중 3가지 이상 조합  - 계정명과 동일한 암호 사용 금지  Case 3) 개인정보 수집∙보유 인터넷 공개 대고객용 응용프로그램  - 사용자 계정 암호는 최소 8자리 이상  - 관리자 계정 암호는 최소 8자리 이상  - 문자/숫자/특수문자 중 3가지 이상 조합  - 계정명과 동일한 암호 사용 금지  증적제출) 별도의 증적은 없으며, 응용프로그램에서 사용되는 암호 생성 시 그룹 정책기준에 맞춰 사용하는지에 대해 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 10 | 소스코드 개발 보안 | B-01 | 소스코드에 공격자에게 유용한 응용프로그램 특정 정보 및 민감한 정보가 삽입되어 있지는 않습니까?  설명) 잘못된 주석 처리나 과도한 주석 처리로 인해서 민감한 정보가 노출되는 경우 자동화된 도구 또는 ‘HTML 소스 보기’ 를 통해서 공격자에게 유용한 정보로 제공될 수 있으며, 이를 통해서 애플리케이션의 작동원리나 주요한 정보가 노출되어 공격의 단초로 활용될 수 있습니다.   Case 1) 소스코드에 불필요한 주석(Comment)로 정보노출 금지    서버정보, Database 연결정보, 테스트계정    관리자페이지 정보   Case 2) 소스코드에 인증우회코드를 삽입하여 비정상 로그인 금지    인증우회용 계정 코드, 인증우회용 인터넷주소    인증우회용 IP Address, MAC Address   조치방법) 응용프로그램을 개발할 때 주석을 통해서 주요한 정보를 남겨 놓는 것은 디버깅, 추후 업데이트 등을 고려하여 바람직한 것이나 애플리케이션의 역할, 수행과정, 디버깅 정보, 애플리케이션 소스가 주석을 통해서 노출 될 경우 이를 통해서 중요한 정보가 노출될 수 있다. 따라서 주석 정보는 HTML 페이지를 통해서 사용자에게 전달되지 않도록 응용프로그램 개발 페이지에 남기도록 한다.  증적제출) 별도의 증적은 없으며, PM이 개발 구축 완료 후 소스코드 형상관리의 개발 페이지에 불필요한 정보가 노출되고 있는지 확인한 내용을 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 11 | 소스코드 개발 보안 | B-02 | 정상적인 로그인 후 접속할 수 있는 모든 페이지에는 응용프로그램 경로가 포함된 전체 인터넷 주소 를 직접 입력하여 인증 없이 접속할 수 없도록 인증 검증 모듈이 적용되어 있습니까?  설명) 인증과 권한이 필요한 애플리케이션에서 이를 요청한 사용자에 대해 인증 여부와 적정한 권한의 소유 여부를 검증하지 않는 경우 공격자는 유추 또는 애플리케이션의 분석을 통해 획득한 정보를 바탕으로 인증과 권한 검증이 부재한 애플리케이션에 접근할 수 있습니다. 따라서 로그인하지 않은 비인가자가 인증과 권한 후에 접근이 가능한 페이지에 URL로 직접적인 접근이 불가능하도록 하여야 합니다.   조치방법) 게시판, 관리자 페이지 등 로그인 후에 접근 가능한 애플리케이션은 실행 시에 해당 요청이 정상적인 인증을 받은 사용자인가? 인증된 사용자가 요청한 작업을 수행할 수 있는 권한자인가? 반드시 검사하여야 한다. 모든 애플리케이션 페이지 ‘접근 통제 표(Access Control Matrix)’ 등을 작성하고 비정상적인 사용자가 할당되지 않은 권한을 사용할 수 있는 지 검증 기능을 적용하여야 합니다.  ① 전체 페이지 본수 확인   ② 인증필요 페이지 본수 확인   ③ 인증 점검 모듈 적용 확인   증적제출) 응용프로그램 디렉터리 별 페이지의 인증 필요 페이지와 인증 점검 모듈이 적용되어 있는지 확인할 수 있는 자료를 증적으로 첨부하시기 바랍니다.  PM의견) | O | ALL | 응응프로그램 페이지별 인증 점검 모듈 반영여부 확인 데이터 |
| 12 | 소스코드 개발 보안 | B-03 | 응용프로그램의 권한 관리를 위해 생성하는 쿠키값은 암호화하여 적용하고 있습니까?   설명) 애플리케이션에서 주요한 정보를 클라이언트 측의 Cookie 값으로 저장하거나 클라이언트 측에서 전송되어 오는 데이터를 신뢰하는 것은 보안의 위험요소가 됩니다. 공격자는 웹 브라우저와 애플리케이션 간에 오가는 데이터를 수정할 수 있으므로 이를 통해 사용자의 Cookie 정보를 수정, 타 사용자 또는 관리자의 정보 또는 권한이 도용될 수 있습니다.  조치방법) 인증과 권한에 사용되는 주요한 정보는 사용자 측에 해당 정보가 저장되는 Cookie가 아닌 서버 측에 해당 정보가 저장이 되는 Server Side Session 을 사용하여야 합니다. Cookie로 클라이언트 측에 저장되는 정보는 암호화 등을 통하여 해당 정보를 사용자 측에서 식별 불가능하도록 구현하여야 합니다.    Case 1) Cookie 대신 Server Side Session을 사용 권장 예시.   Server Side Session Cookie: SESSIONIDCSBATTCD=JGNDKNCAFDAAOFKNDD  Case 2) Cookie 사용 시, 값에 대한 암호화 필수 적용 예시.   <일반적인 Cookie Value> set cookie: Userid=pcuser;usernumber=123411  <암호화된 Cookie Value> set cookie: userid=3JNDJSAAJ$; usernumber=$JDF$D123FA  증적제출) 응용프로그램에서 생성하는 쿠키값에 대해 클라이언트(사용자)에서 실제 생성되는 값을 화면캡쳐하여 증적으로 제출하시기 바랍니다.   PM의견) | O | ALL | 생성된 쿠키 정보 화면 캡쳐 또는 로그인 후 HTTP 헤더 정보 |
| 13 | 소스코드 개발 보안 | B-04 | 입력값 검증 모듈 적용이 적용되어 있습니까? (불필요한 입력값 필터링/치환)  설명) 응용프로그램에서 동적 데이터 호출 등에 사용되는 매개변수(입력값)를 포함하여 사용자가 생성하는 매개변수(입력값)가 클라이언트에서 서버로 전송될 때 입력값에 대한 검증이 부재하거나, 부적절하게 검증할 경우 공격자에 의해 권한 상승, 주입식 공격 등에 악용될 수 있습니다. 입력값 검증 기능이란 모든 입력값에 대해 응용프로그램 설계 상 의도대로 데이터 타입(Type), 길이(Length) 등에 대해 유효성을 검증하는 것을 의미합니다.  조치방안) 입력값 검증 기능은 허용되어야 하는 문자, 숫자를 정의하고 그 외 문자들에 대해서는 필터링하도록 Positive 방식으로 입력값의 유효성을 검증하여야 하며, 반드시 서버 측에서 수행하도록 구현하여야 합니다. 하기 내용 외 더 자세한 조치 방안은 응용프로그램 보안가이드를 참조하시기 바랍니다.   ① SQL-Injection / Command-Injection 공격 필터링  불필요 메타문자 / Null 문자 입력시 필터링하도록 모듈 적용   - 로그인페이지 input 필터링 적용  - 계정에 메타문자 & null 포함하여 입력 시 필터링 적용  - 검색페이지 input (조회 / 검색 / 찾기 등) 필터링 검증  - 검색단어 및 조건에 null / 메타문자를 포함하여 입력 시 필터링 적용  - 웹 주소 및 파라미터 input 필터링 적용  (DB에 호출하여, 결과를 출력하는 모든 웹페이지 및 파라미터를 보유한 스크립트)  - input에 메타문자(@, ‘, -, +, %, =) 를 입력받을 경우, 필터링을 적용(치환)     ② Cross Site Scripting(XSS) 삽입 구문 필터링  사용자가 입력한 JavaScript, HTML 태그 필터링 모듈 적용   - 게시판 형태 input (제목/본문) 필터링 적용  - 메타문자 치환 치환전 치환후 < < > > 응용프로그램보안가이드 참조   - 결재 형태 input (제목/본문) 필터링 적용  - Email 형태 input (제목/본문) 필터링 적용  증적제출) 응용프로그램에서 사용되는 모든 매개변수에 대해 입력값 검증 적용 여부를 확인할 수 있는 매개변수 입력값 검증 적용표 (Input Validation Check Matrix)를 작성하여 증적으로 첨부하시기 바랍니다.   PM의견) | O | ALL | 매개변수 입력값 검증 기능 적용표 |
| 14 | 소스코드 개발 보안 | B-05 | File Upload 스크립트 보안 검증 모듈이 적용되어 있습니까? (검증 기능은 Server Side로 구현)   설명) 업로드 되는 파일의 확장자에 대한 적합성 여부를 검증하는 루틴이 적절히 구현되지 않으면 공격자가 조작한 Server Side Script 파일을 업로드하고 업로드 된 파일이 서버 상에 저장된 경로를 유추한 후 이 경로를 통해 Server Side Script 파일을 실행하여 쉘을 획득할 수 있고 이러한 과정을 통해 웹 서버의 권한이 공격자에게 제공될 수 있습니다.  조치방법) 아래와 같이 업로드 될 파일의 확장자를 제한하는 기능을 구현하여야 합니다. 게시판 등에 Embeded 되는 에디터는 해당 개발 업체에게 문의하여 조치하여야 합니다.   Case 1) 정해진 파일형식 외에는 Upload 금지 (권고)  Case 2) 금지파일 형식 만 Upload 금지 (다른 확장자는 모두 허용) Upload 금지파일 형식   확장자 웹 스크립트 언어 (확장자 형식 대문자 / 소문자 모두 적용)  .asp / .aspx / .jsp / .php / .inc .cgi / .html / .htm / .js / .jar / .jhtml .php3 / .phtml   Case 3) Upload 형식(헤더)을 검증하는 모듈 적용   (예. 실제 그림파일인지 Header를 검증 후 Upload)  ※ 주의) 웹서버의 구동권한은 제한된 계정(시스템 관리자 권한으로 구동 금지)으로 구동하고, 업로드 되는 경로의 디렉터리는 반드시 실행권한이 제거된 상태로 운영되어야 합니다.  증적제출) 별도의 증적은 없으며, 단 확장자 검증 모듈 적용 / 웹 서버 구동 권한 / 업로드 디렉터리의 실행권한 제거 처리 내역을 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 15 | 소스코드 개발 보안 | B-06 | File Download 스크립트 보안 검증 모듈이 적용되어 있습니까? (검증 기능은 Server Side로 구현)  설명) 파일 다운로드를 처리하는 응용프로그램이 다운로드 할 파일명 처리를 제대로 하지 못할 경우 상대경로 입력 등을 통해 첨부파일이 아닌 시스템상의 임의의 파일을 다운받는 것이 가능할 수 있습니다. 일반적으로 URL 경로에서 임의의 문자(“../”, “..\” 등)나 주요 파일명(passwd 등)에 대한 필터링을 하지 않는 경우 발생합니다.  조치방법) 경로와 파일의 입력값으로 상대경로( ” ../ ”, “ ..\ ” ) 또는 지정된 다운로드 경로 이외의 입력값이 삽입되어 다운로드 될 수 없도록 문자 필터링 등을 통해 차단한다. 일반적으로 파일을 다운로드 하는 웹 스크립트 파일에 파일의 경로나 파일명을 통하여 다운로드 하는 것이 아니라 게시물의 번호와 게시판 번호에 대한 파일의 실제이름과 경로를 DB 항목에 만들어 놓고 게시물의 번호와 게시판 번호를 통하여 다운로드 받도록 구현 합니다.   ① 게시판, 결재, 메일(첨부파일)   ② 상위디렉토리, 절대경로 시스템파일 접근 불가 설정   예시 1). http://host/download.jsp?file=../../../../../../etc/passwd   예시 2). http://host/download.jsp?file=c:\boot.ini   증적제출) 별도의 증적은 없으며 파일 다운로드 기능 시 필터링 적용여부에 대해 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 16 | 소스코드 개발 보안 | B-07 | 오류로 인한 에러페이지에 대해 출력보호를 적용하였습니까? (Client가 비정상적인 요청으로 HTTP 오류를 발생시킬 경우)   설명) 응용프로그램의 에러처리를 부적절하게 설정하여 운영하는 경우, 서버정보, 디렉터리 혹은 DB 연결 정보 등의 시스템 정보가 노출되어 공격의, 단초가 되는 경우가 발생할 수 있습니다.   조치방법) 비정상적인 요청으로 HTTP 오류를 의도적으로 발생시킬 경우 출력되는 에러페이지는 별도의 에러페이지로 Redirect 하거나 적절한 에러 처리 루틴을 설정하여 처리하여야 합니다.   ① HTTP 400 / 401 / 403 / 404 / 500 HTTP Error 적용   Case 1) 오류 발생 시, 특정 웹페이지로 유도하도록 설정(Redirection)   ASP / PHP / JSP 파일에 에러페이지 지정 (예. <%@ page errorPage=”error.jsp" %>   Case 2) 오류 발생 시, 이전 웹페이지로 되돌아가도록 설정(Return)   증적제출) HTTP 상테 메시지(에러코드) 별 출력되는 페이지 화면을 캡쳐하여 증적으로 첨부하여 주시기 바랍니다.  PM의견) | O | ALL | 에러코드 별 오류페이지 화면 캡쳐 |
| 17 | 소스코드 개발 보안 | B-08 | 관리자 메뉴 페이지는 유추하기 어려운 디렉토리로 설정였습니까?  설명) 관리용 웹 페이지는 다양한 기능과 권한을 가지고 있어서 관리페이지의 접근 경로를 일반사용자가 알 수 없도록 설정하여 관리하여야 하며 강력한 인증을 구현하여 운영되어야 합니다.  조치방안) 관리용 웹 페이지의 접근경로를 admin, manager 등과 같이 추측하기 쉬운 디렉터리 명이나 파일 명을 사용하지 않도록 하고, 가능한 호스트 대역이나 IP만 접근 가능하도록 설정하여야 합니다. 추가로 관리자 로그인 시 권한 검증을 수행하여 관리자메뉴에 접근할 수 있도록 구현되어야 합니다.    Case 1) 로그인 시 권한 검증 후 자동으로 관리자메뉴 호출   Case 2) 관리자 메뉴 웹페이지가 별도로 존재 관리자 메뉴 금지 디렉토리   - http://admin.host.co.kr/   - http://manager.host.co.kr/   - http://system.host.co.kr/   - http://root.host.co.kr/  - http://www.host.co.kr/admin/   - http://www.host.co.kr/root/   - http://www.host.co.kr/system/   - http://www.host.co.kr/manager/  증적제출) 별도의 증적은 없으며 관리자 페이지의 접근경로 및 관리자 권한 관리는 어떻게 적용되어 있는지 PM의견란에 기술하여 주시기 바랍니다. | X | ALL |  |
| 18 | 소스코드 개발 보안 | B-09 | 응용프로그램을 구동하는 서버에는 공유폴더(shared folder) 사용을 금지하고 있습니까?  설명) 서버는 지침에 근거하여 공유폴더 사용이 금지되어 있습니다. 정보보안예외신청 승인을 득한 후 공유폴더를 사용하고 있다면, 보안설정 또는 패스워드 설정을 적용하여야 합니다.   조치방법) 공유폴더는 원칙적으로 아래와 같이 금지되어 있습니다. 부득히 사용이 필요한 경우 정보보안예외신청(1년 단위) 후 사용하여야 합니다.    ① 로컬 서버시스템에 자동으로 공유폴더 생성/사용 금지   ② 원격 서버시스템에 공유폴더를 이용한 파일전송 금지   증적제출) 서버 상 공유폴더의 설정 여부를 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | ALL | 공유폴더 현황 화면캡쳐 |
| 19 | 웹 서비스 보안 (C-01,05) | C-01 | 웹 서버 구축 시 기본 샘플 디렉터리 및 디폴트 매뉴얼 페이지는 삭제하였습니까?   설명) 기본적으로 샘플디렉터리 및 파일이 생성됩니다. 기본적으로 생성되는 샘플 디렉터리와 디폴트 매뉴얼 페이지에는 공개 또는 미공개 취약점이 존재합니다. 일반적으로 샘플디렉터리 및 파일은 응용프로그램 구동 시 대부분 사용하지 않으며, 불필요하기 때문에 삭제하여야 합니다.  조치방안) 웹 서버 구축 시 기본적으로 생성되는 샘플 디렉터리 및 디폴트 페이지는 삭제합니다.  증적제출) 별도의 증적은 없으며, 웹 서버 종류와 샘플디렉터리 및 매뉴얼 페이지 삭제 여부를 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 20 | 웹 서비스 보안 (C-02) | C-02 | 웹 서버 구축 시 상위 디렉터리 Path 접근을 불가하도록 설정하였습니까?  설명) 상위 경로 이동을 허용하도록 부적절하게 설정되어 있는 경우, 상대경로를 이용하여 Path 를 이동하면서 여러 취약점들이 발생할 수 있습니다.  조치방안) 웹서버 환경설정 또는 모듈에서 부모경로(또는 상위경로 이동)을 제한하도록 설정합니다.  증적 제출) 웹 서버 환경설정 부분의 상위 경로 이동을 제한하도록 설정한 부분을 화면 캡쳐하여 증적으로 첨부하시기 바랍니다.   PM의견) | O | ALL | 상위경로이동제한 설정 화면캡쳐 |
| 21 | 웹 서비스 보안 | C-03 | 웹 서비스 디렉토리를 절대경로로 접근 시, 디렉터리 내 파일 목록이 출력 되지 않도록 설정하였습니까?   설명) 디렉터리 리스팅은 일반적으로 디렉터리에 대해 직접 요청 시 그 디렉터리에 Directory Index(예: index.html)가 없을 경우, 디렉터리 내에 존재하는 모든 파일의 목록을 보여주는 것을 말합니다. 디렉터리 리스팅이 허용되면 외부에서 디렉터리 내의 모든 파일에 접근이 가능하여 백업 파일이나 소스파일 등 공개되어서는 안되는 중요한 파일들이 노출될 수 있습니다.  조치방안) 웹 서버의 환경설정에서 디렉터리 리스팅(Directory Listing)을 중지하도록 설정합니다.  증적제출) 디렉터리 리스팅를 중지하도록 설정한 부분을 화면캡쳐하여 증적으로 첨부하시기 바랍니다.  PM의견) | O | ALL | 디렉터리 리스팅 중지 설정 화면캡쳐 |
| 22 | 웹 서비스 보안 | C-04 | 웹 서버 및 응용프로그램에서 제공하는 기본 계정 / 암호는 변경 또는 삭제하여 사용하고 있습니까? (불필요한 계정인 경우 반드시 삭제)  설명) 웹 서버 또는 응용프로그램 설치시 기본적으로 생성되는 계정정보를 그대로 사용할 경우 비인가자가 쉽게 해당 시스템에 접근할 수 있습니다. 그러므로 admin/admin 등과 같은 추측 가능한 계정 및 root/root 과 같은 기본적으로 생성되는 계정정보를 그대로 사용하면 안되며, 사용하지 않을 시 반드시 사용중지 또는 삭제하여야 합니다.  조치방안) 설치시 기본적으로 생성되는 계정정보는 반드시 패스워드 변경 또는 불필요할 경우 삭제하여야 합니다.  증적제출) 별도의 증적은 없으며, 설치 시 기본 생성된 계정에 대해 패스워드 변경 또는 삭제 조치여부를 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 23 | 웹 서비스 보안 | C-05 | 웹 서버 및 응용프로그램에서 사용하지 않은 불필요한 HTTP Method는 사용 안함으로 조치하였습니까?  설명) HTTP 통신은 클라이언트가 웹 서버에 어떤 방식으로 어떤 페이지를 요청(Method)할지를 정하여 통신합니다. 불필요한 Mehtod를 허용하는 경우, 공격자는 해당 method를 통해 시스템 정보 및 권한을 획득할 수도 있습니다. 일반적으로 가장 많이 사용되는 Method는 HEAD, GET, POST 가 있으며, 그 외 Method들은 잘 사용하지 않기 때문에 사용하지 않도록 설정하는 것이 좋습니다.  조치방안) PUT, DELETE, COPY, MOVE, TRACE Method는 사용하지 않도록 웹서버에서 설정합니다.  증적제출) 불필요 HTTP Method를 제한하도록 설정한 부분을 화면캡쳐하여 증적으로 제출하여 주시기 바랍니다.  PM의견) | O | ALL |  |
| 24 | 웹 서비스 보안 | C-06 | 웹 서비스의 디렉토리 경로에 백업파일 및 불필요한 파일은 보관하지 않고 있습니까?  설명) 개발 과정에서 생긴 백업 파일은 웹 상에 쉽게 노출 될 수 있으며 .bak, .org, .old, .zip, .log 등 사용자 임의의 파일명을 사용하게 됩니다. 이러한 확장자는 웹 서버 설정에서 php, asp, jsp 등 해당 스크립트를 해석하도록 설정이 되어 있지 않은 경우가 많아 웹 상에서 소스가 그대로 노출 될 수 있어 악의적인 사용자에게 정보가 제공될 수 있습니다.  조치방안) 웹 디렉터리를 조사하여 다음과 같은 백업파일을 모두 삭제하고, \*.txt 같이 작업 중 생성된 일반 텍스트 파일이나 이미지 파일 등도 제거합니다.  - \*.bak, \*.new, \*.log, \*.org, \*.tmp, \*.sql, \*.zip, \*.bakup, \*.txt, \*.!, \*.old, \*.temp  증적제출) 별도의 증적은 없으며, 웹 서비스의 디렉터리에 불 필요 파일 존재 여부 확인 후 PM의견란에 확인 내용을 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 25 | 웹 서비스 보안 | C-07 | 웹서버 및 응용프로그램은 제공업체에서 권고하는 최신버전(패치) 적용하였습니까?  설명) 웹 서버 및 응용프로그램의 보안성 및 시스템 안정성을 확보하기 위해선 최신버전을 설치하거나, 제조사에서 제공하는 패치를 주기적으로 적용하여야 합니다.   조치방안) 제조사가 제공하는 최신버전 및 패치 확인 후 적용하시기 바랍니다.  증적제출) 별도의 증적은 없으며, 웹 서버의 최신 버전 및 패치 적용 여부를 확인하여 PM의견란에 확인된 내용을 기술하여 주시기 바랍니다.   PM의견) | X | ALL |  |
| 26 | 로깅 관리 | D-01 | 웹 서버 / 사용자 접속 로그는 다음의 항목이 저장되도록 설정하였습니까?  설명) 웹 서버 및 사용자 접속로그는 사후추적 및 감사를 위해 아래와 같은 항목이 저장될 수 있도록 설정하여 운영되어야 합니다.  조치방안) 웹 서버 로그 및 사용자 접속로그에 아래 항목이 저장될 수 있도록 설정하여야 합니다.  ① Server & Client 접속 IP Address, HTTP상태, Method, URI Stream   ② 사용자계정 login/ logout 일시   증적제출) 별도의 증적은 없으며, 웹 서버 로그 및 사용자 접속로그를 어떻게 설정하여 운영되고 있는지 확인한 후 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |
| 27 | 로깅 관리 | D-02 | 웹 서버 / 사용자 접속 로그는 최소 3개월 이상 보관하고 있습니까?  설명) 웹 서버 / 사용자 접속 로그는 사고 발생 시 추적 및 사실 등을 확인하기 위한 중요한 용도로 활용됩니다. 따라서 이러한 로그 데이터는 최소 3개월 이상 보관되도록 설정하여야 합니다.  조치방안) 생성된 로그는 최소 3개월 이상보관 될 수 있도록 해당 서버 외 다른 스토리지에 보관될 수 있도록 조치합니다. (스토리지 부족 시 미디어 보관 가능)  증적제출) 별도의 증적은 없으며, 로그 저장 및 보관주기에 대해 PM의견란에 기술하여 주시기 바랍니다.  PM의견) | X | ALL |  |