목차

1. 개요

1.1 목적

본 문서는 국방인사정보체계고도화의 개발표준정의서로서 개발방법 및 개발 표준을 제정하고 가이드를 제시하여 프로그램의 품질 향상과 일관성을 유지한다.
또한 여러 프로그램에서 공통적으로 사용되는 모듈을 컴포넌트화하여,
한번 작성된 컴포넌트의 재사용을 높인다.
이는 개발기간 단축 및 비용절감을 가져오고, 운영자 측면에서 효율적인 시스템 유지보수, 그리고 시스템 사용자측면에서 일관된 구성과 기능을 제공한다.
본 문서는 기록을 목적으로 한 문서가 아니라, 개발자들간의 이해와 공유를 위한 문서이므로, 개발자간에 본 문서의 내용을 이해하고, 적용하며 또한 내용 중 변경이 되어야 할 사항이나 추가로 공유되어야 할 사항이 있다면, 피드백을 통해 본 문서에 반영 될 수 있도록 한다.

2. 업무코드정의

2.1 시스템 코드

최상위 시스템코드를 정의.

시스템 명	시스템코드	전체이름
국방인사정보체계	DHRMIS	Defence Human Resource Millitary Information System

2.2 단위업무 코드

단위업무를 Level1, 2, 3 로 나눠 하위코드를 정의한다.

이 업무코드를 기준으로 Java Package 와 Websquare UI 디렉토리 구조를 정의한다.

시스템명	Level1	전체이름	약어	Level2	전체이름	약어	Level3	전체이름	약어
인사정보체계 (DHRMIS)	체계통합	System Integration	SIN	상호운영성	Iteroperability	IOP			
				기반체계	Infrastructure	IFR			
				체계연동	Data connection	DCO			
				데이타분석	Data Analysis	DAN			
				DW	Data Warehouse	DWH			
	인력관리	Manpower Management	MAN	인력획득	Sourcing	SOU			
				인력운영계획	Workforce Plan for Normal	NOR			
				전시인력 운영계획	Workforce Plan for War	WAE			
				초임분류	First	FST			
	인사관리	Human Management	HUM	보직관리	Assignment	ASG			
				평정관리	Appraisal	APR			
				진급관리	Promotion	PRO			
				기록관리	Person	PER			
				지원/심의관리	Resources Rreview Management	RES			
				장기/복무연장	Long-term Service	LNG			
				지휘추천	Command Recommended	REC			
				전역관리	Discharge Management	DCH			
	근무/행정	Work Administration	WAD	행정지원	Administration Support	ADS			
				병력일보관리	Military Force Ddaily Report Management	MFD			
	전시업무	Wartime Work	WAR	부대병력유지	Military Force Maintain	MAI			
				전사/순직관리	Death in Battle Management	DEA			
				전시병적관리	Wartime Military Register Management	REG			
				의무근무	Duty Work	DUT			
				군기군법	Military Discipline Law	LAW			
				전시인사 데이터 파일관리	War Human Resources affairs Data File Management	WDF			
	체계관리	System Management	SYS	권한관리	Authoity Management	AUT			
				조직관리	Organization Management	ORG			
				서비스관리	Service Management	SER			
				자료전환	Data Transformation	DTR			

3. 명명표준

3.1 기본 명명 Style

명명규칙에는 아래와 같이 Pascal Camel Case, Upper Case, Lower Case등 여러가지 종류가 있고, 본 프로젝트는 이들을 조합하여 복합적으로 사용한다.

가. Camel Case

의미있는 단어의 조합에서 각 단어 앞자리는 대문자, 나머지는 소문자로 표기.

첫자리는 소문자로 시작

ex) userList, myName

나. Pascal Case

첫자리를 대문자로 시작하는 Camel Case (aka Upper Camel Case)

ex) UserList, MyName

다. Lower Case

식별자의 모든 단어를 소문자로 조합한다.

ex) package mil.dhrmis.sin.iop.web;

라. Upper Case

식별자의 모든 단어를 대문자로 조합한다. 단어들간의 구분자는 underscore('_')로 구분한다.

ex) MY_NAME

마. Hungarian Notation

Data 유형을 Prefix 로 붙어놓은 것

ex) strUserName, mapUserList

바. Snake Notation

단어들간의 구분자를 unserscore('_')로 구분한다.

ex) my_name

3.2 본 프로젝트 프로그램 명명 규칙

모든 이름은 영문, 숫자의 조합 (a-z, A-Z, 0-9)을 원칙으로 한다.
숫자로 시작하지 않는다.
특수기호는 사용하지 않는 것을 원칙으로 한다.
과도한 약어의 사용은 피하고 정확한 의미 전달을 위해 되도록 FullName 을 사용한다.

가. 서버 프로그램 (Java) 명명규칙

(1) Package

- 모든 패키지는 Lower Case Style 로 명명한다.
- 각 Layer 는 아래와 같은 명명규칙으로 한다.
  - mil.dhrmis.{업무분류 Level1}.{업무분류 Level2}.{업무분류 Level3}. web, service, (vo), sqlmap

(2) Class

- Entity + ActionGroup + Layer 명 + .java 로 명명
- Entity 명
  - 업무주제명
- ActionGroup 명
  - 해당 Entity 를 가지고 처리할 Action 들을 아우르는 이름

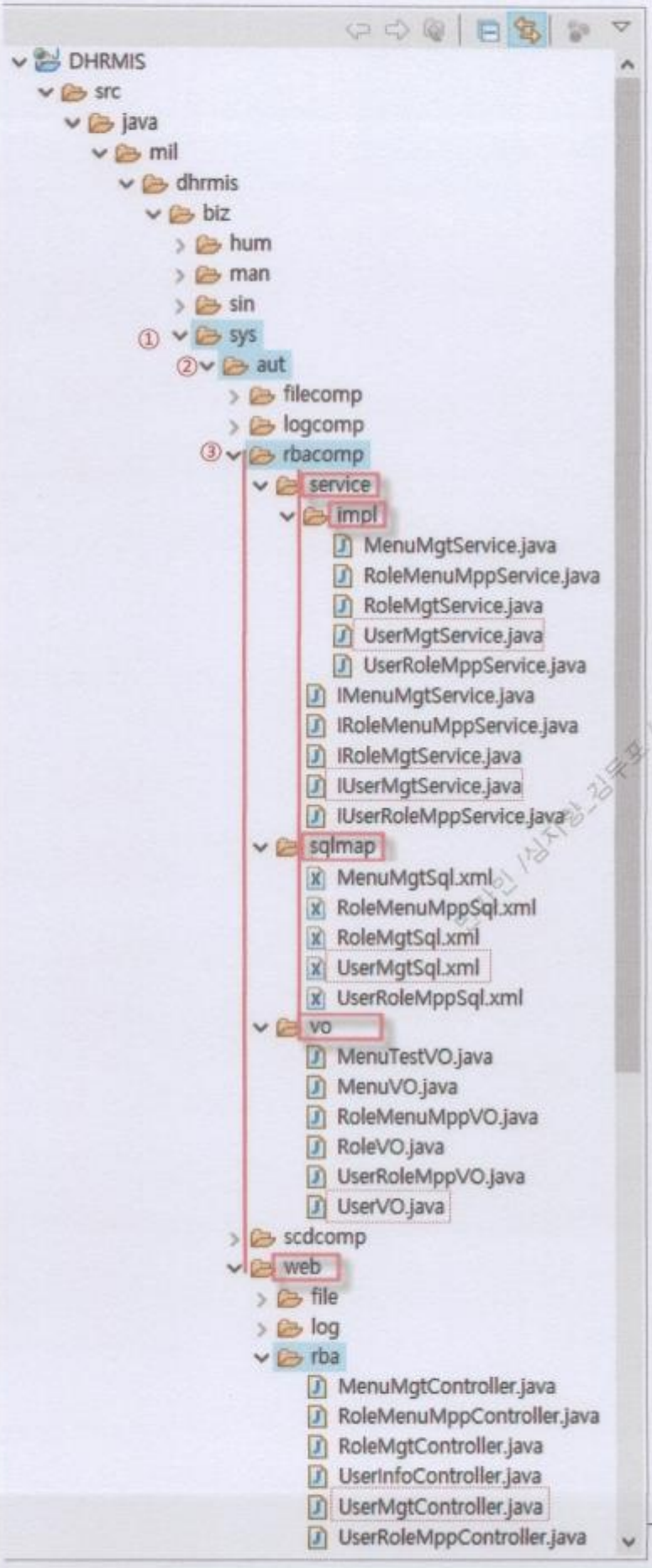
유형	영문명 Full Name	Action Group (약어)
조회	Inquery	Inqr
입력	Registration	Rgst
처리	Update	Updt
삭제	Delete	Delt
관리	Management	Mgt
처리	Process	Prcs
매핑	Mapping	Mpp
배치	Batch	Bat

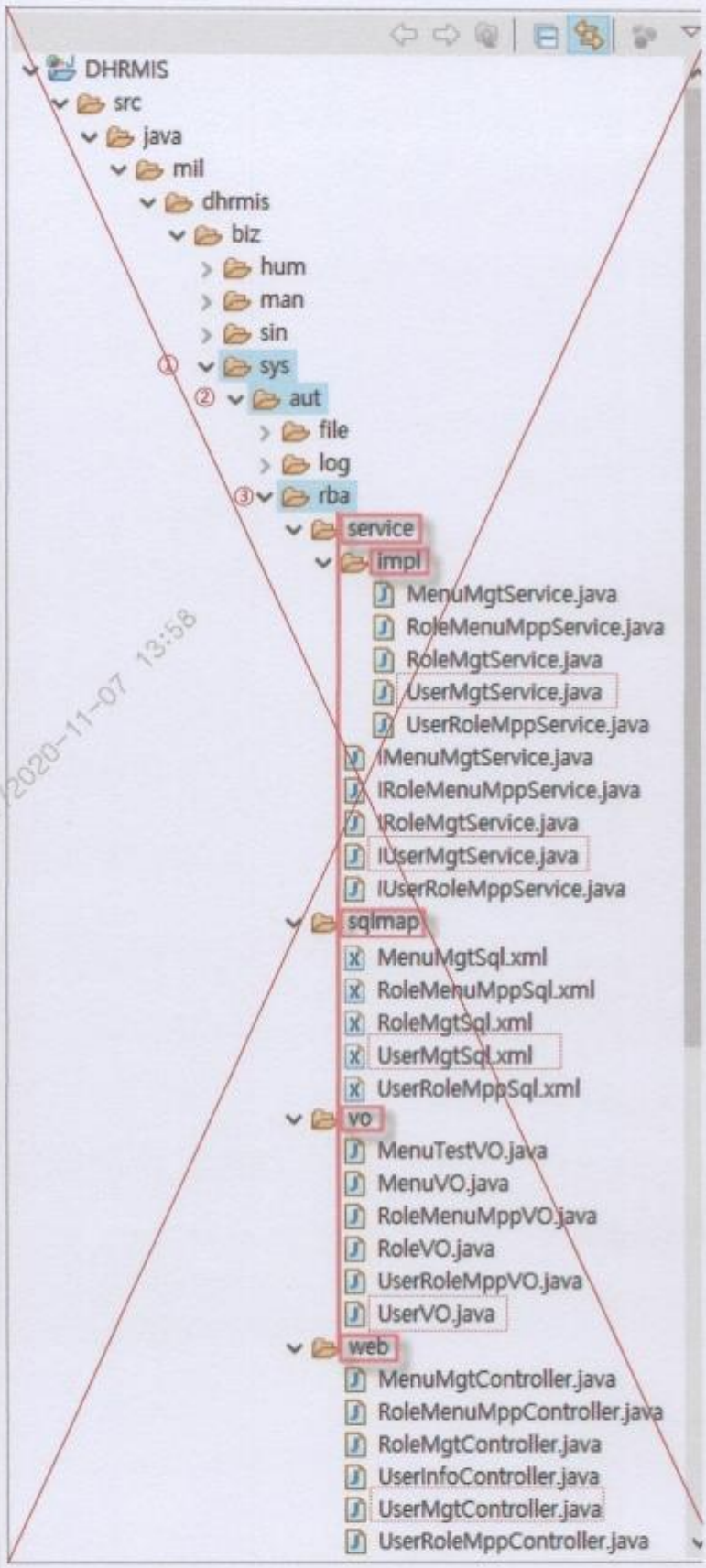
ex1) select, insert, update, delete 등 전체적인 관리를 위한 Action 들의 Group → Manage → Mgt

ex2) process 위주의 Action 들이 모인 Action 들의 Group → Process → Prcs

- Layer 명
  - ~Controller, ~Service, ~Sql

- 예제
  - EmpMgtController.java, EmpMgtService.java, EmpMgtSql.xml
  - FileGarbageCollectingPrcsController.java, FileGarbageCollectionPrcsService.java, FileGarbageCollectionSql.xml


DHRMIS
src
java
mil
dhrmis
biz
hum
man
sin
① sys
②aut
filecomp
logcomp
③rbacomp
service
impl
MenuMgtService.java
RoleMenuMppService.java
RoleMgtService.java
UserMgtService.java
UserRoleMppService.java
IMenuMgtService.java
IRoleMenuMppService.java
IRoleMgtService.java
IUserMgtService.java
IUserRoleMppService.java
sqlmap
MenuMgtSql.xml
RoleMenuMppSql.xml
RoleMgtSql.xml
UserMgtSql.xml
UserRoleMppSql.xml
vo
MenuTestVO.java
MenuVO.java
RoleMenuMppVO.java
RoleVO.java
UserRoleMppVO.java
UserVO.java
scdcomp
web
file
log
rba
MenuMgtController.java
RoleMenuMppController.java
RoleMgtController.java
UserInfoController.java
UserMgtController.java
UserRoleMppController.java


DHRMIS
src
java
mil
dhrmis
biz
hum
man
sin
① sys
② aut
file
log
③rba
service
impl
MenuMgtService.java
RoleMenuMppService.java
RoleMgtService.java
UserMgtService.java
UserRoleMppService.java
IMenuMgtService.java
IRoleMenuMppService.java
IRoleMgtService.java
IUserMgtService.java
IUserRoleMppService.java
sqlmap
MenuMgtSql.xml
RoleMenuMppSql.xml
RoleMgtSql.xml
UserMgtSql.xml
UserRoleMppSql.xml
vo
MenuTestVO.java
MenuVO.java
RoleMenuMppVO.java
RoleVO.java
UserRoleMppVO.java
UserVO.java
web
MenuMgtController.java
RoleMenuMppController.java
RoleMgtController.java
UserInfoController.java
UserMgtController.java
UserRoleMppController.java

(3) Method

- 첫문자를 소문자로 시작하고, 두 단어 이상의 조합일 경우 Camel Case 명명규칙을 사용한다.
  ex) selectUserList()
- Method 명 내에는 구분자("_" 등)를 사용하지 않으며, Method 의 행위를 충분히 설명할 수 있는 의미있는 단어를 사용한다.
- 한번에 여러 개의 목록을 받거나 리턴 하는 경우에는 Method 명의 마지막에 List 를 붙인다.
  ex) seletUser() vs selectUserList(), saveUser() vs saveUserList()
- Method 의 명명은 기본셋을 정해 사용.

작업 내용	동사	예제
데이터 조회 (목록)	select	selectUser(List)
데이터 입력 (목록)	insert	insertUser(List)
데이터 수정 (목록)	update	updateUser(List)
데이터 삭제 (목록)	delete	deleteUser(List)
입력, 수정, 삭제 동시 처리	save	saveUser(List)
처리	process	processDayClosing
배치	batch	batchDailyTestJob

- Method 명은 일관성있게 명명해야 함.
  ∵ Intercept나 AOP를 걸 때 메소드명이 일관적이어야 공통관심사항(AOP)들을 주입하는데 문제가 없음. 추가적으로 필요한 Method명은 일단 작성해서 Commit해주세요.
  검토후 취합하겠습니다.

- 예제

```
@RequestMapping("/rba/selectUserList.do")
public void selectUserList(MciRequest req, Model model) throws Exception
{
    // get parameter
    Map<String, Object> params = req.getParameters();

    // invoke service method
    List<Map<String, Object>> list = userManageService.selectUserList(params);

    // response
    model.addAttribute("userList", list);
}
```

(4) Variable

- 첫문자는 소문자로 시작하고, 두 단어 이상의 조합일 경우 Camel Case 명명규칙을 따른다.
  ex) userList
- 해당 변수의 의미를 명확히 하기 위해 접두어(Prefix)로 그 속성을 명시한다. (Hungarian 표기법)
  ex) mapUser, lstUser, objUser 정도

(5) Attribute

- 첫문자는 소문자로 시작하고, 두 단어 이상의 조합일 경우 Camel Case 명명규칙을 따른다.
  ex) model.addAttribute("userList", lstUseList);

(6) Constant

- 모두 Upper Case 를 사용하고, 단어와 단어사이는 "_"(Underbar)로 구분한다.
  ex) ERROR_MESSAGE

나. UI 프로그램 (Websquare) 명명규칙

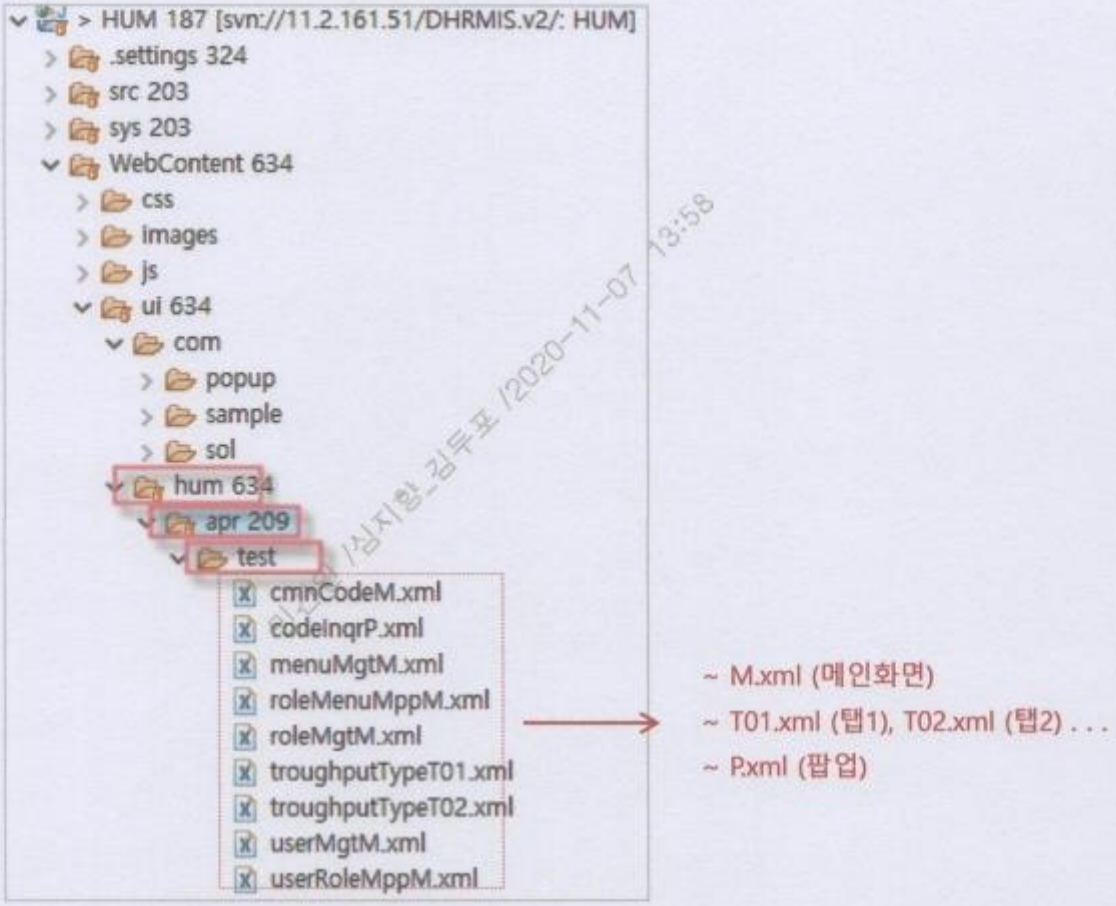
(1) Directory

- 모든 디렉토리는 소문자(Lower Case Style)로 명명한다.
- 각 디렉토리는 아래와 같은 명명규칙으로 한다.
  - ui / 업무분류Level1 / 업무분류Level2 / 업무분류Level3 / 화면XML

(2) UI

- Entity + ActionGroup + .xml 로 명명

- 예제



(3) Variable

- 첫문자는 소문자로 시작하고, 두 단어 이상의 조합일 경우 Camel Case 명명규칙을 따른다.
  ex) userList
- 해당 변수의 의미를 명확히 하기 위해 접두어(Prefix)로 그 속성을 명시한다. (Hungarian 표기법)

컨트롤타입	접두어	예제
Button	btn	btnSearch
Label	lbl	lblTitle
TextBox	div	divTitle
Combobox (Selectbox)	cmb	cmbMenuLevel
Radio	rdo	rdoSex
InputBox	txt	txtUserName
TextArea	txa	txaContent
Grid	grd	grdUser
TreeView	trv	trvMenu
DataCollection	dl	dlUserList
DataMap	dm	dmParam
Checkbox	chk	chkCode
InputCalendar	ica	icaDueDay
Chart	cht	chtSale
Generator	gen	genMenu
PageList	pgl	pglBoard
TabControl	tab	tabDetail
WFrame	wfm	wfmBody
Span	spa	spaTitle
Division	div	divFile

(4) Function

- 첫문자는 소문자로 시작하고, 두 단어 이상의 조합일 경우 Camel Case 명명규칙을 따른다.
- 변수와 함수를 구분하기 위해 함수 앞에는 fn 을 붙인다.
  ex) fnSearchUserList()
- Function 명 내에는 구분자("_" 등)를 사용하지 않으며, Function 의 행위를 충분히 설명할 수 있는 의미있는 단어를 사용한다.
  단, 이벤트 함수는 "_" (under bat)를 허용한다.
  ∵ 자동으로 생성되는 Event함수들은 자동으로 _("언더바") 가 붙음.
  ex) btnSearch_onclick

3.3 기본 주석 Style

주석은 블럭 주석, 단일라인 주석, 라인 주석, Markup 주석등 여러 종류가 있고, 본 프로젝트는 이들을 조합하여 복합적으로 사용한다.

가. 블럭 주석

```
/**      /*
 *       *
 */  또는 */    (문서주석 vs 구현주석)
```

블럭 주석문은 클래스나 메소드 등을 설명하기 위해 주로 사용한다.

Java API Documentation 에 넣을 설명은 ex1 과 같이 처리해야 나중에 문서 Generation 시 발췌된다. 자체 참조를 위해선 ex2 와 같이 처리

ex1)

```
/**
 * XX 데이타 저장결과 조회
 *
 * @author dp.kim
 * @since 2020.05.29
 * @param MciRequest 저장할 데이타
 * @param Model 저장 결과
 * @return
 * @throws Exception If an exception occured
 */
```

ex2)

```
/*
 * TODO
 */
```


나. 단일라인 주석

```
/*  */
```

한줄 주석.

Indent 는 해당코드와 같은 Depth 로 맞춘다.

ex)

```
if (condition)
{
    /* Handle the condition. */
    insertTest();
}
```

다. 라인 주석

//

한줄 주석.

라인 전체 또는 라인의 일부분을 주석처리 할 때 사용.

ex)

```
if (foo > 1)
{
    // Do a double-flip.
    ...
}
Else
{
    return false; // Explain why here.
}
```

라. MarkUp Language 주석

<!-- -->

HTML 및 XML 에서 사용하는 주석으로, Markup 언어 내에서 사용할 수 없는 꺽쇠문자 (<>) 는 <, > 를 사용한다.

Indent 는 해당코드와 같은 Depth 로 맞춘다.

ex)

```
<!—사용자 정보 조회-->
<select id="selectUserList" parameterType="map">
    /* 사용자 정보 조회 */
    ...
</select>
```

3.4 본 프로젝트 주석 명명 규칙

모든 소스는 Java API Documentation 생성에 부합하도록 주석문을 작성한다.
(참고: http://www.oracle.com/technetwork/java/codeconvtoc-136057.html > 5 - Comments)

가. 서버 프로그램 (java) 주석 명명규칙

(1) Class

- Class 선언문 전에 위치하고 주석 형식은 문서형 블럭주석(/** ... */)을 사용한다.
- 클래스 설명은 가능하면 5, 6 줄 정도로 자세히 기술한다.
- 클래스에 대한 설명, 작성자명, 생성일자 등을 기술한다.
- 예제

```
/**
 * 해당 시스템에 접속하는  전체 사용자를 관리하는 컴포넌트로
 * 본 컴포넌트는  요청 URL과 Servlet을 Mapping해주고, 그 처리결과를 응답해주는
 * Class 이다.
 *
 * @author dp.kim
 * @since 2020.05.29
 */
@Controller
public class UserMgtController
```


(2) Method

- Method 선언문 전에 위치하고 주석 형식은 문서형 블럭주석(/** ... */)을 사용한다.
- 메소드 기능 설명은 한두 줄로 간결하게 기술한다.
- 메소드에 대한 설명, Parameter, Return Type, 작성자명과 생성일자, 예외 등을 기술한다.
- 값이 없는 경우 빈값으로 두지말고 아예 삭제한다.
- 예제

```
/**
 * 시스템에 접속가능한 사용자 목록을 조회.
 *
 * @author dp.kim
 * @since 2020.05.29
 * @param MciRequest 조회 조건
 * @param Model 조회 결과
 * @return
 * @throws Exception If an exception occured
 */
@RequestMapping("/rba/selectUserList.do")
public void selectUserList(MciRequest req, Model model) throws Exception
```

(3) SQL

```
<!-- 사용자정보 조회 -->
<select id="selectUserList" parameterType="map" resultType="egovMap">
      /* com.rbacomp.UserMgt.selectUserList */ WAS에서 Log를 추적할수 있게 하기 위해 추가
       select T1.emplyr_id,
              T1.orgnzt_id
              . . .
         from COMTNEMPLYRINFO T1
        where 1 = 1
</select>
```

나. UI 프로그램 (Websquare) 주석 명명 규칙

(1) UI

화면에 대한 메뉴경로, 화면명, 설명, 작성자명, 생성일자 등을 기술한다.

```
/*-------------------------------------------------------------------------------------
  메뉴경로     :    공통-사용자-사용자관리
  화면이름     :    사용자관리
  화면설명     :    사용자 관리
  작성자       :    dp.kim
  작성일       :    2018.07.24
-------------------------------------------------------------------------------------*/
```

(2) Function

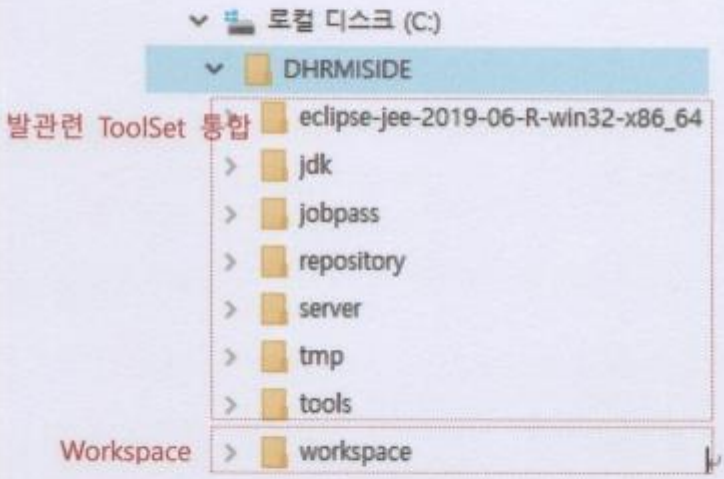
Function 에 대한 설명, Parameter, Return Type, 작성자명과 생성일자, 예제 등을 기술한다.

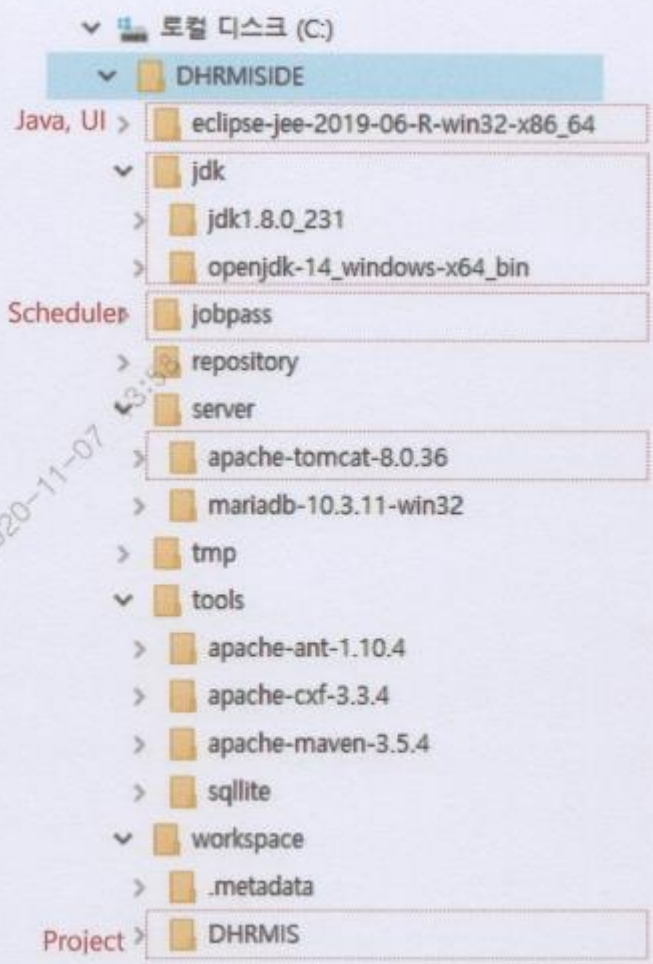
```
/**
* main.xml의 컨텐츠영역의 화면을 이동시킨다.
*
* @since 2020.06.10
* @author 전기완
* @param {String} psTitle 화면타이틀
* @param {String} psSrc 화면 src
* @param {JSON} poParam 화면에 넘길 JSON파라메터
* @return {void}
* @example
* 메뉴가 목록화면일때 등록화면으로 이동시
* com.changeMenu("프로그램등록", "/ui/sample/programRegister.xml",
{"AAA":"aaa", "BBB":"bbb"});
*
* 등록화면에서 목록화면으로 이동시
* com.changeMenu(null, null, {"DDD":"ddd", "CCC":"ccc"});
*/
com.changeMenu = function(psTitle, psSrc, poParam)
```

4. 통합개발환경

4.1 개요

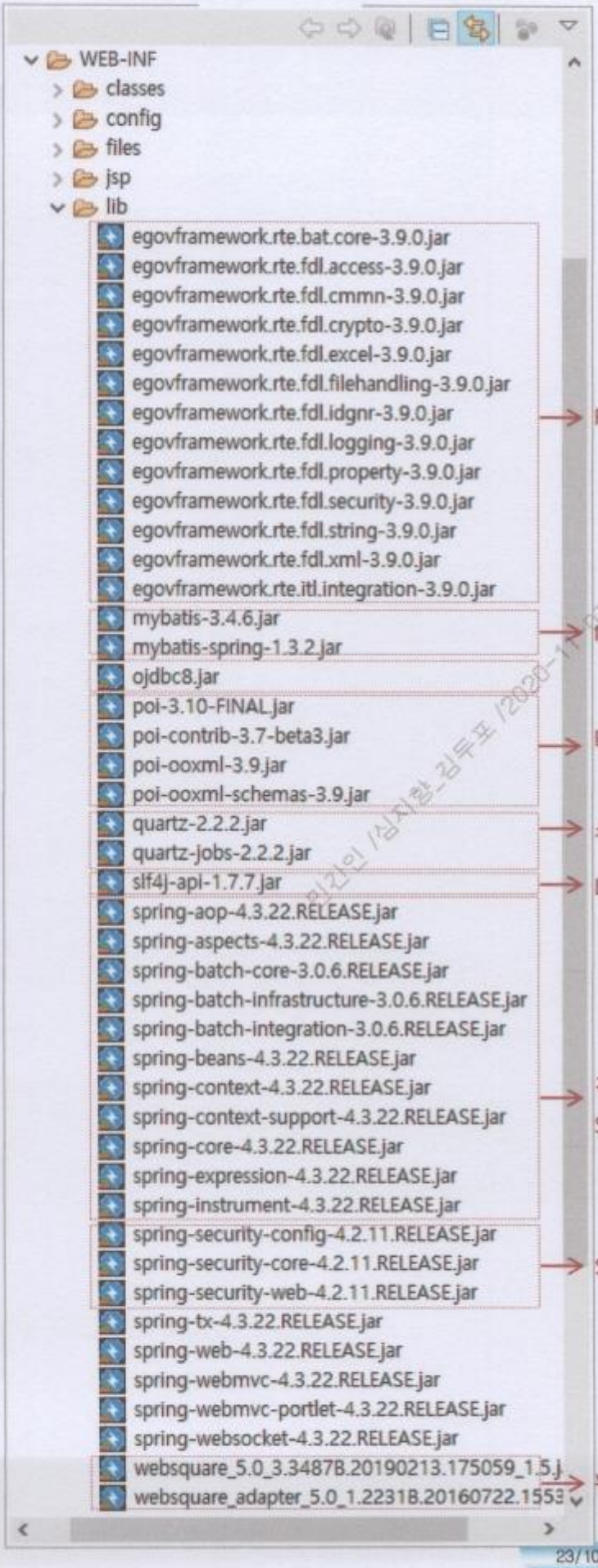
DHRMISIDE.7z 으로 제공되고, 압축을 풀면 해당 디렉토리 안에 개발과 관련된 Jdk, Eclipse, Tomcat, Ant, Maven, Repository, Workspace 등이 모여있음.





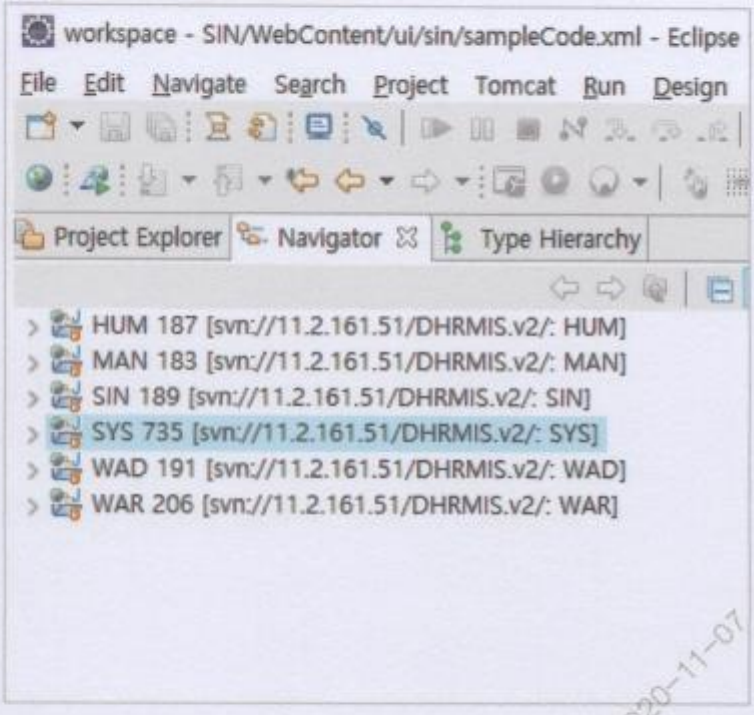
※ Eclipse 2019.06 (Eclipse v4.12) - Eclipse Photon (Eclipse v4.8) 이후 연월로 명명
Open JDK 1.8 (Redhat), Oracle JDK 1.8
Subversion- Polarion Suversive Team Provider vs ~~Tigris Subclipse~~
OZ 는 개별설치해서 개발

4.2 WebApplication Libraries


WEB-INF
classes
config
files
jsp
lib
egovframework.rte.bat.core-3.9.0.jar
egovframework.rte.fdl.access-3.9.0.jar
egovframework.rte.fdl.cmmn-3.9.0.jar
egovframework.rte.fdl.crypto-3.9.0.jar
egovframework.rte.fdl.excel-3.9.0.jar
egovframework.rte.fdl.filehandling-3.9.0.jar
egovframework.rte.fdl.idgnr-3.9.0.jar
egovframework.rte.fdl.logging-3.9.0.jar
egovframework.rte.fdl.property-3.9.0.jar
egovframework.rte.fdl.security-3.9.0.jar
egovframework.rte.fdl.string-3.9.0.jar
egovframework.rte.fdl.xml-3.9.0.jar
egovframework.rte.itl.integration-3.9.0.jar
mybatis-3.4.6.jar
mybatis-spring-1.3.2.jar
ojdbc8.jar
poi-3.10-FINAL.jar
poi-contrib-3.7-beta3.jar
poi-ooxml-3.9.jar
poi-ooxml-schemas-3.9.jar
quartz-2.2.2.jar
quartz-jobs-2.2.2.jar
slf4j-api-1.7.7.jar
spring-aop-4.3.22.RELEASE.jar
spring-aspects-4.3.22.RELEASE.jar
spring-batch-core-3.0.6.RELEASE.jar
spring-batch-infrastructure-3.0.6.RELEASE.jar
spring-batch-integration-3.0.6.RELEASE.jar
spring-beans-4.3.22.RELEASE.jar
spring-context-4.3.22.RELEASE.jar
spring-context-support-4.3.22.RELEASE.jar
spring-core-4.3.22.RELEASE.jar
spring-expression-4.3.22.RELEASE.jar
spring-instrument-4.3.22.RELEASE.jar
spring-security-config-4.2.11.RELEASE.jar
spring-security-core-4.2.11.RELEASE.jar
spring-security-web-4.2.11.RELEASE.jar
spring-tx-4.3.22.RELEASE.jar
spring-web-4.3.22.RELEASE.jar
spring-webmvc-4.3.22.RELEASE.jar
spring-webmvc-portlet-4.3.22.RELEASE.jar
spring-websocket-4.3.22.RELEASE.jar
websquare_5.0_3.3487B.20190213.175059_1.5.j
websquare_adapter_5.0_1.2231B.20160722.1553
RFP에 최신 전자정부프레임워크 → 3.9
Mybatis 3.4.6
Excel Library
스케줄러 (JobPass 지원)
Logging
전자정부프레임워크 3.9 Dependency의
SpringFramework 4.3.22
SpringSecurity 4.2.11
WebSquare 공통컴포넌트, Adapter

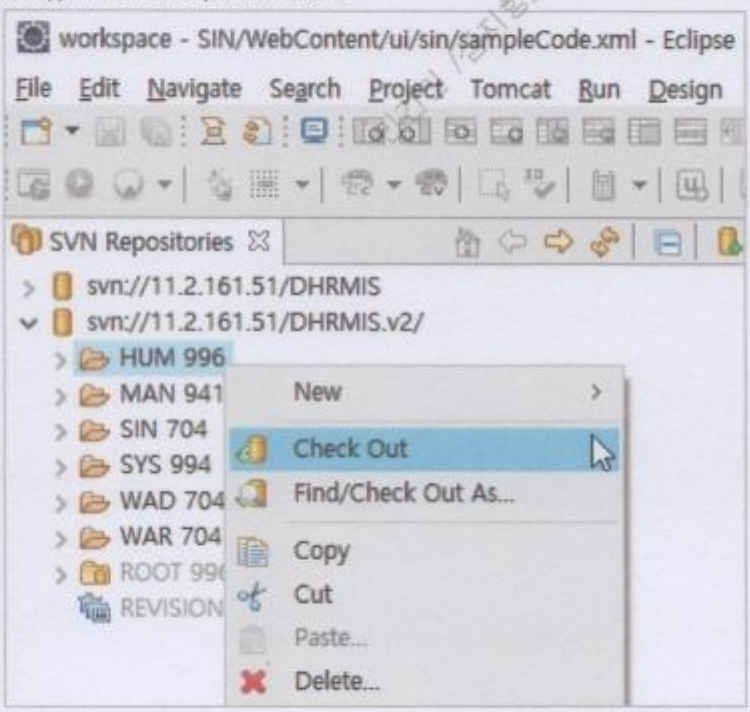
4.3 Workspace

1. 파트별 프로젝트가 분리

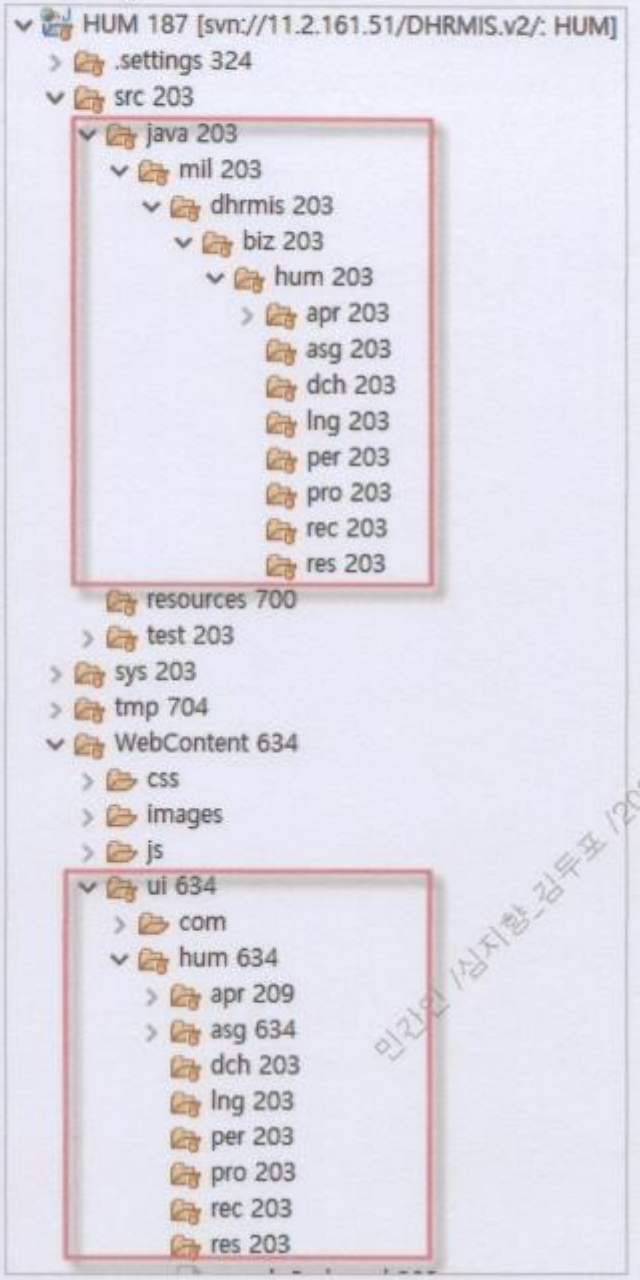


2. 아래 주소에서 해당 프로젝트와 SYS 프로젝트 CheckOut

svn://11.2.161.51/DHRMIS.v2

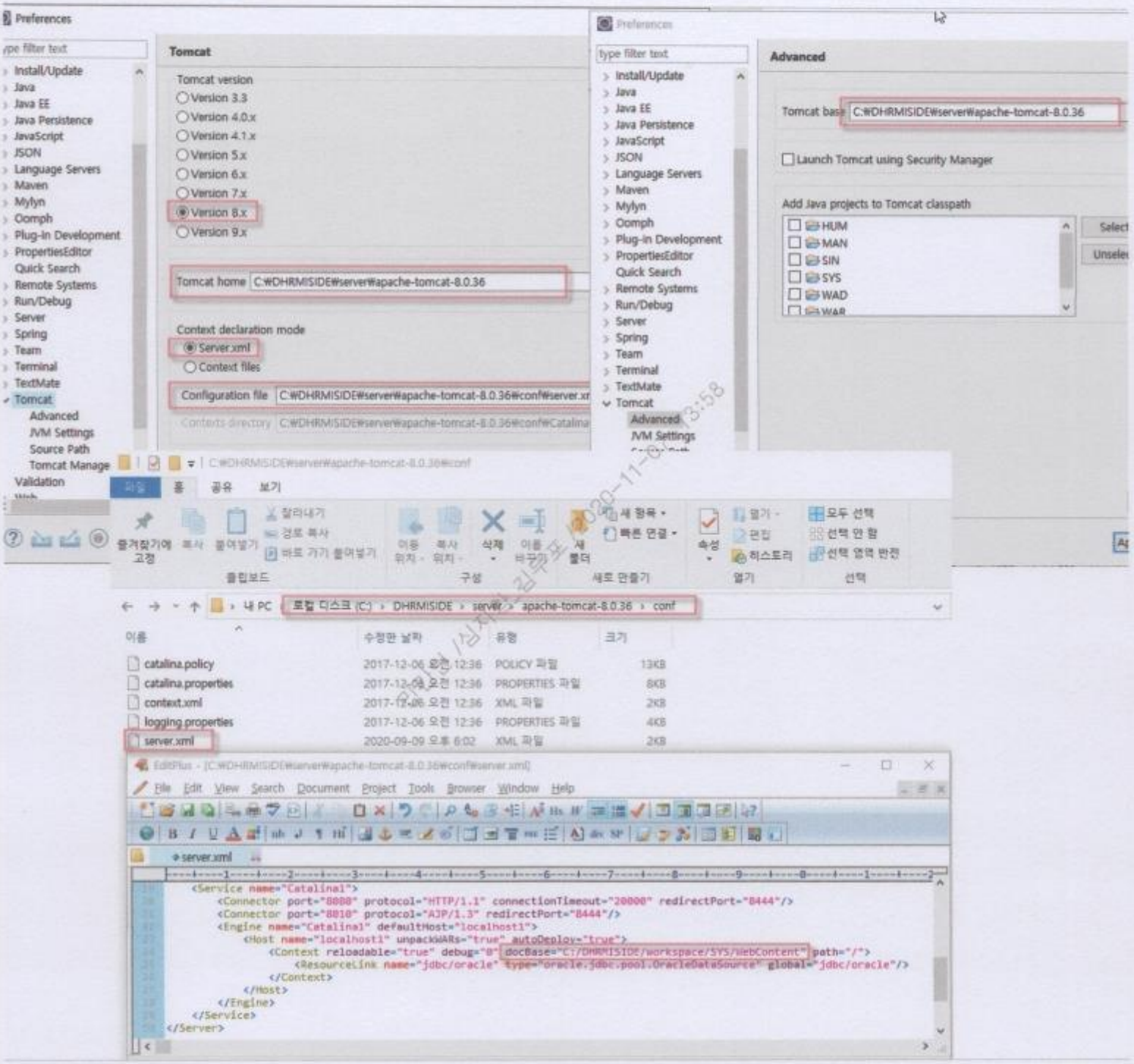


3. 업무별 java 소스 작성 폴더와, ui 작성폴더 확인



5. 서버구동 설정

- Eclipse 톰캣 Plugin 설정
- Tomcat 서버 설정



※ 서버 구동시 자신 프로젝트만 띄우고자 하면 dispacher-servlet.xml, context-common.xml scan범위 수정

5. Framework

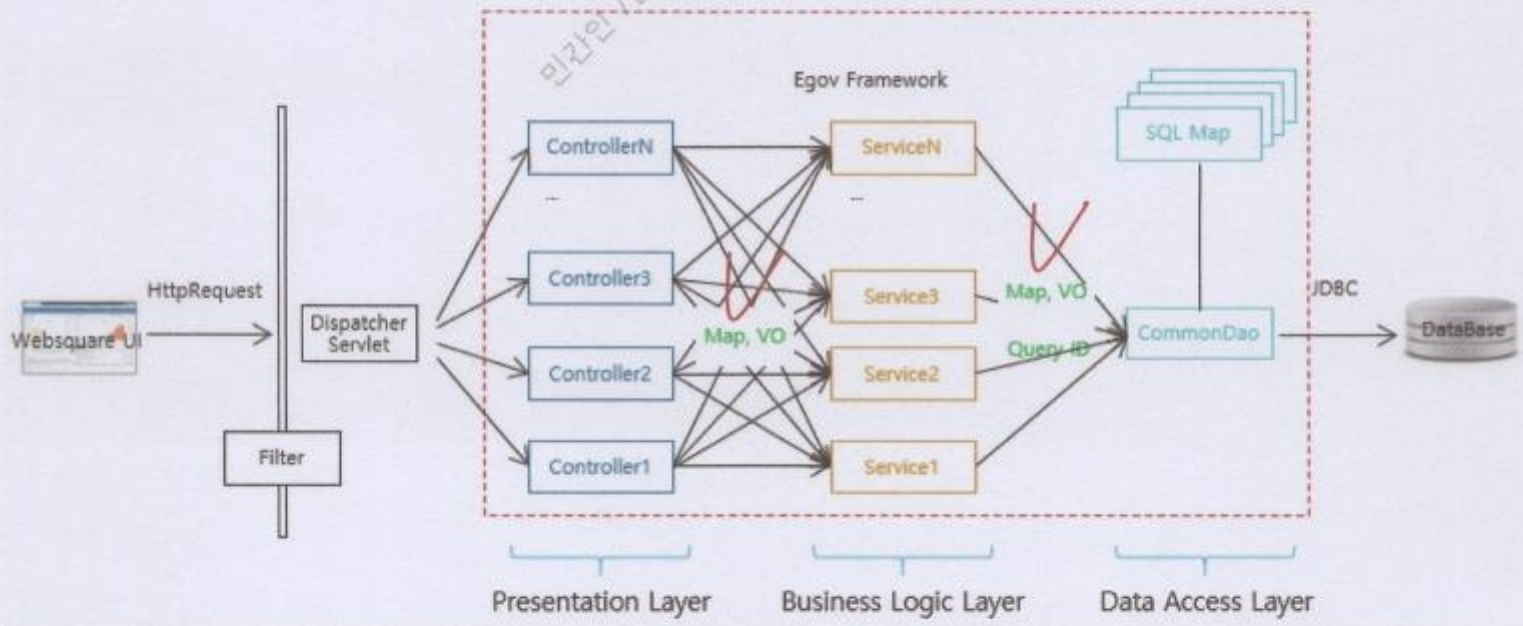
그림 1

<< System View >>

그림 2

<< Software View >>

그림 3



<< Framework >>

6. 코딩 가이드

6.1 서버 프로그램 (java) 개발

- 해당 프로젝트에서는 Controller, Service, SqlMap 만 작성하고 Dao 는 CommonDao 를 사용한다.

가. Controller

(1) 개요

Controller는 화면처리를 담당하는 구현체로, 단순히 UI로부터 넘겨받은 Parameter로 Service를 호출하고, 결과를 다시 화면에 전송하는 역할을 한다.

(2) 작성요령

■ 개발자가 작성해야 할 Package, Class 명은
mil.dhrmis.{업무분류Level1}.{업무분류Level2}.web.{업무분류Level3}.{Entity + ActionGroup}.Controller.java

■ 등록을 위해 별도의 환경설정파일을 필요로 하지 않는다.

■ Spring Framework 의 Bean 으로 등록을 위해 @Controller 라는 어노테이션을 Implementation Class 앞에 붙여준다.

■ @RequestMapping 을 이용해 호출경로를 지정한다. Class 에 붙어있는 @RequestMapping 경로와 Method 에 붙어있는 @RequestMapping 경로가 조합되어 URL 이 만들어 진다.

■ 특별한 예외사항이 없는 경우 Controller 의 파라미터는 MciRequest 와 Model 두개의 파라미터로 처리한다.

ex) public void selectUserList(MciRequest req, Model model)

※ MciRequest는 Grid Data Request를 받기 위한 Websquare UI Adapter를 Wrapping한 Class로 표준웹 파라미터를 받는 getParameter(String paramName), getParameterValues와 같은 메소드들을 기본으로 제공하면서, 추가로
getGridMap(String datasetName)
getGridVO(String datasetName)
등과 같은 그리드 메소드를 제공한다.

※ Model은 Service로부터 받은 결과데이터를 UI에 내려줄 때 사용된다.

■ 예외처리는 무조건 메소드 밖으로 Throw 시킨다. 프레임워크에서 처리.
오류 메시지의 변경을 위해서 try ~ catch 구문 사용은 허용하나 여전히 NestedExcetion 은 밖으로 Throw 시킨다.

※ 유의사항
Controller에서 Service에 처리를 위임할 때 Service Method는 반드시 하나만 호출한다.
만약, 두개 이상의 Service Method를 호출하고 싶을 경우 두개의 Service Method를 하나로 묶는 별도의 Service Method 만들어 호출한다.

(3) 예제

```java
/**
 * UserMgtController Provides infrastructure for determining view names from URLs
 *
 * <p>
 * 자세한 설명
 * @author dp.kim
 * @since 2020.05.29
 */
@Controller
public class UserMgtController
{
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private UserMgtService userMgtService;

    /**
     * Returns User Information. Request must specified
     *
     * <p>
     * 시스템에 접속가능한 사용자 목록을 조회.
     * @author dp.kim
     * @since 2020.05.29
     * @param MciRequest 조회 조건
     * @param Model 조회 결과
     * @return
     * @throws Exception If an exception occured
     */
    @RequestMapping("/rba/selectUserList.do")
    public void selectUserList(MciRequest req, Model model) throws Exception
    {
        // get parameter
        Map<String, Object> params = req.getParameters();

        logger.debug("..." + params.toString());

        // invoke service method
        List<Map<String, Object>> lists = userMgtService.selectUserList(params);

        // response
        model.addAttribute("userList", lists);
    }

    /**
     * Returns Result Information for Saving. Request must specified
     *
     * <p>
     * 시스템에 접속가능한 사용자 목록을 저장.
     * @author dp.kim
     * @since 2020.05.29
     * @param MciRequest 저장할 데이타
     * @param Model 저장 결과
     * @return
     * @throws Exception If an exception occured
     */
    @RequestMapping("/rba/saveUserList.do")
    public void saveUserList(MciRequest req, Model model) throws Exception
    {
        // get parameter
        List<Map<String, Object>> userList = (List<Map<String, Object>>) req.getGridMap("userList");

        // invoke service method
        userMgtService.saveUserList(userList);
    }
}
```

단순히 Request 받고, Service 호출하고, Response

※ Autowired vs Resource

- Autowired 는 Spring 전용 Annotation 이고, Resource 는 JSR-250 에 따른 JDK Standard Annotation.
- Resource 를 사용하려면 ("name="userMgtService")와 같은 속성을 더 지정해줘야 하는 조금 더 귀찮음이 존재. (Type 으로 찾냐 Name 으로 찾냐의 차이)

나. Service

(1) 개요

- Service Class는 업무로직이 쓰여지는 구현체로, Controller로부터 넘겨받은 Parameter를 Mybatis Query Id와 함께 CommonDao를 호출하고, 결과를 다시 Controller에게 반환한다.
- Transaction의 단위가 된다.

(2) 작성요령

■ 개발자가 작성해야 할 Package, Class 는 Interface 와 Implementation 2 개.로 아래와 같은 명명규칙을 갖는다.

mil.dhrmis.{업무분류Level1}.{업무분류Level2}.{업무분류Level3+comp}.service.I + {Entity + ActionGroup}.Service.java

mil.dhrmis.{업무분류Level1}.{업무분류Level2}.{업무분류Level3+comp}.service.impl.{Entity + ActionGroup}.Service.java

■ 등록을 위해 별도의 환경설정파일을 필요로 하지 않는다.

■ Spring Framework 의 Bean 으로 등록을 위해 @Service 라는 어노테이션을 Implementation Class 앞에 붙여준다.

■ 전자정부 표준 프레임워크와의 호환성을 위해 EgovAbstractServiceImpl 을 extends 한다.

■ 예외처리는 무조건 메소드 밖으로 Throw 시킨다. 프레임워크에서 처리.
오류 메시지의 변경을 위해서 try ~ catch 구문 사용은 허용하나 여전히 NestedExcetion 은 밖으로 Throw 시킨다.

(3) 예제

1. Interface

```
public interface IUserMgtService
{
    public Map<String, Object> loadUserByUserName(String id, String pw);

    public void saveUserList(List<Object> lstData);
}
```

2. Implementation

```
/**
 * UserMgtService is a concrete implementation of the abstract EgovAbstractServiceImpl class
 *
 * <p>
 * 자세한 설명
 * @author dp.kim
 * @since 2020.05.29
 */
@Service
public class UserMgtServiceImpl extends EgovAbstractServiceImpl implements IUserMgtService
{
    @Autowired
    private CommonDao dao;

    /**
     * Returns User Information. Request must specified
     *
     * <p>
     * 자세한 설명
     * @author dp.kim
     * @since 2020.05.29
     * @param String 조회 조건
     * @return Map 조회 결과
     */
    public Map<String, Object> loadUserByUserName(String id)
    {
        return dao.queryForObject("UserMgtSql.selectUserById", param);
    }

    /**
     * Returns Result Information for Saving. Request must specified
     *
     * <p>
     * 자세한 설명
     * @author dp.kim
     * @since 2020.05.29
     * @param List 저장할 데이타
     * @return
     */
    @Transactional(propagation = Propagation.REQUIRED)
    public void saveUserList(List<Object> lstData)
    {
        for(int i = 0; i < lstData.size(); i++)
        {
            Map<String, Object> userData = (Map<String, Object>) lstData.get(i);
            String rowStatus = userData.getRowStatus();

            if(rowStatus.equals(RowStatus.INSERT))
            {
                dao.insert("UserMgtSql.insertUser", userData);
            }
            else if(rowStatus.equals(RowStatus.UPDATE))
            {
                dao.update("UserMgtSql.updateUser", userData);
            }
            else if(rowStatus.equals(RowStatus.DELETE))
            {
                dao.delete("UserMgtSql.deleteUser", userData);

            }
        }
    }
}
```

다. CommonDAO

(1) 개요

전자정부 표준프레임워크와의 호환성을 위해 EgovAbstractMapper를 extends 하였고,

단순 DataBase 조회, 입력, 수정, 삭제 이외에 File Upload, Download등의 부가기능을 제공한다.

(2) API

CommonDao의 Method는 기본적인 DataBase CRUD와 관계된

queryForList

queryForObject

queryForString

queryForInteger

insert

update

delete

batchInsert

batchUpdate 가 있고,

Query수행후 File을 Octet-stream으로 바로 다운로드 받을수 있게

downloadCsvFile, downloadXlsFile, downloadXlsxFile 메소드 외

upload와 관련한 메소드도 csv, xls, xlsx관련 3가지 메소드를 제공한다. 아래 "사. File처리" 에서 설명

CommonDao APIs	설명
List queryForList(String statementName)	statementName으로 List형태의 다건 데이터를 조회한다.
List queryForList(String statementName, Object parameterObject)	statementName과 parameterObject로 List형태의 다건 데이터를 조회한다.
Object queryForObject(String statementName)	statementName으로 Object형태의 단건 데이터를 조회한다.
Object queryForObject(String statementName, Object parameterObject)	statementName과 parameterObject로 Object형태의 단건 데이터를 조회한다.
int insert(String statementName)	statementName으로 데이터를 입력한다.
int insert(String statementName, Object parameterObject)	statementName과 parameterObject로 데이터를 입력한다. insert 후에 paramObject 로부터 다시 getter를 통해 PK를 받을 수 있다.
int update(String statementName)	statementName으로 데이터를 수정한다. 리턴되는 값은 수정결과 변경된 Row Count가 넘어온다.
int update(String statementName, Object parameterObject)	statementName과 parameterObject로 데이터를 수정한다. 리턴되는 값은 수정결과 변경된 Row Count가 넘어온다.
int delete(String statementName)	statementName으로 데이터를 삭제한다. 리턴되는 값은 삭제결과 변경된 Row Count가 넘어온다.
int delete(String statementName, Object parameterObject)	statementName과 parameterObject로 데이터를 삭제한다. 리턴되는 값은 삭제결과 변경된 Row Count가 넘어온다.
void downloadCsvFile(String statementName, Map param, String fileName, HttpServletResponse res)	statementName과 param으로 조회한 다건의 데이터를 Csv파일로 스트리밍 다운로드한다.
void downloadXlsFile(String statementName, Map param, String fileName, HttpServletResponse res)	statementName과 param으로 조회한 다건의 데이터를 Xls파일로 스트리밍 다운로드한다.
void downloadXlsxFile(String statementName, Map param, String fileName, HttpServletResponse res)	statementName과 param으로 조회한 다건의 데이터를 Xlsx파일로 스트리밍 다운로드한다.

라. SqlMap

(1) 개요

본 시스템은 ORM (Object Relational Mapping)으로 MyBatis를 선정했다.

Query를 Java소스로부터 분리함에 따라 업무 로직에 집중하게 함으로써 개발자의 생산성 향상 및 업무 효율을 높여준다.

(2) 작성요령

■ 개발자가 작성해야 할 Sqlmap 및 경로는 mil.dhrmis.{시스템명}.{Level1}.{Level2}.{level3}comp.sqlmap.{Entity + ActionGroup}Sql.xml이고, 별도의 환경설정파일 등록은 필요하지 않다.

■ 동적쿼리 작성시 적절한 Tag 와 Attribute 를 사용한다.

■ 조회쿼리 작성시 컬럼 개수를 동적으로 분리하는 분기처리는 절대 금지.
(최초 1회 실행시 만들어진 ResultSet 컬럼개수로 컴파일 되어, 이후 더 많은 컬럼이 요청될 때 오류발생)

ex) select

```
<if tets="xxx">emp_no</if>
<if tets="xxx">emp_name</if>
<if tets="xxx">dept_id</if>
from EMP
```


■ Tag

유형	설명
<if>	조건
<choose> <when> <otherwise>	선택
<trim>	Where 절에서 맨 앞에 and나 or 없애주는 태그
<foreach>	루프

■ Attribute

속성	설명
test	조건비교
prefix. prefixOvverides	Where 절 앞에 and나 or를 처리

■ Parameter 와 ResultSet 에 대해 MyBatis 가 기본적으로 제공하는 Alias 와 사용자정의 Alias 를 나열하였다. (ibatis TypeAliasRegistry 및 sqlmap-config.xml 에서 제공)

alias	type
string	java.lang.String
byte	java.lang.Byte

long	java.lang.Long
short	java.lang.Short
int, integer	java.lang.Integer
double	java.lang.Double
float	java.lang.Float
boolean	java.lang.Boolean
date	java.util.Date
decimal	java.math.BigDecimal
object	java.lang.Object
map	java.util.Map
hashmap	java.util.HashMap
list	java.util.list
arraylist	java.util.ArrayList
collection	java.util.collection
iterator	java.util.Iterator
camelCaseMap	egovframework.rte.psl.dataaccess.util.EgovMap
orderedMap	java.util.LinkedHashMap

(3) 예제

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.rbacomp.UserMgtSql">

    <select id="selectUserById" parameterType="map" resultType="egovMap">
        select T1.emplyr_id,
               T1.user_nm,
               T1.passwd
          from COMTNEMPLYRINFO T1
         where emplyr_id = #{id}
    </select>

    <select id="selectUserList" parameterType="map" resultType="egovMap">
        select T1.emplyr_id,
               T1.orgnzt_id,
               T1.user_nm,
               T1.passwd,
               to_char(T1.sbscrb_de,'YYYY-MM-DD') as sbscrb_de,
               (select count(1) from COMTNEMPLYRAUTHS M1 where M1.emplyr_id = T1.emplyr_id) as auth_cnt
          from COMTNEMPLYRINFO T1
         where 1 = 1
        <if test=" srchText != null and srchText != ''">
          and (emplyr_id like '%'||#{srchText}||'%' or user_nm like '%'||#{srchText}||'%')
        </if>
    </select>

    <insert id="insertUser" parameterType="map">
      <selectKey keyProperty="emplyrId" resultType="String" order="BEFORE">
        select 'U' || lpad(nvl(max(substr(emplyr_id, 2)), 0) + 1, 7, '0') as emplyr_id from COMTNEMPLYRINFO where
emplyr_id not like 'test%'
      </selectKey>
        insert into COMTNEMPLYRINFO (
            emplyr_id,
            orgnzt_id,
            user_nm,
            passwd,
            sbscrb_de)
        VALUES (
            #{emplyrId},
            #{orgnztId},
            #{userNm},
            #{passwd},
            sysdate)
    </insert>

    <update id="updateUser"  parameterType="map">
        update COMTNEMPLYRINFO
           set emplyr_id = #{emplyrId},
               orgnzt_id = #{orgnztId},
               user_nm = #{userNm},
               passwd = #{passwd},
               crtfc_dn_value = #{crtfcDnValue},
               sbscrb_de = SYSDATE
         where emplyr_id = #{emplyrId}
    </update>

    <delete id="deleteUser" parameterType="mil.dhrmis.sys.rba.vo.UserVO">
        delete from COMTNEMPLYRINFO
        where emplyr_id = #{emplyrId}
    </delete>
</mapper>
```

Null과 빈문자열 동시 체크

마. Request & Response 처리

(1) Request 처리

■ Controller 에서 특별한 경우를 제외하곤, Parameter 로 HttpServletRequest 을 쓰지말고 MCIRequest 사용

■ 본 시스템에서 제공하는 MCIRequest 는 사용자의 요청이 Websquare 이든, Nexacro 든 기타 상용 UI Framework 이든 jsp 이든 표준웹이든 다양한 UI 클라이언트와 연동할 수 있도록 하는 멀티채널서비스 이다.

Method	설명
String getParameter(String paramName)	지정한 파라미터명으로 String형태의 데이터를 받는다.
Map getParam(String datasetName)	지정한 데이터셋명으로 Map형태의 데이터를 받는다.
Map getMap(String datasetName)	지정한 데이터셋명으로 Map형태의 데이터를 받는다.
List getGridMap(String datasetName)	지정한 데이터셋명으로 List<Map>형태의 그리드데이터를 받는다.
List getGridVO(String datasetName, Class cls)	지정한 데이터셋명으로 List<VO>형태의 그리드데이터를 받는다.
void fileUpload()	파일 업로드
List<FileVO> getFileList()	업로드한 파일정보 목록 취득

(2) Response 처리

■ 표준웹 데이터 통신시 (text/html 로 응답)

```
@Controller
public class UserMgmtController{

@RequestMapping("/rba/userMgt")
public String selectUserList(MciRequest req, Model model) throws Exception
{
    // get parameter
    Map<String, Object> param = req.getParam("param");

    // invoke service method
    List<Map<String, Object>> list = userMgmtService.selectUserList(param);

    // response
    model.addAttribute("dsUserList", list);

    return "/userMgt";
}

@RequestMapping("/rba/userMgt2")
public ModelAndView selectUserList(MciRequest req, Model model) throws Exception
{
    // get parameter
    Map<String, Object> param = req.getParam("dsParam");

    // invoke service method
    List<Map<String, Object>> list = userMgmtService.selectUserList(param);

    // response
    ModelAndView mv = new ModelAndView();
    mv.addAttribute("userList", list);
    mv.setView("/userMgt");

    return mv;
}
```

지정한 String의 문자열 이름의 jsp페이지로 Forwarding.

지정한 View 이름의 jsp로 Forwarding하고, userList라는 속성명으로 데이터를 Return.

■ Ajax 데이터 통신시 (json data 로 응답)

```
package mil.dhrmis.oes.biz.ct.cncl.cmgt.web;

@Controller
public class UserMgmtController
{
    @RequestMapping("/rba/selectUserList")
    public void selectUserList(MciRequest req, Model model) throws Exception{
        // get parameter
        Map<String, Object> param = req.getParam("dsParam");

        // invoke service method
        List<Map<String, Object>> list = userMgmtService.selectUserList(param);

        // response
        model.addAttribute("userList", list);
    }
}
```

리턴페이지가 없는 경우 void로 처리한다.

바. Transaction

(1) 개요

Bisuness 처리중 오류 발생시 전체 Commit 또는 일부 Commit 또는 전체 Rollback등의 처리.

(2) 작성요령

■ 트랜잭션 처리를 위한 여러방법이 있는데 (Programming 방식 vs Declarative 방식)
본 프로젝트는Annotaion 선언방식을 사용한다.
트랜잭션처리를 위해 @Transactional 이라는 어노테이션을 Service Method 선언부 상단에
적어주는데 경우의 수는 아래와 같이 여러가지가 있다.

Transactional(propagation=Propagation.REQUIRED),
Transactional(propagation=Propagation.REQUIRES_NEW)
Transactional(readOnly=true)
Transactional(propagation=MANDATORY),
Transactional(propagation=SUPPORTS),
Transactional(propagation=NOT_SUPPORTS),
Transactional(propagation=NEVER),
Transactional(propagation=NESTED)

■ Propagation 은 트랜잭션 전파에 관한 설정으로
하나의 서비스가 다른 서비스를 호출하고 그 서비스가 또 다른 서비스를 호출하는 등
여러 서비스가 하나로 묶였을 때 해당 트랜잭션의 전파에 관한 설정.

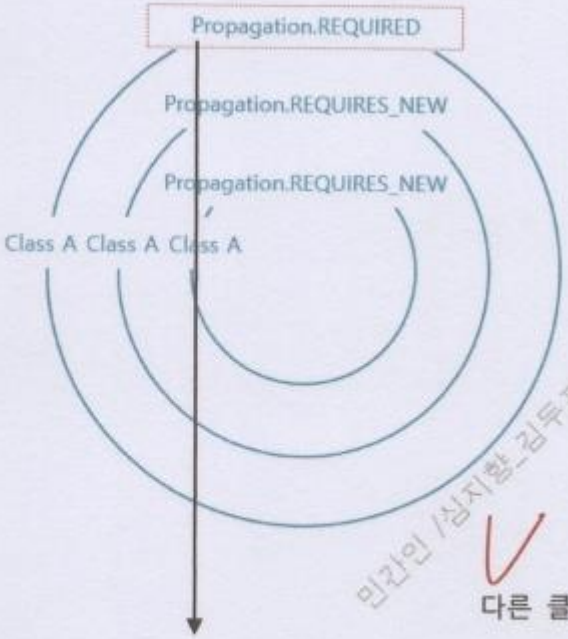
■ 일반적으로 트랜잭션이 필요할 경우에는 @Transactional(propagation=Propagation.REQUIRED)를 쓴다.

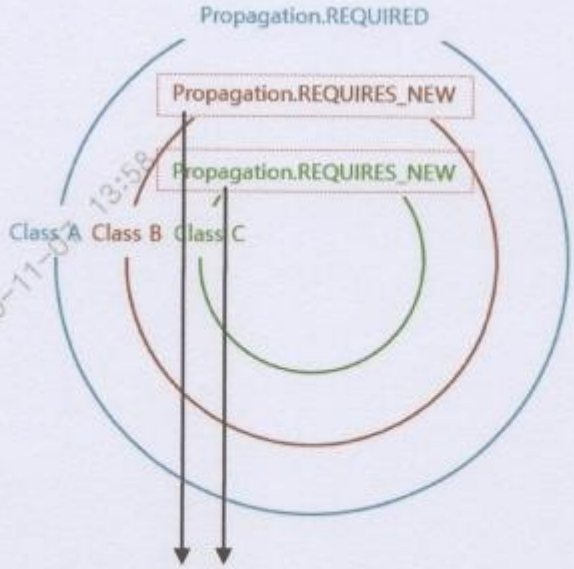
■ 트랜잭션이 필요없는 경우에는 @Transactional 을 쓰지 않는다.

■ 하나의 클래스 내에서는 최초 호출되는 메소드 이외에 나머지 메소드에 선언된 Propagation 속성은 아무 의미가 없다.

■ 단순 조회시에는 @Transactinal(readOnly=true)을 쓰는데 사실상 의미가 없다.

■ Propagation.REQUIRED 로 선언된 메소드가 다른 Propagation.REQUIRED 로 선언된 메소드를 호출하게 되면 하나의 트랜잭션으로 묶이는데 이때 해당 메소드의 트랜잭션을 별도로 빼고 싶으면 Propagation.REQUIRES_NEW를 쓰는데 별도의 Class로 새로 작성해야 한다.





다른 클래스 메소드를 호출하면 개별 트랜잭션으로 실행될수 있다.

같은 클래스내에서 이면 제일 밖에 있는 메소드의 Propagation이 적용된다.

(3) 예제

```
package kr.go.acrc.sys.rba.service;

import java.util.HashMap;
import java.util.List . . .

@Service
public class UserMgtService
{
    @Autowired
    private CommonDao dao;
    @Autowired
    private LogService logService;

    @Transactional(readOnly=true)
    public List<Map<String, Object>> selectUserList(Map<String, Object> params)
    {
        return dao.queryForList("UserMgt.selectUserList", params);
    }

    @Transactional(propagation = Propagation.REQUIRED)
    public void saveUserList(List<Map<String, Object>> lstData)
    {
        for(int i = 0; i < lstData.size(); i++)
        {
            if(rowStatus.equals(RowStatus.INSERT))
            {
                // leaveaLog(userData);
                logService.leaveaLog(userData);
                dao.insert("UserMgt.insertUser", userData);
            }
        }
    }

    @Transactional(propagation=Propagation.REQUIRES_NEW)
    public void leaveaLog(Object obj)
    {
        dao.insert("LogMgt.insertJobLog", obj);
    }
}
```

```
public class LogService
{
    @Transactional(propagation=Propagation.REQUIRES_NEW)
    public void leaveaLog(Object obj)
    {
        dao.insert("LogMgt.insertJobLog", obj);
    }
}
```

사. Stored Procedure

(1) 개요

비즈니스 로직을 Stored Procedure에서 처리하고자 하는 경우의 예제

(2) 작성요령

Mybatis Query 작성시 statementType은 CALLABLE로, parameterType은 map으로,
이 외, input parameter/ output parameter 관련하여 각 jdbcType과 javaType에 맞게 선언하여 사용한다.

(3) 예제

① 단순 Procedure

- Return 이 없는 경우

```
<select id="test_proc1" statementType="CALLABLE" parameterType="map">
    { call test_proc1(#{ipt,  mode=IN,  jdbcType=VARCHAR, javaType=string})}
</select>

public void procedureCall1() throws Exception
{
    Map<String, Object> param = new HashMap<String, Object>();
    param.put("ipt", "2");
    dao.queryForObject("TestMgt.test_proc1", param);
}
```

② String Out Parameter Procedure

- Return이 있는 경우

```
<select id="test_proc2" statementType="CALLABLE" parameterType="map">
    { call test_proc2(
        #{ipt,  mode=IN,  jdbcType=VARCHAR, javaType=string},
        #{rst1, mode=OUT, jdbcType=VARCHAR, javaType=string},
        #{rst2, mode=OUT, jdbcType=VARCHAR, javaType=string})}
</select>

public void procedureCall2() throws Exception
{
    Map<String, Object> param = new HashMap<String, Object>();
    param.put("ipt", "2");
    dao.queryForObject("TestMgt.test_proc2", param);
    Logger.debug("..." + param);
}

// 결과 : ...{rst2=b, rst1=a, ipt=2}
```

③ Cursor Out Parameter Procedure

- Return이 Cursor인 경우

```
<resultMap id="testMap" type="egovframework.rte.psl.dataaccess.util.EgovMap"/>
<select id="test_proc3" statementType="CALLABLE" parameterType="map">
    { call test_proc3
        (#{ipt, mode=IN,  jdbcType=VARCHAR, javaType=string},
         #{rst, mode=OUT, jdbcType=CURSOR,  javaType=ResultSet, resultMap=testMap})}
</select>

public void procedureCall3() throws Exception
{
    Map<String, Object> param = new HashMap<String, Object>();
    param.put("ipt", "2");
    dao.queryForList("TestMgt.test_proc3", param);
    Logger.debug("..." + param);
}

// 결과 : ...{rst=[{a=a, b=b, c=c}, {a=a, b=b, c=c}], ipt=2}
```

④ 트랜잭션

```
CREATE OR REPLACE procedure TEST.test_proc4
(ipt in varchar2)
is
begin

   -- insert into test1 values((select max(seq) + 1 from test1), '1',
'2', '3', '4');
   --insert into test1 values((select max(seq) + 1 from test1), '1',
'2', '3', '4');
   insert into test1 values(9, '1', '2', '3', '4');

   insert into test1 values(10, '1', '2', '3', '4');
   commit;
exception when others then
   rollback;
   raise;
end;
/
```

Default 로는 Stored Procedure 가 알아서 Transaction 을 관리하고,
Spring 의 Transaction 이 해당 Procedure 의 Transaction 을 관리할 수 없다.
단지, Procedure 로부터 Exception 을 받으려면 rollback; 다음에 raise;를 써줘야 한다.

아. Exception 처리

(1) 개요

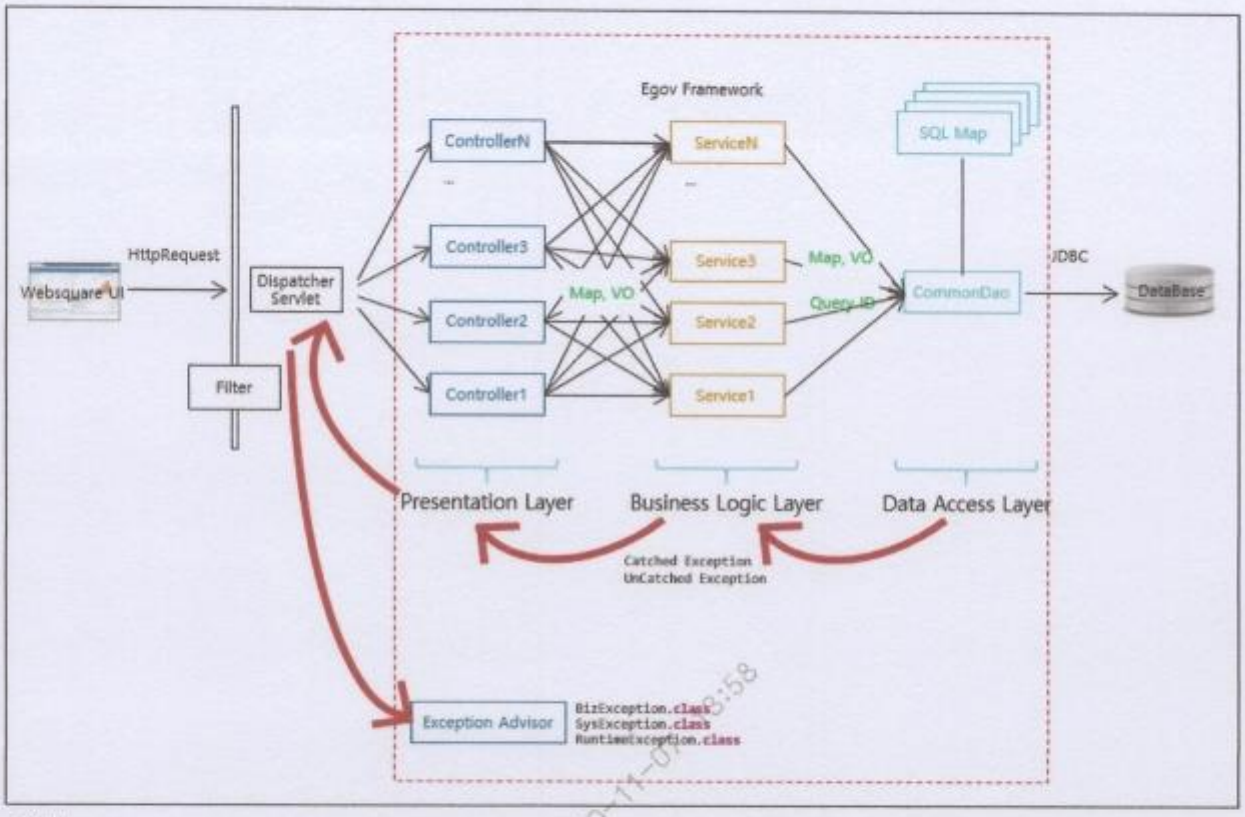
- Controller 를 시작으로 Service, DAO Business Logic 을 수행하는 도중에 Error 가 발생하게 되었을때, try/catch Block 으로 별도 제어하지 않는 한 Dispatcher Servlet 으로 에러가 전달되며, Dispatcher Servlet 은 Exception Advisor 로 예외를 던져 공통단에서 이를 수집하여 처리한다.
- Exception 은 2 가지 Case 에서 발생한다. Checked Exception, Unchecked Exception. Exception 이 예상이 되느냐, 안되느냐의 차이.
- Checked Exception 과 같은 경우 catch 구문에서 new SysException 으로 새로 예외메시지를 작성해서 처리한다.
- 실제 오류가 아니고, Business 상황의 오류인 경우싶은경우 new BizException 으로 예외메시지를 작성해서 처리한다.
- BizException 의 오류메시지는 msg.properties 를 이용하여 아래의 예와 같이 작성한다.

ex1. throw new BizException(MessageSourceAccessor.*getMessage("msgCode1")*)

- 복잡한 메시지인 경우 그냥 한글로 작성하여 아래와 같이 던진다.

ex2. throw new BizException(numRow + *"(" + data1 + ") 에서 오류가 발생하였습니다."*);

- NullPointException 과 같은 예기치 못한 RuntimeException 의 경우 default 로 "알수없는 오류가 발생하였습니다"의 메시지로 처리함.



(2) 예제

```
@Transactional(propagation=Propagation.REQUIRED)
public voidtest(List<Map<String, Object> gridData)
{
    File file = newFile("c:/test.txt" );
    try{
        file.createNewFile();
    }
    catch(IOExceptin e)
    {
        throw new SysException(MessageSourceAccessor.getMessage("msgCode1"));}
```

→ Checked Excepion 처리 방법

```
    if(file.getName().equlas("test.txt"))
    {
        ...
    }
```

→ UnChecked Excepion
으로 "알수없는 오류가 발생하였습니다" 로 처리됨.

```
    List list = dao.queryForList("UserMgmt.selectUserList", param);
    if(list.size() == 0)
    {
        throw new SysException(MessageSourceAccessor.getMessage("msgCode1")
        // new SysException(MessageSourceAccessor.getMessage(numRow + "( " + data1 + ") 에서 오류가 발생하였습니다.");
    }
}
```

→ 업무상 에러 처리 방법

자. Properties 사용

(1) 개요

- 어플리케이션 공통적인 속성 또는 고정적인 속성들을 별도의 파일로 관리.
- System, Message 2 가지.
- 연계 URL 이라든가 로그 디렉토리 경로등 시스템 관련 속성들은 sys.properties 를 사용.
- 메시지는 message.properties 를 사용한다.
- 메시지는 전체공통 메시지 및 각 파트별 메시지로 구분하여 관리한다.

(2) 작성요령

key = value 쌍으로 입력한다.

① sys.properties

- 시스템 Properties

```
db.driverclassname = cubrid.jdbc.driver.CUBRIDDriver
db.url = jdbc:cubrid:192.168.0.201:30000:actsdb:::?charset=utf-8

db.username = actsapp
db.password = actsapp
file.upload.dir = C:/DHRMISIDE/workspace/DHRMIS/WebContent/WEB-INF/files
file.upload.tmp.dir = C:/DHRMISIDE/workspace/DHRMIS/WebContent/WEB-INF/files/tmp
file.upload.threshold = C:/DHRMISIDE/workspace/DHRMIS/WebContent/WEB-INF/files/tmp
file.upload.max.size =2048576000
```

② message.properties

- 업무 메시지 Properties

```
#########################################################
# 공통메세지
#########################################################
error.cmn.unkonwnerror = 알수없는 오류가 발생하였습니다. 관리자에게 문의하여 주세요.
error.cmn.access.denied = 권한이 없습니다.

#########################################################
# 인사메세지
#########################################################

#########################################################
# 평정메세지
#########################################################

#########################################################
# 자격메세지
#########################################################
...
```

(3) 예제

1. Java에서 Value Annotation으로 사용

```
public class CommonDao {
@Value("#{sysProps['file.upload.path']}")
private String fileUploadPath;

public void downloadXlsFile(String statementName, Map<?, ?> param, String fileName, HttpServletResponse res) throws IOException
{
    ...
    String tmpFileName = UUID.randomUUID().toString();
    File tmpFile = new File(fileUploadPath + "/tmp/" + tmpFileName);
    FileOutputStream fos = new FileOutputStream(fileUploadPath + "/tmp/" + tmpFileName);
    wb.write(fos);
    ...
}
```

@Value annotation을 이용하여 Properties값을 가져올수 있다.

2. Java에서 MessageAccessor Util을 이용해서 사용

```
@ControllerAdvice
public class ExceptionHandlingController{

@Autowired
private MessageAccessor messageAccessor;

@ExceptionHandler({ BizException.class })
Public void bizError(HttpServletRequest req, HttpServletResponse res, BizException e){
    if(req.getHeader("Accept").startsWith(MediaType.APPLICATION_JSON_VALUE))
    {
        String message = messageAccessor.getMessage("ct.info.aaa");

...
}
}
```

3. Spring Configuration Xml File에서 사용

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beansxmlns="http://www.springframework.org/schema/beans"
  ...
<beanid="dataSource"class="org.springframework.jdbc.datasource.DriverManagerDataSource"
    p:driverClassName="#{sysProps['db.driverClassName']}"
    p:url="#{sysProps['db.url']}"
    p:username="#{sysProps['db.username']}"
    p:password="#{sysProps['db.password']}">
</beans>
```

차. Session 처리

(1) 개요

SpringSecurity의 UserDetails를 확장한 UserInfoHolder의 Static method를 사용
개발 요구사항에 맞게 필드 추가 예정.

(2) API

Method	설명
static UserDetails getUserInfo()	세션에서 사용자객체를 가져온다.
static String getUserId()	세션에서 사용자의 Id를 가져온다
static String getUserNm()	세션에서 사용자의 이름을 가져온다.
static String getDeptCd ()	세션에서 사용자의 부서코드를 가져오다.
static String getDeptNm()	세션에서 사용자의 부서명을 가져온다.
...	...

(3) 예제

1. Java단에서 쓸때

```
@Service
public class TestService
{
    ...
    @Transactional(propagation=Propagation.REQUIRED)
    public void calculateSomeLogic(Map param)
    {
        ...
        logger.debug("사용자아이디 : " + UserInfoHolder.getUserId());
        ...
    }
}
```


2. MyBatis단에서 쓸때

```
<select id="selectUserRoleMapList" parameterType="hashmap" resultType="camelCaseMap">
    select decode(T1.user_id, null, '0', '1') chk,
           nvl(T1.user_id, #{userId}) user_id,
           T2.role_cd,
           T2.role_nm,
           T2.role_desc
      from T_USER_ROLE_MPNG T1,
           T_ROLE_MGMT T2
     where T1.role_cd(+) = T2.role_cd
       and T1.user_id(+) = #{SS_USER_ID}
  order by T2.role_cd
</select>
```

카. File 처리

(1) 사전지식

1. 파일 테이블 구조는 아래와 같고, 업무테이블에서는 파일 마스터 테이블의 PK 인 atch_file_id 만 가져간다

업무테이블

person_id	person_name	atchFileId
1234	홍길동	1

파일마스터

atch_file_id	file_desc
1234	

파일상세

atch_file_id	file_seq	file_name	file_path
1234	1	test1.pdf	/u01/files/2020/07/29/test1.pdf
123	2	tet2.pdf	/u01/files/2020/07/29/test2.pdf

2. atch_file_id 는 Hashing 함수로 생성된 값을 사용한다.

한 예로, http://www.dhrmis.mil/file/download.do?atchFileId=1&fileSeq=1

의 주소를 발견하게 된다면

http://www.dhrmis.mil/file/download.do?atchFileId=1&fileSeq=2 와 같이

다른 seq 번호를 유추할수 있어 다른 사람의 데이타도 볼수 있게 된다.

sequence 번호를 유추할수 없게 난수로 표현하는 방법을 사용.

Hash 함수 SHA-2 Series 중 하나인 SHA-256 을 사용.

이 알고리즘으로 Hashing 하면 16 진수 표기법에 따라 64 자리 난수가 리턴.

참고로 중복발생 가능성은 $1 / 2^{256}$ 로 사실상 불가능 하다.

1

/

157920892373161945235985000868907852323340560394578400791312963993642357098561157920892373119452359850008689078523233405603945784007913129639936

정도 (78 자리)

예제는 B94D27B9934D3E08A52E52D7DA7DDABFAC484EFE37A5380EE9088F7ACE2EFCD9 와 같다.

(2) 첨부파일 처리

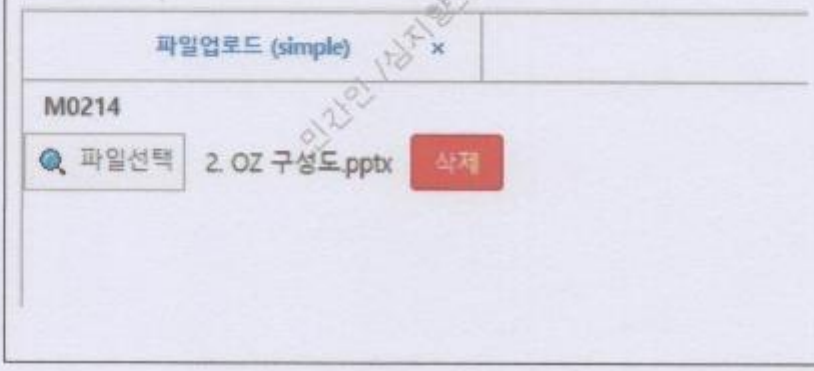
① 첨부파일 업로드 샘플

■ 파일 업로드 UI 컴포넌트를 이용하는 경우

- 파일업로드폼을 넣고자 하는 위치에 생성하고,해당 div id 를 parameter 로 com.file.multiForm 호출

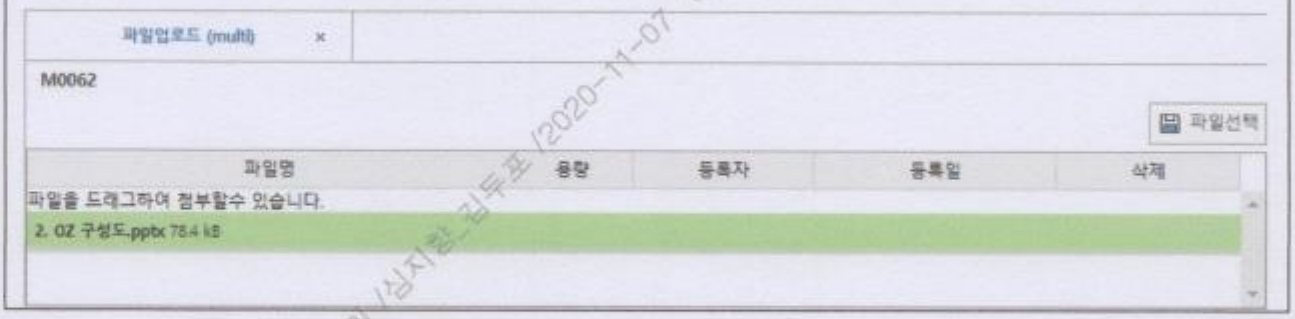
simpleform

```
<body ev:onpageload="scwin.onpageload" ev:onpageunload="scwin.onpageunload">
    <div id='test1' style='width:850px;height:161px;'></div>
</body>


scwin.onpageload = function()
{
    com.file.simpleForm('test1', '10', '1',
        function()
        {
            console.log('폼이 로드되고 초기화 하고 싶은 코드를 넣어주세요.');
        },
        function(response)
        {
            console.log('파일이 업로드 되고 나서 실행할 코드를 넣어주세요.');
            console.log(response['list'][0].fileId);
            console.log(response['list'][0].fileSn);
        }
    );
};
```



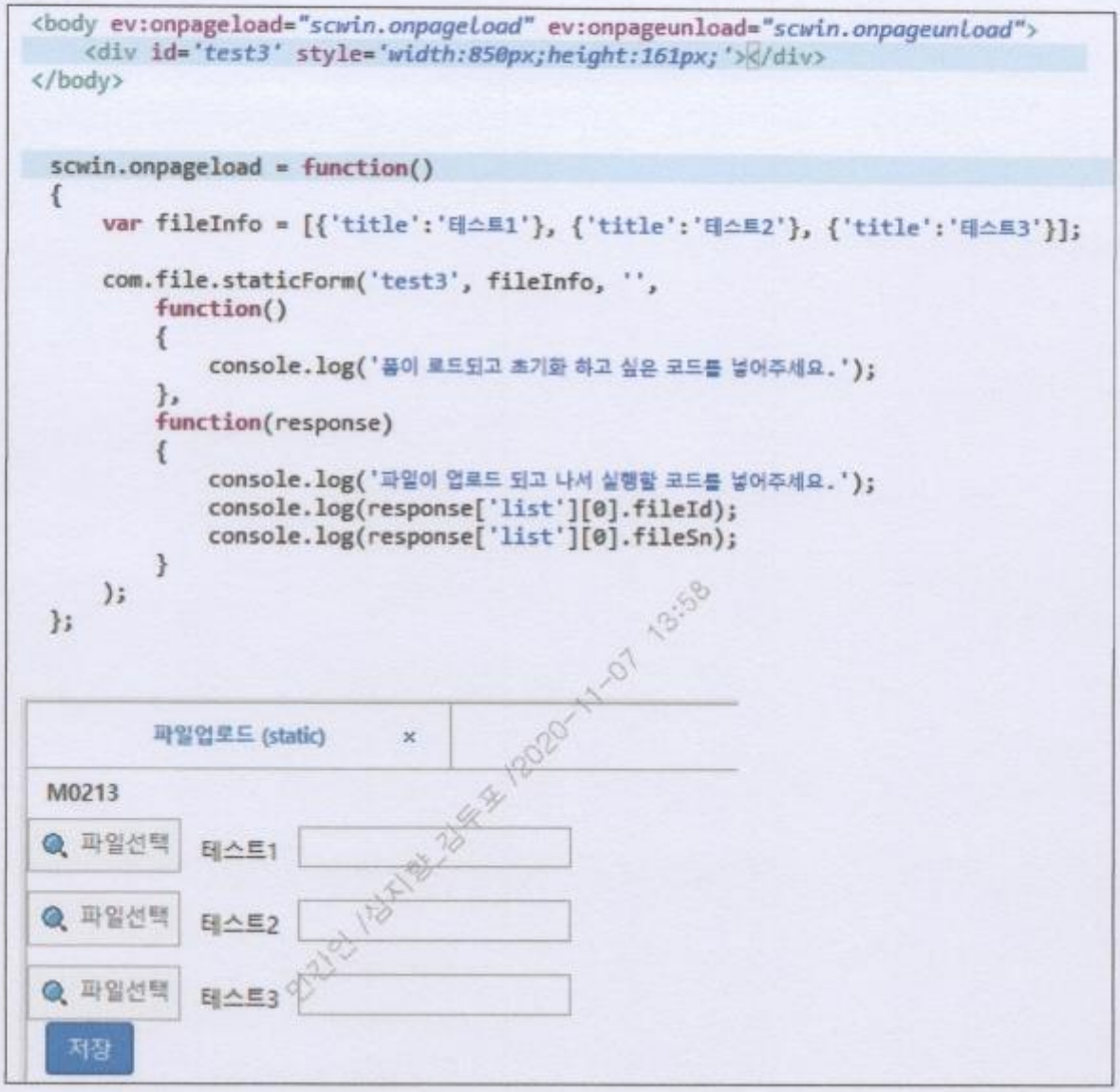
multiform

```
<body ev:onpageload="scwin.onpageload" ev:onpageunload="scwin.onpageunload">
    <div id='test1' style='width:850px;height:161px;'></div>
    <!--xf:group style="width:850px;height:161px;" id="test1" tagname="div"-->
</body>

scwin.onpageload = function()
{
    com.file.multiForm('test1', '',
        function()
        {
            console.log('폼이 로드되고 초기화 하고 싶은 코드를 넣어주세요.');
        },
        function(response)
        {
            console.log('파일이 업로드 되고 나서 실행할 코드를 넣어주세요.');
            // JSON.parse(response).list[0].atchFileId
            // JSON.parse(response).list[0].fileSn
        }
    );
};
```



staticform

```
<body ev:onpageload="scwin.onpageload" ev:onpageunload="scwin.onpageunload">
    <div id='test3' style='width:850px;height:161px;'></div>
</body>


scwin.onpageload = function()
{
    var fileInfo = [{'title':'테스트1'}, {'title':'테스트2'}, {'title':'테스트3'}];

    com.file.staticForm('test3', fileInfo, '',
        function()
        {
            console.log('폼이 로드되고 초기화 하고 싶은 코드를 넣어주세요.');
        },
        function(response)
        {
            console.log('파일이 업로드 되고 나서 실행할 코드를 넣어주세요.');
            console.log(response['list'][0].fileId);
            console.log(response['list'][0].fileSn);
        }
    );
};
```



■ 직접 파일 업로드를 할 경우

- Multipart ContentType 으로 (<form method="post" enctype="multipart/form-data">...</form>) Form 을 Submit 하고, Controller 에서 아래와 같이 파일 정보를 받아서 처리.

DB 에 등록할지는 업무의 성격에 맞게 설계

```
@RequestMapping("/rba/saveUserList.do")
public void saveUserList(MciRequest req, Model model) throws Exception
{
    // get parameter
  req.fileUpload();
  List<FileVO> fileVOs = req.getFile();

    // get file information
  for(int i = 0; i < fileVOs.size(); i++)
  {
    System.out.println(fileVOs.get(i).getFileStreCours());
    System.out.println(fileVOs.get(i).getOrignlFileNm());
    System.out.println(fileVOs.get(i).getFileMg());
  }
}
```

② 첨부파일 다운로드 샘플

- atchFileId 와 fileSeq 를 넘겨 다운로드

```
scwin.btnDownload_onclick = function(e)
{
    com.file.download(1, 2);
};
```

(3)

(4) .엑셀파일 처리

① Grid 엑셀파일 업로드

```
var gridId = "grdUser";
var type = "xlsx";
var options = { fileName : "test.xlsx" };

com.gridDataUpload(grdObj, type, options);
```

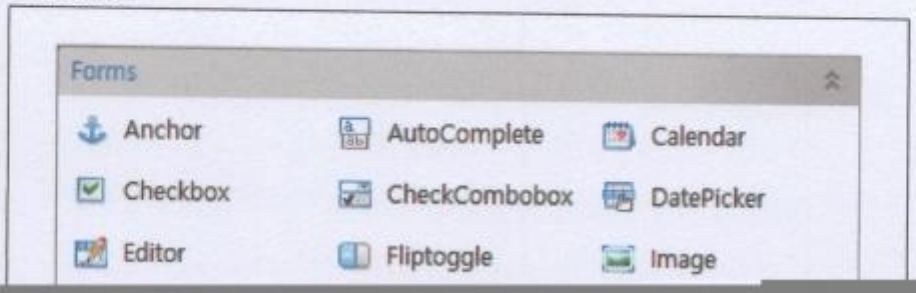
② Grid 엑셀파일 다운로드

```
var gridId = "grdUser";
var infoArr = {}; // infoArr 그리드에 대한 내용을 추가로 다른 셀에 표현하는 경우 사용하는 배열
var options = { fileName : "test.xlsx" };

com.gridDataDownLoad(gridId, options, infoArr);
```

③ DB 엑셀파일 업로드 (csv, xls, xlsx)

■ 엑셀파일을 업로드하여 DB에 Insert하는 기능

- ui source



- script source

Upload 컴포넌트 이용하고, 속성에서 id와 action 부여한 다음 아래와 같이 특정한 버튼을 클릭했을때 스크립트 처리

```
scwin.btnSubmit_onclick = function(e)
{
    upload1.submit();
};
```

- server source

Controller에서 file 정보 받아서 CommonDao 호출

```
@RequestMapping("/rba/saveUserList.do")
public void saveUserList(MciRequest req, Model model) throws Exception
{
    // get parameter
    req.fileUpload();
    List<FileVO> fileVOs = req.getFile();

    // get file information
    System.out.println(fileVOs.get(0).getFileStreCours());

}
```

CommonDao의

```
public void uploadCsvFile(String sqlId, Class cls, String fileName, String cvsSplitBy)
public void uploadXlsFile(String sqlId, String fileName, RowConverter rowConverter)
public void uploadXlsxFile(String sqlId, String fileName, RowConverter rowConverter)
```

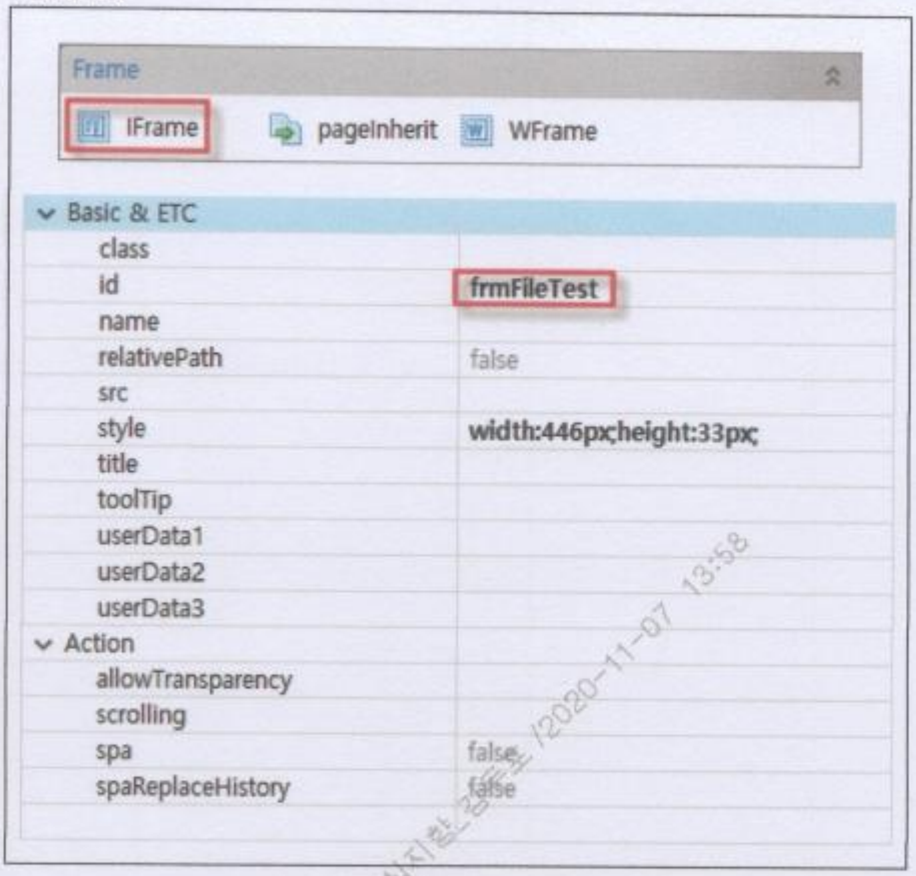
이용

```
@Override
@Transactional(propagation = Propagation.REQUIRED)
public void uploadCsvTest(FileVO fileVO) throws Exception
{
    dao.uploadCsvFile("TestMgt.insertTest2", fileVO.getFileStreCours(), ",");
}

@Override
@Transactional(propagation = Propagation.REQUIRED)
public void uploadXlsTest(FileVO fileVO) throws Exception
{
    RowConverter<TestVO> rowConverter = new RowConverter<TestVO>()
    {
        @Override
        public TestVO convert(Object[] row)
        {
            return new TestVO((Object)row[0], (Object)row[1], (Object)row[2], (Object)row[3]);
        }
    };
    dao.uploadXlsFile("TestMgt.insertTest3", fileVO.getFileStreCours(), rowConverter);
}

@Override
@Transactional(propagation = Propagation.REQUIRED)
public void uploadXlsxTest(FileVO fileVO) throws Exception
{
    RowConverter<TestVO> rowConverter = new RowConverter<TestVO>()
    {
        @Override
        public TestVO convert(Object[] row)
        {
            return new TestVO((String)row[0], (String)row[1], (String)row[2], (String)row[3]);
        }
    };
    dao.uploadXlsxFile("TestMgt.insertTest3", fileVO.getFileStreCours(), rowConverter);
}
```

④ DB 엑셀파일 다운로드 (csv, xls, xlsx)

■ 쿼리를 던져 엑셀파일을 Download하는 기능

- ui source



IFrame 컴포넌트 이용하고, 속성에서 id를 부여한 다음 아래와 같은 스크립트 처리

```
scwin.btnCsvDownload_onclick = function(e)
{
    frmFileTest.setSrc('/test/downloadCsvTest.do');
};
```

- server source

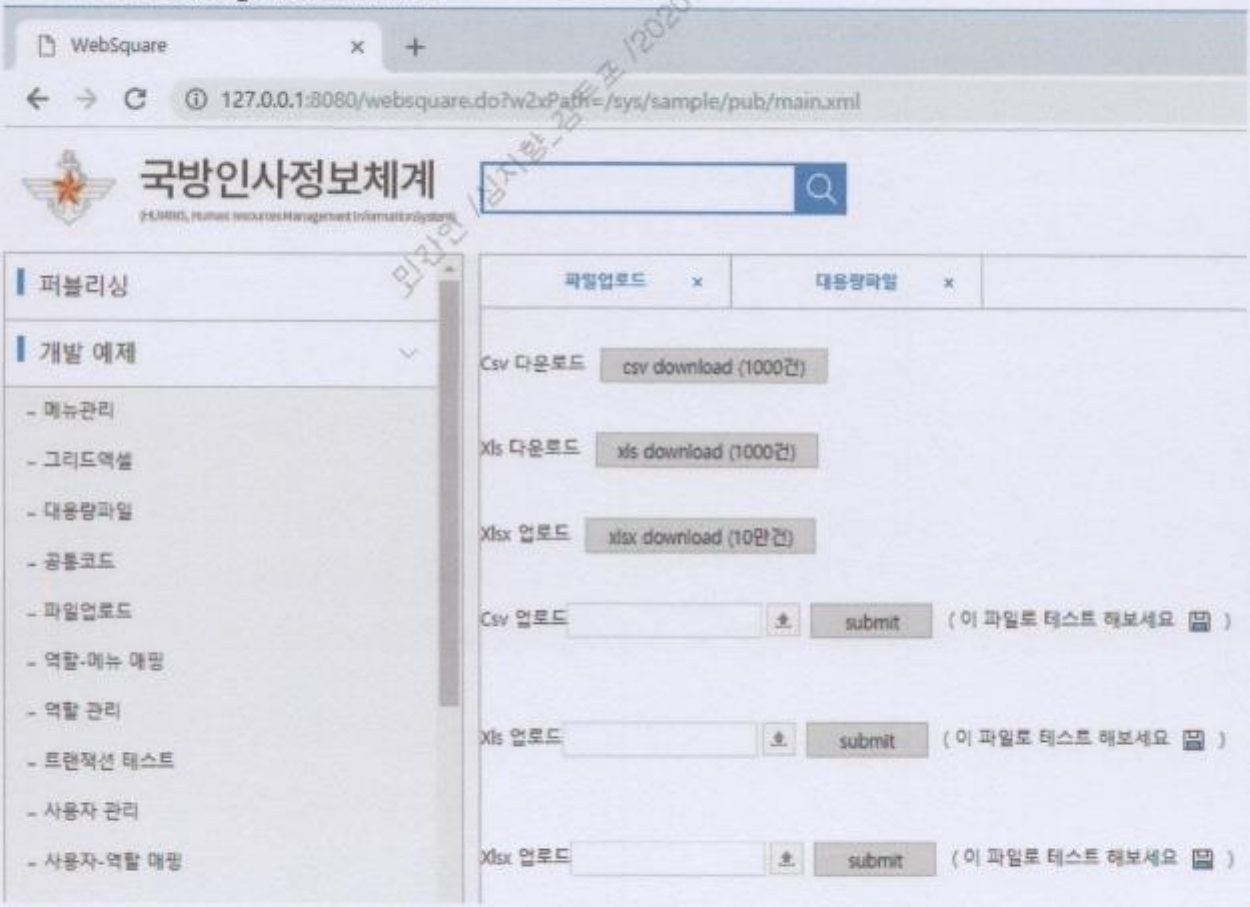
CommonDao의

```
public void downloadCsvFile(String statement, Map<String, Object> params, String fileName,
, HttpServletResponse res)
public void downloadXlsFile(String statement, Map<String, Object> params, String fileName,
, HttpServletResponse res)
public void downloadXlsxFile(String statement, Map<String, Object> params, String fileName,
, HttpServletResponse res)
```

이용

```java
@Override
public void downloadCsvTest(HttpServletResponse res) throws Exception
{
    Map<String, Object> params = new HashMap<String, Object>();
    params.put("srchTxt", "aaa");

    dao.downloadCsvFileWithRowHandler("TestMgt.selectCsvDownloadTest", params, "test-csv.csv", res);
}

@Override
public void downloadXlsTest(HttpServletResponse res) throws Exception
{
    Map<String, Object> params = new HashMap<String, Object>();
    params.put("srchTxt", "aaa");
    dao.downloadXlsFileWithRowHandler("TestMgt.selectXlsDownloadTest", params, "test-xls.xls", res);
}

@Override
public void downloadXlsxTest(HttpServletResponse res) throws Exception
{
    Map<String, Object> params = new HashMap<String, Object>();
    params.put("srchTxt", "a");

    dao.downloadXlsxFileWithRowHandler("TestMgt.selectXlsxDownloadTest", params, "test-xlsx.xlsx",
res);
}
```

※ /ui/test/largeFileTest.xml 참고



※ 한셀 지원범위 합의 필요
Hancell 은 2007 년까지 .nxl 넥셀 포맷으로 서비스. 데이터 제한이 65535 건.
이후 .cell 포맷이 나오면서 메모리가 허용하는 범위내에서 무제한 데이터 저장가능.

xls 와 xlsx 의 차이와 같음.

2007 년 이후 .cell 포맷이나 .xlsx 나 둘다 OpenXML 기반의 엑셀프로그램이 제작되어 같은 Platform 이라 어느정도 호환은 되나 기반이 같은것이지 똑같은건 아니라서 호환의 범위가 좁다.

그 범위에 대한 설명.
서버에서 xls 포맷으로 다운로드 받아 .cell 로 저장하면 정상 한셀로 작동.
서버에서 xlsx 포맷으로 다운로드 받아 .cell 로 저장하면 정상작동 안함.

결론 : 65535 이상 건수의 데이터에 대해서는 무조건 xlsx 를 사용한다.
65535 이하 건수의 데이터에 대해서는 .xls, .cell 둘다 지원한다.

한컴오피스에서 .cell 에 대한 download Java API 를 제공한다면 해결이 되겠지만. 그러하지 않다.

타. Paging 처리

(1) 개요

Java단은 전자정부 Paging Utility를 사용하여 Paging을 계산하고,
Query단에선 해당 Data를 찾기 위한 Cost를 최소한으로 줄이기 위해 P.K만 먼저 조회하는 Temp Table을 선언하는 방식을 사용하다.

(2) 예제

1. Controller

```
import egovframework.rte.ptl.mvc.tags.ui.pagination.PaginationInfo;

@RequestMapping("/selectUserPaginatedList")
public void selectUserPaginatedList(MciRequest req, Model model) throws Exception{
//  get parameter
Map<String, Object> params = req.getParam("dsParam");

    // set pagint parameter
    int currentPageNo = 1; // 현재페이지
    int recordCount = 10; // 한화면에 보여줄 목록 개수
    if(params.get("currentPageNo") != null ) currentPageNo = Integer.valueOf(params.get("currentPageNo")+"");
    if(params.get("recordCount") != null ) recordCount = Integer.valueOf(params.get("recordCount")+"");
    PaginationInfo paginationInfo = new PaginationInfo();
    paginationInfo.setCurrentPageNo(currentPageNo);
    paginationInfo.setRecordCountPerPage(recordCount);
    params.put("firstIndex", paginationInfo.getFirstRecordIndex());
    params.put("recordCount", paginationInfo.getRecordCountPerPage());

    // invoke service method
    List<?> result = userMgmtService.selectUserPaginatedList(params);

    // response
    model.addAttribute("dsUserList", result);
}
```

2. MyBatis

```
    with temp as
    (select *
      from (select paging.*,
                   rownum as rnum,
                   count(1) over() as total_count
              from (select user_id, usernm,
                      from T_USER_MGMT where 1 = 1
                     sdfasd
                   order by user_id desc) paging)
     where rnum <![CDATA[>]]>  ${firstIndex}
       and rnum <![CDATA[<=]]> (${firstIndex) + ${recordCount}))
select T1.total_count,
       A1.user_id
       A1.user_nm,

  from T_USER_MGMT A1,
       TEMP T1
 where A1.user_id = T1.user_id
order by T1.rnum

<select id="selectUserPaginatedList" parameterClass="hashmap" resultClass="camelCaseMap">
  select PAGE.total,
```

```
        T1.emplyr_id,
        T1.user_nm
    from COMTNEMPLYRINFO T1,
        (select *
           from (select PAGING.*, rownum as rnum, count(1) over() as total
                  from (select emplyr_id,
                               orgnzt_id,
                               user_nm,
                               passwd,
                               empl_no,
                               ihidnum
                          from COMTNEMPLYRINFO
                         order by emplyr_id desc) PAGING)
                 where rnum <![CDATA[>]]> ${firstIndex}
                   and rnum <![CDATA[<=]]> (${firstIndex} + ${recordCount})) PAGE
     where T1.emplyr_id = PAGE.emplyr_id
     order by PAGE.rnum
</select>


<select id="selectUserPaginated2List" parameterClass="hashmap" resultClass="camelCaseMap">
select PAGE.total,
       T1.emplyr_id,
       T1.orgnzt_id,
       T1.user_nm,
       T1.passwd,
       T1.empl_no,
       T1.ihidnum
       to_char(T1.sbscrb_de,'YYYY-MM-DD') as sbscrb_de
  from COMTNEMPLYRINFO T1
   <include refid="fw.sqlmap.CommonSql.pagingBegin"/>
       select emplyr_id from COMTNEMPLYRINFO order by emplyr_id desc
   <include refid="fw.sqlmap.CommonSql.pagingEnd"/>
 where T1.emplyr_id = PAGE.emplyr_id
order by PAGE.rnum
</select>
```

파. 각종 Data 처리

(1) XML Parsing

1. DOM Parsing

- "9. TestCase" 참조

2. SAX Parsing

- "9. TestCase" 참조

(2) JSON Parsing

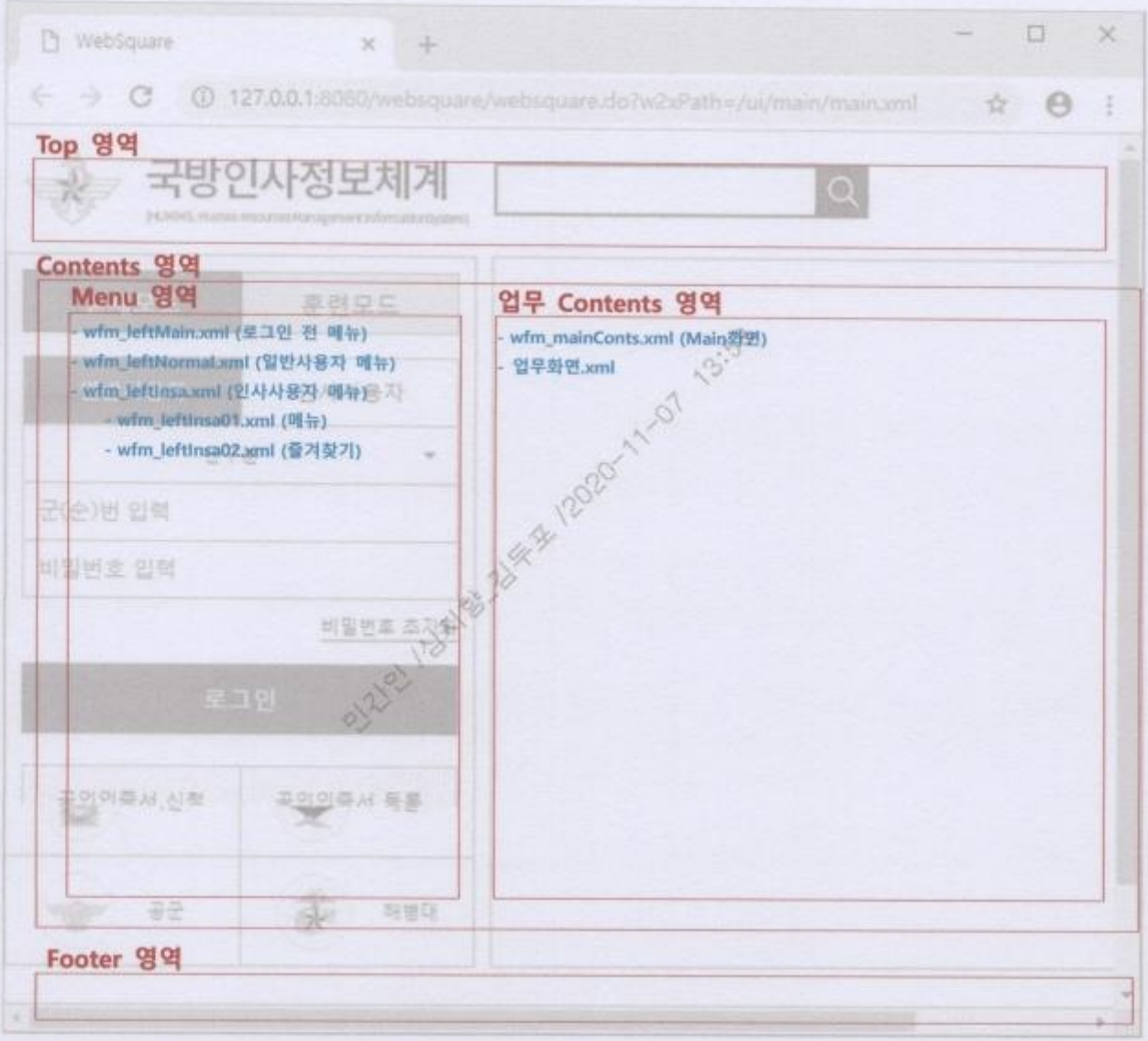
- "9. TestCase" 참조

6.2 UI 프로그램 (Websquare) 개발

가. Frame Set 개요

1. Page는 기본적으로 Top영역과, Contents, Footer 3가지로 구분하고 Contents안에는 다시 Left Menu와 업무Contents 영역으로 구분한다.

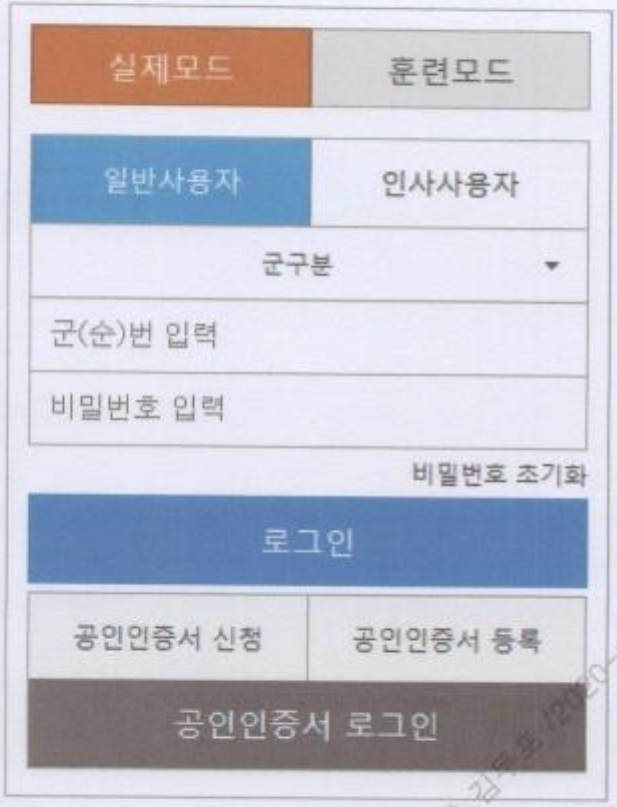
개발과 관련된 실제 영역은 업무Contnets 영역이다.



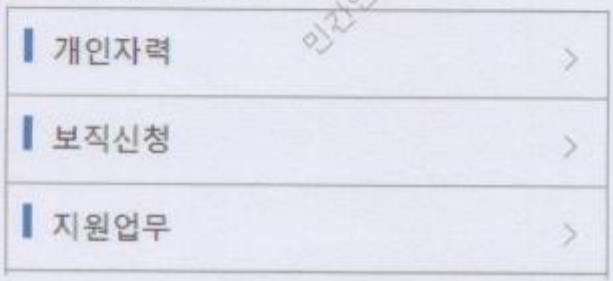
전체 너비는 1920 이고, Left Menu 는 300 외, Padding 10 을 제외하면 업무영역은 1610 내에서 개발

2. 사용자별 Left Menu

■ 로그인 페이지



■ 일반사용자 페이지



■ 인사사용자 페이지

나. 코딩 개요

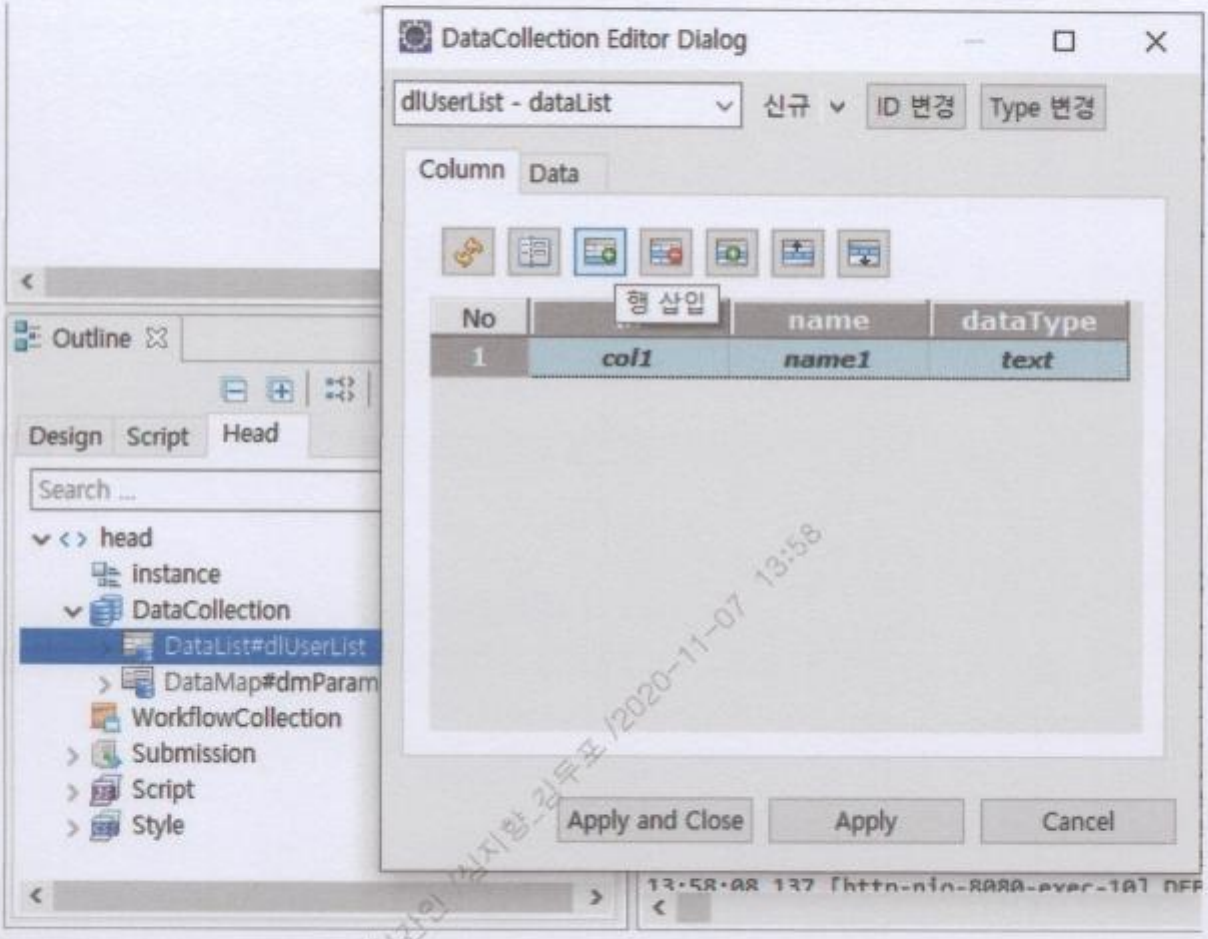
개발자의 생산성 및 이후 유지보수를 위한 가독성(readability)를 고려하여
아래와 같이 변수선언, 화면초기화, 이벤트함수, 트랜잭션함수, 사용자정의함수 순으로 작성한다.

```
//------------------------------------------------------------------------------
// 초기화
// ------------------------------------------------------------------------------
scwin.onpageload = function()
{
    scwin.fnSearch();
};

//------------------------------------------------------------------------------
// Event 함수
// ------------------------------------------------------------------------------
scwin.btnSearch_onclick= function()
{
    scwin.fnSearch();
};

scwin.btnSave_onclick = function()
{
    scwin.fnSaveList();
};

//------------------------------------------------------------------------------
// Transaction 함수
// ------------------------------------------------------------------------------
scwin.fnSearch = function()
{
    var searchOption = {
        id       : 'fnSearchList',
        action : '/rba/selectUserList.do',
        ref      : ['dmParam'],
        target : [{"id":"dlUserList", "key":"userList"}]
        // target : [{"id":"dlUserList","key":"userList"}]

    };
    com.executeSubmission(searchOption);
};

scwin.fnSave = function()
{
    var saveOption = {
        id       : 'fnSaveList',
        action : '/rba/saveUserList.do',
        ref      : ['dlUserList'],
    };
    com.executeSubmission(saveOption);
};

//------------------------------------------------------------------------------
// Callback 함수
// ------------------------------------------------------------------------------
scwin.fnCallbackHandler = function(options)
{
    var _callbackid_ = options.id.replace($p.id, '');

    if(_callbackid_ == 'fnSearchList')
    {
        console.log('local submitDone...');
    }
    else if(_callbackid_ == 'fnSaveList')
    {
        scwin.fnSearch();
    }
};
```

DataSet의 이름이 서버와 UI가 다를 경우는
[{"id":"dlUserList", "key":"userList"}]의 예와 같이
id, key를 이용하고,
같을 경우는 ['dlUserList'] 와 같이 DataSet명만 준다.

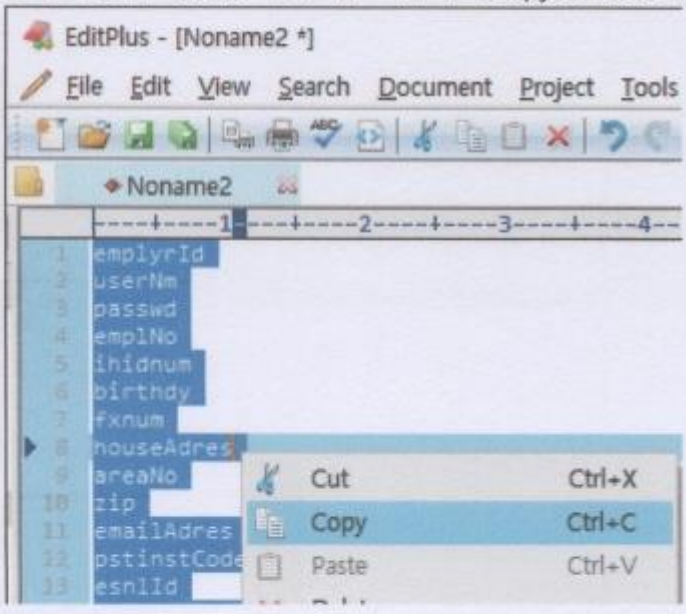
다. DataSet 등록 및 편집

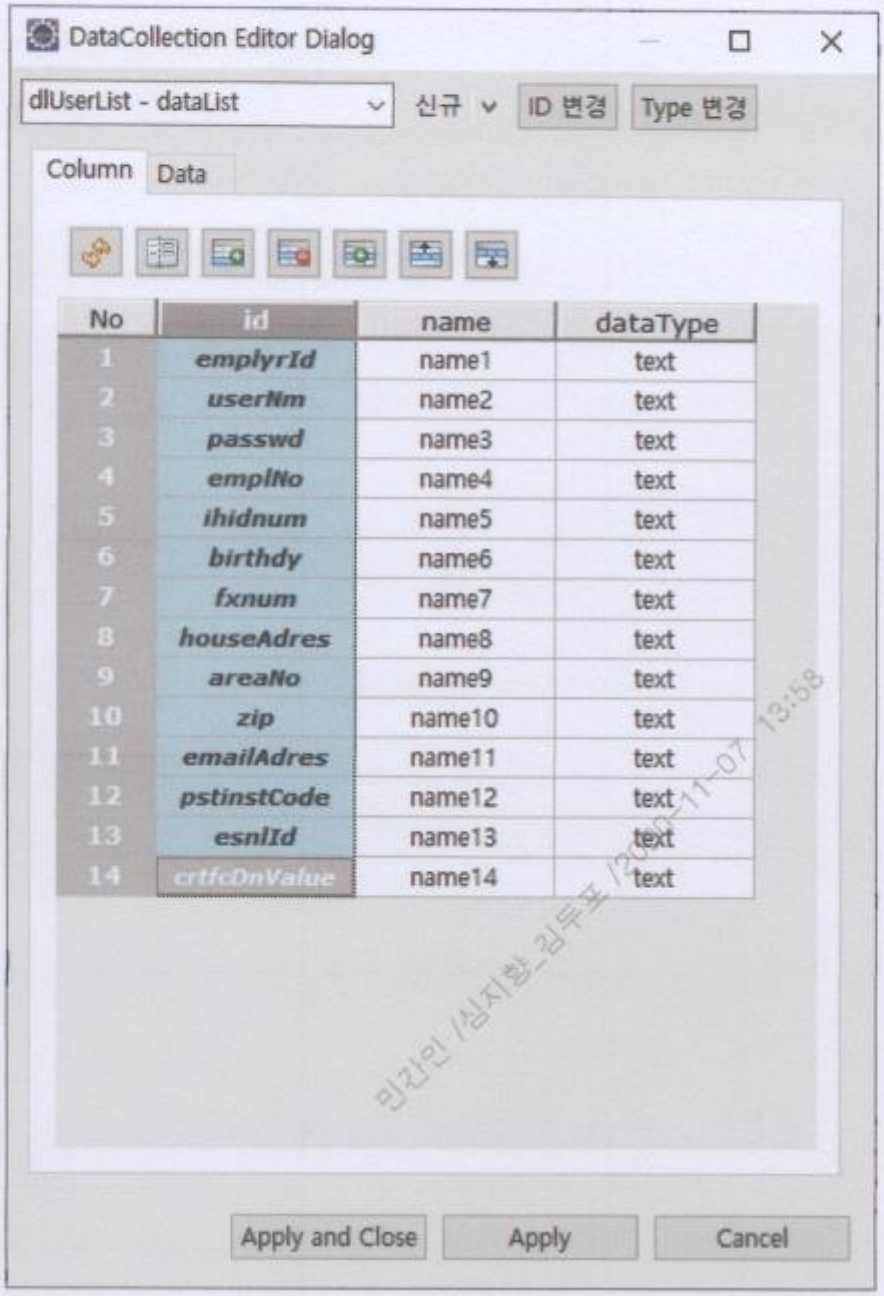
- DataCollection 및 DataMap 을 등록 편집.



DataCollection Editor Dialog 에서

아래와 같은 컬럼명 리스트를 복사하여 Copy & Paste 하면 Column 정보 자동 생성




DataCollection Editor Dialog
dlUserList - dataList
신규
ID 변경
Type 변경
Column
Data
No
id
name
dataType
1 emplyrId name1 text
2 userNm name2 text
3 passwd name3 text
4 emplNo name4 text
5 ihidnum name5 text
6 birthdy name6 text
7 fxnum name7 text
8 houseAdres name8 text
9 areaNo name9 text
10 zip name10 text
11 emailAdres name11 text
12 pstinstCode name12 text
13 esnlId name13 text
14 crtfcDnValue name14 text
Apply and Close
Apply
Cancel

라. 기본 공통 Script

- Websquare Page 에서 사용할 /websquare/config.xml 에 설정된 Default Javascript 3 가지 gcm.js, com.js, valid.js 중에서 주요한 함수들만 설명.

(1) 트랜잭션 함수

DataBase에 실제 CRUD를 하게되는 Websquare Adapter와 인터페이스를 하게 되는 함수.

예제.

```
var searchOption = {
    id       : 'fnSearch',
    action : '/rba/selectMenuList.do',
    ref    : ['dsParam'],
    target : ['menuList']
};
com.executeSubmission(searchOption);
```

(2) Callback 함수 처리

com.executeSubmission 트랜잭션이 실행될 때

1. 트랜잭션 직전 global 로 pre-submission 함수가 실행된다.

2. 트랜잭션이 정상적으로 끝나고 나면 global callback 함수가 실행되고, local 에 선언된 callback 함수가 있으면 그 다음 순차적으로 실행된다.

3. 트랜잭션에 오류가 발생하면 global error-callback 함수가 실행되고, local에 선언된 Callback함수가 있으면 그 다음 순차적으로 실행된다.

(3) Popup

팝업은 LayerPopup을 사용하며 종류에 따라 Alert, Info, Confirm, Error로 구분한다.

팝업화면에서 확인버튼이나 취소버튼등을 누를 때 호출되는 Callback함수에 대해 API확인 필요.

(4) Validation

(5) Grid 관련 Utility

Grid Data Download

ex)

```
var options = { fileName : "test.xlsx" };
var infoArr = {};
com.gridDataDownLoad(grdUser, options , infoArr);
```

→ Grid Id, Excel 종류 (csv, xls, xlsx)

Grid Data Upload

ex)

```
var options = {};
com.gridDataUpload(grdUser, "xlsx", options);
```

→ Grid Id, Excel 종류 (csv, xls, xlsx)

(6) 공통코드

해당 업무화면에서 최초 1회 호출할 때 Transaction을 타고, 이후부터는 SessionStorage에 저장해 놓은 값을 사용.

ex)

```
var codeDatas = com.getCodeData([{id:"COM00001"}, {id:"COM00002"}]);
com.setComboBox('rdoCode1', codeDatas.COM00001);
com.setComboBox('cmbCode2', codeDatas.COM00001);
```

→ 공통코드 Ids

→ 공통코드가 적용될 Element Id

(7) 첨부파일

File Upload Form

ex)

```
<?xml-stylesheet href="/css/file.css" type="text/css"?>
<script src="/js/file/require.js" scopeExternal="true" scopeVariable="cti"/>
<script src="/js/common.js" scopeExternal="true" scopeVariable="cti"/>
<script>
    Fn.File.multiForm('test1', '1',
        function()
     {
            console.log('폼이 로드되고 초기화 하고 싶은 코드를 넣어주세요.');
        },
        function(response)
        {
            console.log('파일이 업로드 되고 나서 실행할 코드를 넣어주세요.');
            // JSON.parse(response).list[0].fileId);
        }
    );
</script>
```

파일선택

파일을 드래그하여 첨부

파일명	용량	등록자	등록일	삭제
7z1805-x64.exe	undefined	undefined	undefined	삭제
ezPDFEditor_Personal_Setup_3.0.3.4.exe	undefined	undefined	undefined	삭제

(8) Fusion Chart

Fusion Chart API를 참고. (WebSquare5 (SP3) 사용자 가이드.pdf – 164 Page)

마. 전체 공통 Script

(1) gcm.js

- 메인화면 (main.xml) 최초 로딩시 Object 1 개만 생성하여 사용
- 메인화면에서만 사용할 스크립트로, 업무화면에서는 접근 불가.

(2) com.js

- 각 업무화면 별로 개별 Object 생성. (대부분 WFrame 관련)
- 모든 업무화면에서 공통적으로 사용할 유틸리티 등을 선언.
  (MDI, Submission, Grid, 공통코드, Popup, Data, Event 및 기타 함수들)

① MDI 관련

```
// main.xml 의 windowContainer 에 메뉴를 open 한다.
com.openMenu = function(psMenuId, poParam)

// main.xml 의 컨텐츠영역의 화면을 이동시킨다.
com.changeMenu = function(psTitle, psSrc, poParam)

// windowContainer 에 Tab 을 add 한다.
com.addWindow = function(poTabObj, poOpts, poParam)

// 메뉴번호에 해당하는 메뉴정보를 return
com.getMenuInfo = function(psMenuId)

// 현재 화면의 웹스퀘어 page 경로를 반환한다.
com.getPageUrl = function()

// 실행 프레임 정보 조회
com.getActiveWindowInfo = function()

// wframe 의 src 를 변경한다.
com.wfmSetSrc = function(poWframe, psSrc, poParam)
```

② 트랜잭션 관련

```
// 트랜잭션 실행
com.executeSubmission = function(options, requestData, obj)

// Submission 객체를 동적으로 생성한다.
com.createSubmission = function(options)
```

③ Grid 및 DataSet 관련

```
// GridView 와 바인딩된 DataList 객체를 반환한다.
com.getGridViewDataList = function(gridViewObj)

// 특정 컴포넌트에 바인된 DataList나 DataMap의 정보를 반환한다.
com.getDataCollection = function(comObj)

// 데이터 리스트 필터 설정 함수.
com.dataListFilter = function(_dataComp, _condition, _append, options)

// 특정 컴포넌트에 바인된 DataList나 DataMap의 컬럼 이름을 반환한다.
com.getColumnName = function(comObj)

// DataCollection 객체의 변경된 데이터가 있는지 검사한다.
com.checkModified = function(dcObjArr)

// GridView Row 삭제를 위한 CheckBox 동작을 세팅한다.
com.setGridViewDelCheckBox = function(gridViewObjArr)

// dataList에 row를 추가한다.
com.insertRow = function(poDlObj)

// dataList의 row를 삭제한다.
com.deleteRow = function(poDlObj, pbDelAllSatus)

// 그리드 엑셀 다운로드 (CSV, EXCEL을 다운로드 한다.)
// downLoadCSV 나 downLoadExcel를 호출
com.gridDataDownLoad = function(grdObj, options, infoArr)

// 그리드 엑셀 업로드
// uploadCSV 나 uploadExcel를 호출
com.gridDataUpload = function(grdObj, type, options)
```

④ 공통코드 관련

```
com.getCodeData = function(params)

com.setComboBox = function(elId, datas)
```

⑤ 팝업 관련

```
com.alert = function(popupId, messageStr, userData, closeCallbackFncName)

com.confirm = function(popupId, messageStr, userData, closeCallbackFncName)

com.error = function(popupId, messageStr, userData, closeCallbackFncName)

com.info = function(popupId, messageStr, userData, closeCallbackFncName)
```

```
// 팝업창을 연다.
com.openPopup = function(url, opt, data)

// 팝업창을 닫는다. callbackFnStr을 이용하여 부모창의 callback함수를 호출한다.
com.closePopup = function(poRetObj, poEtcInfo)
```

⑥ 데이터 관련

```
// 그룹안에 포함된 컴포넌트의 입력 값에 대한 유효성을 검사한다.
com.validateGroup = function(grpObj, valInfoArr, tacObj, tabId)

// GridView를 통해서 입력된 데이터에 대해서 유효성을 검증한다.
com.validateGridView = function(gridViewObj, valInfoArr, tacObj, tabId)

// 날자유효성체크(yyyyMMdd 8자리) 결과를 return한다.
com["regExpDateFunc"] = function(param)

// 문자열을 날자포멧으로 변형
com.strToDateFormat = function(psData)

// 일반전화번호를 유효성 체크 후 형태에 맞게 "-"을 넣어서 변경
com.telChkFormat = function(psData)

// 휴대전화번호를 유효성 체크 후 형태에 맞게 "-"을 넣어서 변경
com.moblChkFormat = function(psData)

// 메일주소 체크한다.
com.checkEmail = function(str)

// 숫자 3자리마다 ","표시 후 return
com.toFormatNum = function(psStr, pnDecimalLength)

// 세번째자리마다 콤마 표시, 금액, setDisplayFormat("#,###&#46##0",
"fn_userFormatter2") - 입력된 str에 포메터를 적용하여 반환한다.
com.transComma = function(value, type)

// dataCollection의 유효성 체크를 수행한다.
com.checkValid = function(poOpts)

// 내외국인 주민등록번호 유효성을 검사한다.
com.checkPersonID = function(str)

// 사업자번호 유효성을 검사한다.
com.checkBizID = function(str)

// 법인등록번호 유효성을 검사한다.
com.checkCorpID = function(str)

// 주민번호 문자열에 Formatter(######-#######)를 적용하여 반환한다.
com.transIdNum = function(str)

// 전화번호, setDisplayFormat("###-####-####") - 입력된 str에 포메터를 적용하여
반환한다.
com.transPhone = function(str)
```

```
// 시간 - 입력된 String 또는 Number에 포메터를 적용하여 반환한다.
com.transTime = function(value)

// 텍스트 - 입력된 str에 포메터를 적용하여 반환한다.
com.transText = function(str)

// 날짜 - 입력된 str에 포메터를 적용하여 반환한다.
com.transDate = function(str, type)

// 문자열 왼쪽에 일정길이(maxLen) 만큼 '0'으로 채우기
com.fillZero = function(str, maxLen)

// JSON Object로 변환해서 반환한다.
com.getJSON = function(str)

// JSON Object인지 여부를 검사한다.
com.isJSON = function(jsonObj)

// XML Document 객체인지 여부를 검사한다.
com.isXmlDoc = function(data)

// Object가 null, undefined, "", {}, []일 경우 true 이외의 경우는 false를
return한다.
com.isEmpty = function(poObj)

// Object가 null, undefined, "", {}, []일 경우 false 이외의 경우는 true를
return한다.
com.isNotEmpty = function(poObj)

// Object가 null, undefined, "", {}, []일 경우 대체Object를 return한다.
com.ifEmpty = function(poObj, poConvObj)

// JSON 객체를 String 타입으로 반환한다.
com.strSerialize = function(object)
```

⑦ 기타

```
// 해당 그룹 안의 컴포넌트에서 엔터키가 발생하면 해당 컴포넌트의 값을 DataColletion에
저장하고 objFunc 함수를 실행한다.
com.setEnterKeyEvent = function(grpObj, objFunc)

// 파라미터를 읽어 온다.
com.getParameter = function(paramKey)

// contextRoot가 포함된 path를 반환한다.
com.getFullPath = function(path)
```

※ Websquare API 는 아래 링크 참고

http://11.11.0.160:8000/test1/index.html

7. TestCase

- JUnitTest

7.1 Controller URL 호출 테스트

```
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;
import javax.inject.Inject;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"/dispatcher-servlet.xml", "/spring/context-*.xml"})
@WebAppConfiguration
public class TestCase1
{
    @Inject
    private WebApplicationContext wac;

    private MockMvc mock;

    @org.junit.Before
    public void setup()
    {
        this.mock = MockMvcBuilders.webAppContextSetup(this.wac).build();
    }

    @Test
    public void myTest() throws Exception
    {
        mock.perform(MockMvcRequestBuilders.get("/rba/selectUserList.do"));
    }
}
```


7.2 Service 호출 테스트

```
import java.util.HashMap;
import java.util.Map;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;

import mil.dhrmis.sys.rba.service.UserMgtService;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"/dispatcher-servlet.xml", "/spring/context-*.xml"})
@WebAppConfiguration
public class TestCase2
{
    @Autowired
    private UserMgtService service;

    @Test
    public void myTest() throws Exception
    {
        // 사용자 정의 Parameter
        Map<String, Object> param = new HashMap<String, Object>();

        // 서비스 호출
        service.selectUserList(param);
    }}
```

7.3 DataSet (DataCollection, DataMap) 테스트

```
@Test
public void test1() throws Exception
{
    // ------------------------------------------------------------------------------------------------------------
    // DataMap 테스트
    // ( {"empId":"U0000001","empNm":"dp.kim"}; )
    // ------------------------------------------------------------------------------------------------------------
    JSONObject jsonObj = new JSONObject();
    jsonObj.put("empId", "U0000001");
    jsonObj.put("empNm", "dp.kim");
    String jsonStr = jsonObj.toJSONString();

    ObjectMapper mapper = new ObjectMapper();
    Emp data = mapper.readValue(jsonStr, Emp.class);
    System.out.println("test1..... : " + data.getEmpId());
}

@Test
public void test2() throws Exception
{
    // ------------------------------------------------------------------------------------------------------------
    // DataCollection 테스트 (List<VO>로 변환 테스트)
    // ( [{"empId":"U0000001","empNm":"dp.kim"}, {"empId":"U0000002","empNm":"ms.kim"}]; )
    // ------------------------------------------------------------------------------------------------------------
    JSONArray jsonArray = new JSONArray();
    JSONObject jsonObj1 = new JSONObject();
    jsonObj1.put("empId", "U0000001");
    jsonObj1.put("empNm", "dp.kim");
    jsonArray.add(jsonObj1);
    JSONObject jsonObj2 = new JSONObject();
    jsonObj2.put("empId", "U0000002");
    jsonObj2.put("empNm", "ms.kim");
    jsonArray.add(jsonObj2);
    String jsonStr = jsonArray.toJSONString();

    ObjectMapper mapper = new ObjectMapper();
    Emp[] dataList = mapper.readValue(jsonStr, Emp[].class);
    System.out.println("test2..... : " + dataList[1].getEmpId());
}

@Test
public void test3() throws Exception
{
    // ------------------------------------------------------------------------------------------------------------
    // DataCollection 테스트 (List<Map>으로 변환 테스트)
    // ( {"userList":[{"empId":"U0000001","empNm":"dp.kim"}, {"empId":"U0000002","empNm":"ms.kim"}]}; )
    // ------------------------------------------------------------------------------------------------------------
    JSONObject jsonObj = new JSONObject();
    JSONArray jsonArray = new JSONArray();
    JSONObject jsonObj1 = new JSONObject();
    jsonObj1.put("empId", "U0000001");
    jsonObj1.put("empNm", "dp.kim");
    jsonArray.add(jsonObj1);
    JSONObject jsonObj2 = new JSONObject();
    jsonObj2.put("empId", "U0000002");
    jsonObj2.put("empNm", "ms.kim");
    jsonArray.add(jsonObj2);
    jsonObj.put("userList", jsonArray);
    String jsonStr = jsonObj.toJSONString();
    System.out.println(jsonStr);

    ObjectMapper mapper = new ObjectMapper();
    Object data = mapper.readValue(jsonStr, new TypeReference<Map<String, Object>>(){});

    Map mapData = (Map<String, Object>)data;
    List lstData = (List)mapData.get("userList");

    System.out.println("test3..... : " + ((Map)lstData.get(0)).get("empId"));
}
```

7.4 DOM 테스트

```
import static org.junit.Assert.assertTrue;

import java.util.Map;
import java.util.Objects;

import org.apache.commons.beanutils.BeanUtils;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;

// JAXP API
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

// Exception
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

// W3C DOM Definition
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.Entity;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;

import java.io.File;
import java.io.PrintWriter;

@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
/*
 * DOMTest
 */
public class DOMTest
{
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Test
    public void test1() throws Exception
    {
        String fileName = "test.xml";
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.parse(new File(fileName));
        for(Node child = doc.getFirstChild(); child != null; child = child.getNextSibling())
        {
            echo(child);
        }
    }

    private void echo(Node n)
    {
        int type = n.getNodeType();
        switch(type)
        {
            case Node.ATTRIBUTE_NODE:
                System.out.println("ATTR:");
                printlnCommon(n);
                break;

            case Node.CDATA_SECTION_NODE:
                System.out.println("CDATA:");
                printlnCommon(n);
                break;
```

```
            case Node.COMMENT_NODE:
                System.out.println("COMM");
                printlnCommon(n);
                break;

            case Node.DOCUMENT_FRAGMENT_NODE:
                System.out.println("DOC_FRAG:");
                printlnCommon(n);
                break;

            case Node.DOCUMENT_NODE:
                System.out.println("DOC:");
                printlnCommon(n);
                break;

            case Node.DOCUMENT_TYPE_NODE:
                System.out.println("ELEM:");
                printlnCommon(n);
                NamedNodeMap nodeMap = ((DocumentType)n).getEntities();
                for(int i = 0; i < nodeMap.getLength(); i++)
                {
                    Entity entity = (Entity)nodeMap.item(i);
                    echo(entity);
                }
                break;

            case Node.ELEMENT_NODE:
                System.out.println("ELEM:");
                printlnCommon(n);
                NamedNodeMap atts = n.getAttributes();
                for(int i = 0; i < atts.getLength(); i++)
                {
                    Node att = atts.item(i);
                    echo(att);
                }
                break;

            case Node.ENTITY_NODE:
                System.out.println("ENT:");
                printlnCommon(n);
                break;

            case Node.ENTITY_REFERENCE_NODE:
                System.out.println("ENT_REF:");
                printlnCommon(n);
                break;

            case Node.NOTATION_NODE:
                System.out.println("NOTATION:");
                printlnCommon(n);
                break;

            case Node.PROCESSING_INSTRUCTION_NODE:
                System.out.println("PROC_INST:");
                printlnCommon(n);
                break;

            case Node.TEXT_NODE:
                System.out.println("TEXT:");
                printlnCommon(n);
                break;

            default:
                System.out.println("UNSUPPORTED NODE : " + type);
                printlnCommon(n);
                break;
        }
        for(Node child = n.getFirstChild(); child != null; child = child.getNextSibling())
        {
            echo(child);
        }
    }
```

```
    private void printlnCommon(Node n)
    {
        System.out.print(" nodeName=\"" + n.getNodeName() + "\"");

        String val = n.getNamespaceURI();
        if(val != null)
        {
            System.out.print(" uri=\"" + val + "\"");
        }

        val = n.getPrefix();
        if(val != null)
        {
            System.out.print(" pre=\"" + val + "\"");
        }

        val = n.getLocalName();
        if(val != null)
        {
            System.out.print(" local=\"" + val + "\"");
        }

        val = n.getNodeValue();
        if(val != null)
        {
            System.out.print(" nodeValue=");
            if(val.trim().equals(""))
            {
                // Whitespace
                System.out.print("[WS]");
            }
            else
            {
                System.out.print("\"" + n.getNodeValue() + "\"");
            }
        }
        System.out.println();
    }
}
```


7.5 SAX 테스트

```
import java.io.PrintStream;
import java.util.Enumeration;
import java.util.Hashtable;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

// JAXP API
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

// SAX Parser
import org.xml.sax.Attributes;

// Exception
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class SAXTest extends DefaultHandler
{
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    private Hashtable tags;
```

```
    public static void main(String[] args) throws Exception
    {
        String fileUrl = "file:///c:/DHRMISIDE/workspace/DHRMIS/src/resources/spring/context-
app.xml";

        SAXParserFactory spf = SAXParserFactory.newInstance();
        spf.setNamespaceAware(true);
        SAXParser saxParser = spf.newSAXParser();

        SAXTest saxTest = new SAXTest();
        XMLReader xmlReader = saxParser.getXMLReader();
        xmlReader.setContentHandler(saxTest);
        xmlReader.parse(fileUrl);

        // System.out.println("..... " + saxTest.tags);
    }

    public void startDocument() throws SAXException
    {
        tags = new Hashtable();
    }

    public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
throws SAXException
    {
        String key = localName;
        Object value = tags.get(key);

        if(value == null)
        {
            tags.put(key, new Integer(1));
        }
        else
        {
            int count = ((Integer)value).intValue();
            count++;
            tags.put(key, new Integer(count));
        }

        System.out.println(key);
        int attrCount = atts.getLength();
        for(int i = 0; i < attrCount; i++)
        {
            String qName1 = atts.getQName(i);
            String value1 = atts.getValue(i);
            // String uri1 = atts.getURI(i);
            // String lName1 = atts.getLocalName(i);
            System.out.println("   " + qName1 + "=" + value1);
        }
    }

    public void endDocument() throws SAXException
    {
        Enumeration e = tags.keys();
        while(e.hasMoreElements())
        {
            String tag =  (String)e.nextElement();
            int count = ((Integer)tags.get(tag)).intValue();
        }
    }

    private class MyErrorHandler implements ErrorHandler
    {
        private PrintStream out;
        MyErrorHandler(PrintStream out)
        {
            this.out = out;
        }
        private String getParseExceptionInfo(SAXParseException spe)
        {
            String systemId = spe.getSystemId();
```

```
            if(systemId == null)
            {
                systemId = "null";
            }
            String info = "URI=" + systemId + " Line=" + spe.getLineNumber() + ": " +
spe.getMessage();
            return info;
        }

        @Override
        public void warning(SAXParseException spe) throws SAXException
        {
            System.out.println("Warning : " + getParseExceptionInfo(spe));
        }

        @Override
        public void error(SAXParseException spe) throws SAXException
        {
            String message = "Error : " + getParseExceptionInfo(spe);
            throw new SAXException(message);
        }

        @Override
        public void fatalError(SAXParseException spe) throws SAXException
        {
            String message = "Fatal Error : " + getParseExceptionInfo(spe);
            throw new SAXException(message);
        }
    }
}
```


7.6 POI 테스트

```
```

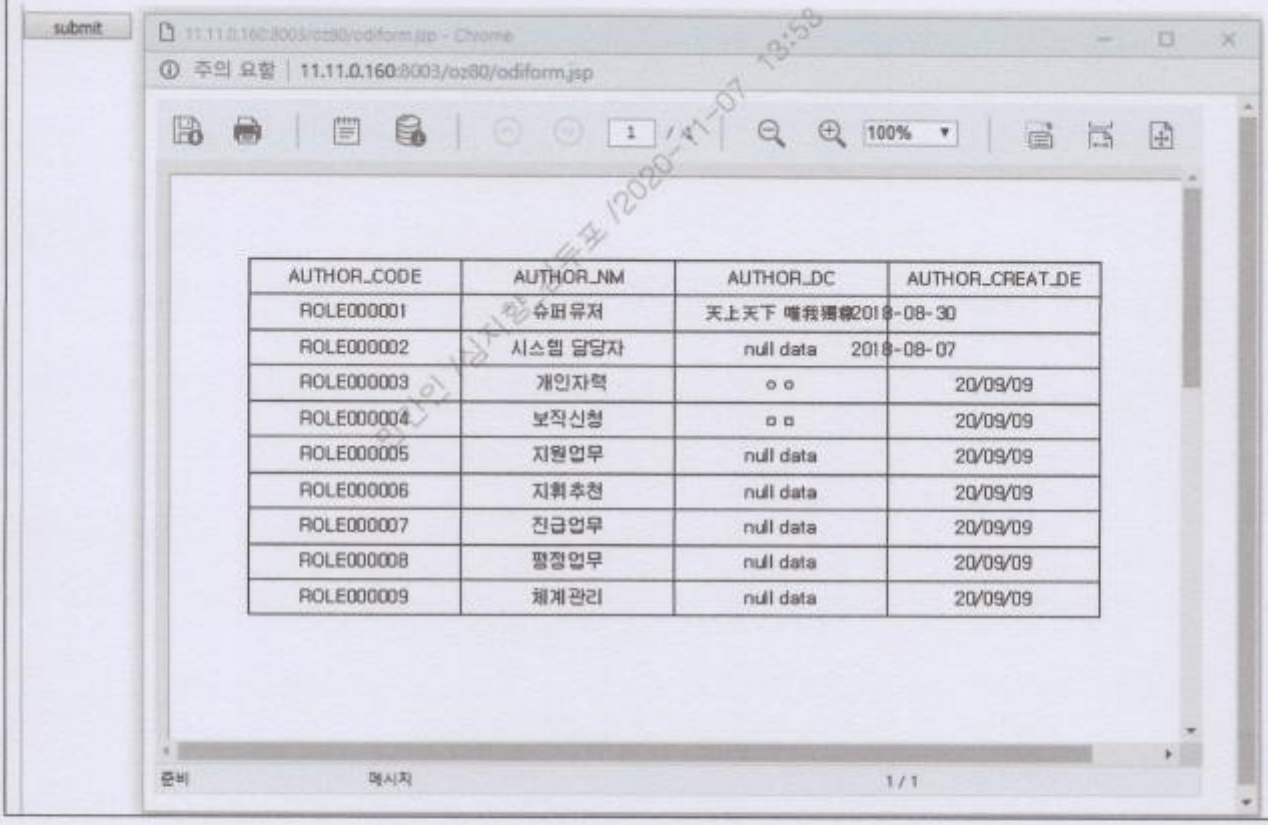
8. 솔루션

8.1 OZ Report

가. 레포트 샘플

ODI 샘플

```
<body ev:onpageload="scwin.onpageload" ev:onpageunload="scwin.onpageunload">
    <div id='test1' style='width:753px;height:701px;'></div>
</body>


scwin.onpageload = function()
{
    var opt = {};
    opt.filename = 'B6,D5,DB,CD,27,67,D6,3E,23,E6,FB,B1,34,81,5D,47,9F,EF,4C,7F,A3,4F,E1,23,F5,C9,55,30,6B,DD,EC,23';
    opt.url = '/sys/exp/tst/selectJsonTest2.do';
    opt.param1 = '13,2B,FA,3C,04,60,9A,0B,A2,F6,B9,E2,82,83,9B,2A';

    com.report.odiForm('test1', opt);
};
```

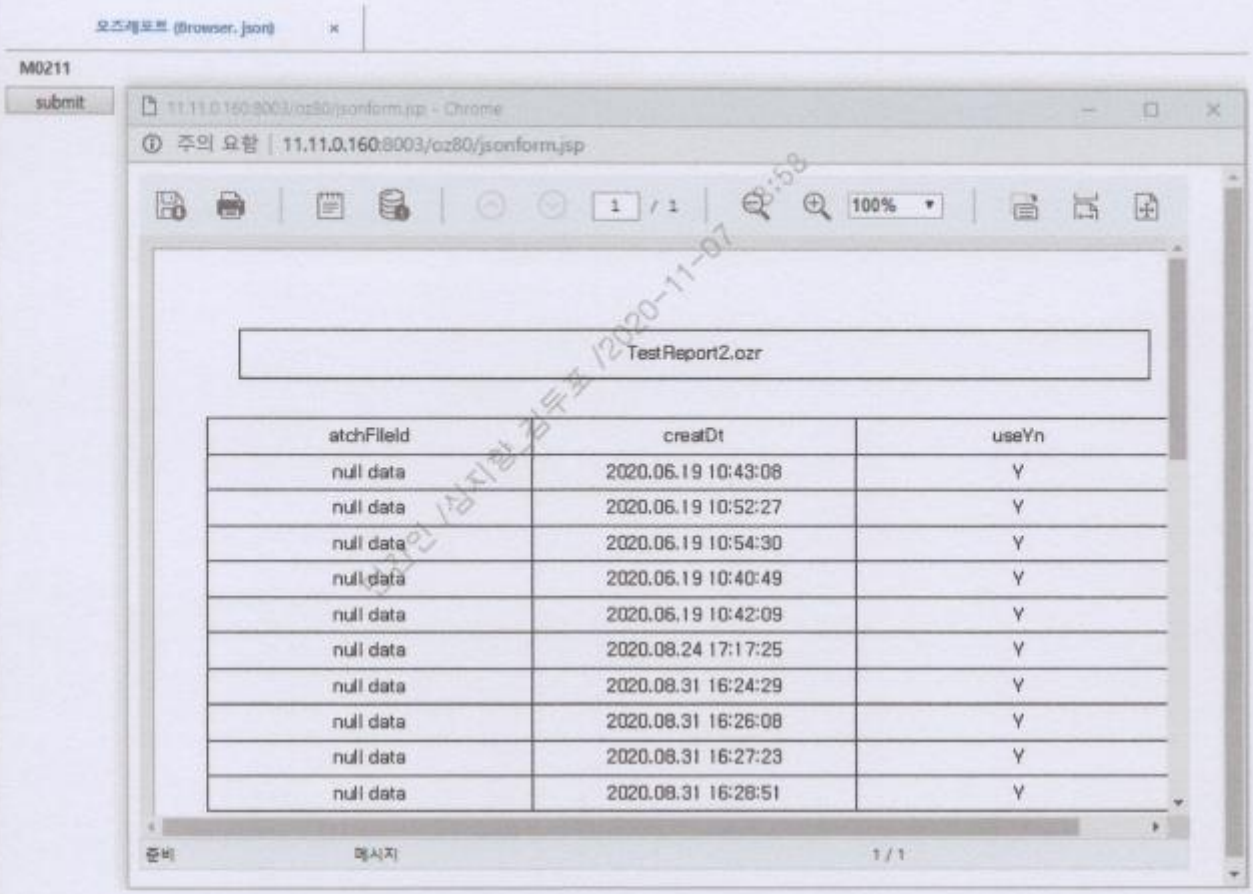


AUTHOR_CODE	AUTHOR_NM	AUTHOR_DC	AUTHOR_CREAT_DE
ROLE000001	슈퍼유저	天上天下 唯我獨尊2018-08-30	
ROLE000002	시스템 담당자	null data	2018-08-07
ROLE000003	개인자력	ㅇㅇ	20/09/09
ROLE000004	보직신청	ㅁㅁ	20/09/09
ROLE000005	지원업무	null data	20/09/09
ROLE000006	지휘추천	null data	20/09/09
ROLE000007	진급업무	null data	20/09/09
ROLE000008	평정업무	null data	20/09/09
ROLE000009	체계관리	null data	20/09/09

JSON 샘플

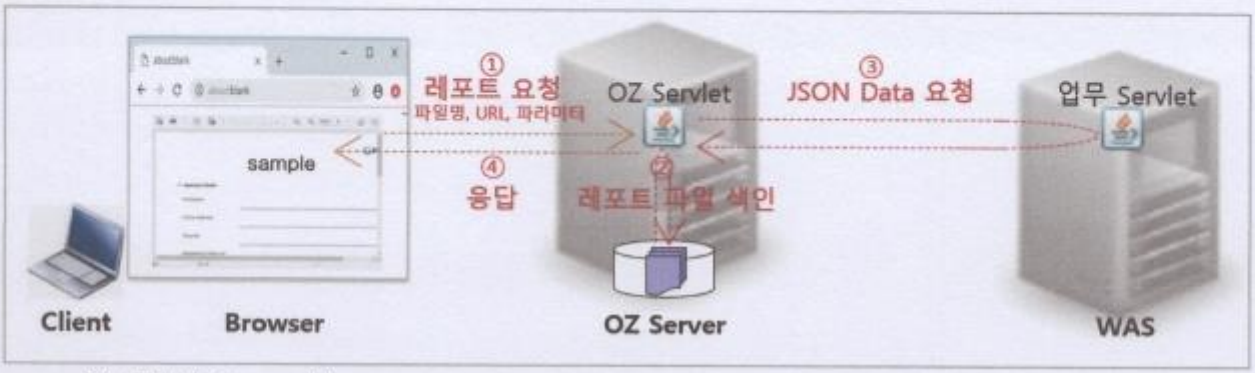
```
<body ev:onpageload="scwin.onpageload" ev:onpageunload="scwin.onpageunload">
    <div id='test1' style='width:753px;height:701px;'></div>
</body>
```

```
scwin.onpageload = function()
{
    var opt = {};
    opt.filename = 'EC,FA,54,A4,21,7D,D8,2A,00,4F,20,88,4B,0C,3A,DC,9F,EF,4C,7F,A3,4F,E1,23,F5,C9,55,30,6B,DD,EC,23';
    opt.url = '/sys/exp/tst/selectJsonTest4.do';
    opt.param1 = '06,5A,F2,66,2C,91,24,91,5B,9C,6F,7B,10,E9,87,0C';

    com.report.jsonForm('test1', opt);
};
```



나. Client Report 생성 요청

- 개념도



(1) HTML5 Canvas Viewer

- HTML5 <canvas> 이용해서 Report 생성
- 서버에서 서식파일과 Data 을 따로 내려주어 Client 에서 조립하여 본다.
- Client 가 적당한 부하를 감수하고자 할 때 사용.
- Data 10 ~ 50 건 사이 정도 의 데이터 처리시에 사용 권장.
- 예제

```
<script type="text/javascript" src="/oz80/ozhviewer/OZJSViewer.js" charset="utf-8"></script>
<script type="text/javascript" >                    → Canvas Viewer Script 선언
    var jsonStr = '{"testData":[{"key":"a1", "name":"b1"},{"key":"a2", "name":"b2"}]}';
    function SetOZParamters_OZViewer() {
        var oz = document.getElementById("OZViewer");
        oz.sendToActionScript("connection.servlet","http://11.11.0.160:8000/oz80/server");
        oz.sendToActionScript("connection.reportname","/sample.ozr"); → OZ Report 파일 위치
        oz.sendToActionScript("toolbar.save", "false");
        oz.sendToActionScript("toolbar.print", "false"); → 인쇄 미리보기 화면에서 저장/인쇄 버튼 조정
        oz.sendToActionScript("export.applyformat", "xls,xlsx,doc"); → 미리보기에서 저장가능 타입 지
        return true;
    }
    start_ozjs("OZViewer","/oz80/ozhviewer/");
    </script>
```

(2) HTML5 SVG Viewer

- HTML5 <svg> 이용해서 Report 생성.
- 서버에서 서식파일과 Data 을 조립하여 Client 에 내려준다.
- 50 건 이상의 데이터 처리시에 사용 권장.
- 예제/oz80/ozhviewer/OZJSSVGViewer.js

```
<script type="text/javascript" src="/oz80/ozhviewer/OZJSSVGViewer.js" charset="utf-8"/>
<script type="text/javascript" >          → SVG Viewer Script 선언
    var jsonStr = '{"testData":[{"key":"a1", "name":"b1"},{"key":"a2", "name":"b2"}]}';
    function SetOZParamters_OZViewer() {
        var oz = document.getElementById("OZViewer");
        oz.sendToActionScript("connection.servlet","http://11.11.0.160:8000/oz80/server");
        oz.sendToActionScript("connection.reportname","/sample.ozr");  → OZ Report 파일 위치
        oz.sendToActionScript("toolbar.save", "false");
        oz.sendToActionScript("toolbar.print", "false");  → 인쇄 미리보기 화면에서 저장/인쇄 버튼 조정
        oz.sendToActionScript("export.applyformat", "xls,xlsx,doc");  → 미리보기에서 저장가능 타입 지
        return true;
    }
    var opt = [];
    opt["rendering_mode"] = "svg";  →SVG로 처리하기 위한 옵션
    start_ozjs("OZViewer","/oz80/ozhviewer/");
    </script>
```

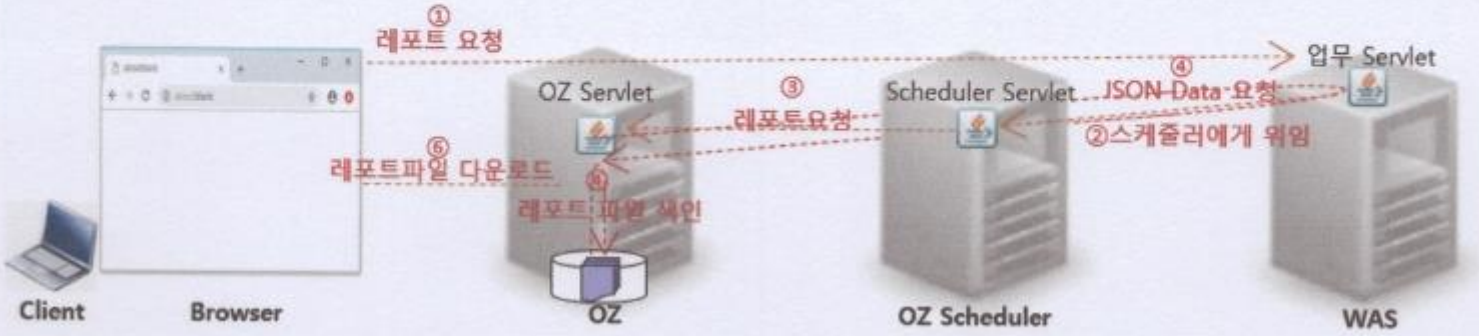
(3) Exe Viewer

- Exe CS Program 을 이용

※ 표준웹버전으로 변경되면서 접근성 측면에선 좋아졌으나, 속도면에서는 더 느려졌다.
버전이 업그레이드 되었다고 속도가 더 빨라졌을거라는 오해 하지마세요.

다. Server Report 생성 요청

- 개념도



- 예제

```
String ozServerUrl = "http://11.11.0.160:8002/oz80";
String ozAdmin = "admin";
String ozPasswd = "password";

ServerInfo serverInfo = new ServerInfo();
serverInfo.setID(ozAuthId);
serverInfo.setPWD(ozAuthPw);
serverInfo.setIsDaemon(false);
serverInfo.setURL(ozServerUrl);

SortProperties configMap = new SortProperties();
configMap.setProperty("task_type", "viewerTag");
configMap.setProperty("launch_type", "immediately");
configMap.setProperty("cfg.type", "new");

SortProperties exportMap = new SortProperties();
exportMap.setProperty("connection.servlet", ozServerUrl);
exportMap.setProperty("connection.reportname", "overview.ozr");
exportMap.setProperty("export.format", "pdf");
exportMap.setProperty("pdf.filename", "test-report");

    exportMap.setProperty("child1.connection.servlet", ozServerUrl);
    exportMap.setProperty("child1.connection.reportname", "testreport1.ozr");
    exportMap.setProperty("child1.export.format", "pdf");
    exportMap.setProperty("child1.pdf.filename", "test-child-report.pdf");
멀티보고서
    exportMap.setProperty("child2.connection.servlet", ozServerUrl);
    exportMap.setProperty("child2.connection.reportname", "testreport2.ozr");
    exportMap.setProperty("child2.export.format", "pdf");
    exportMap.setProperty("child2.pdf.filename", "test-child-report.pdf");

ozScheduler.directExport(serverInfo, configMap, exportMap);
```

※ 하나의 PDF 로 Merge 하고자 할때는 exportMap.setProperty("global.concatpage", "true")와 같이 작성.

※ 개별 PDF 로 나눠 출력하고자 하면 exportMap.setProperty("global.concatpage", "false")로 작성한다.

※ 각 레포트에서 데이터가 많을 경우 concurrent 를 주고, pageque 를 적정하게 나눠준다.

※ 데이터를 요청하는 방법으로 여러가지가 있는데, 크게 업무서비스 URL을 호출하거나, 직접 쿼리를 호출하거나 하는 2가지 방법이 있다. (다음장에서 논의.)

※ JSON vs ODI

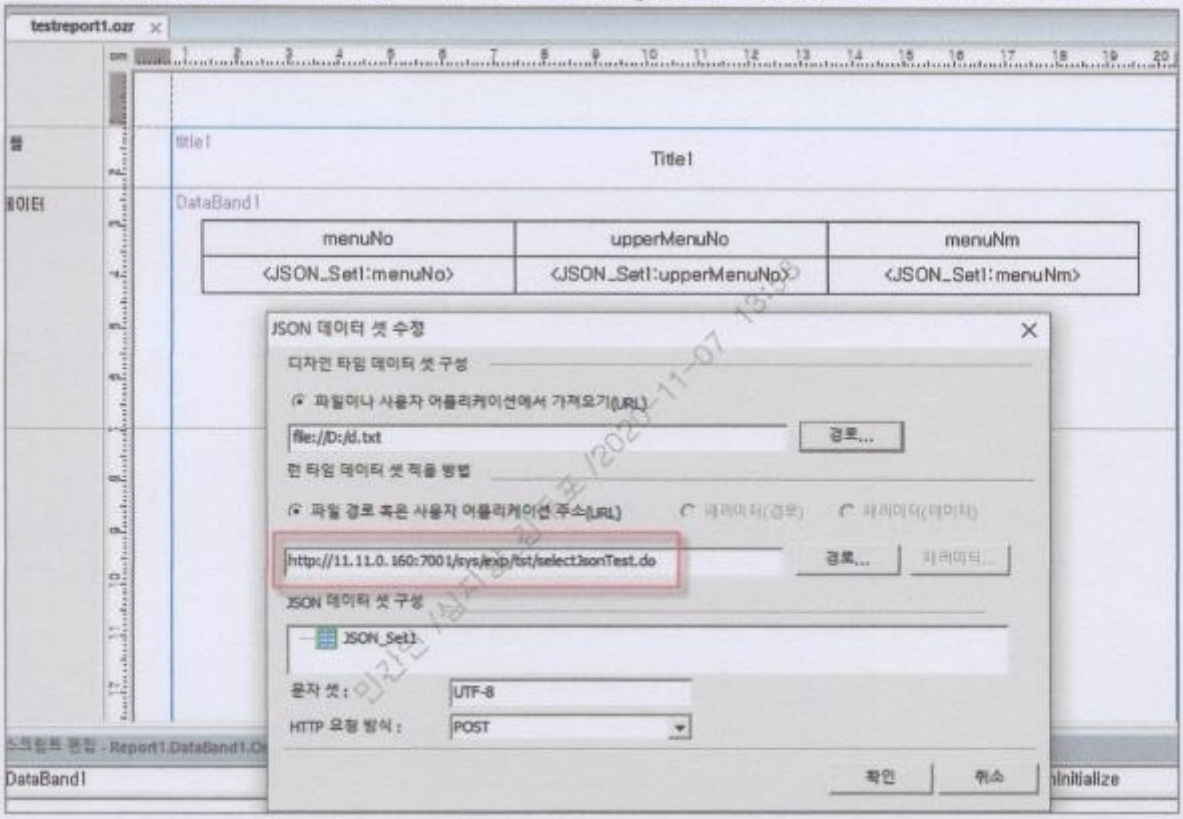
1. JSON 방식

- 업무 URL 호출.
- 장점

  업무쿼리, 레포트 쿼리 2벌 작성 필요 없음.

  레포트가 유출될시 보안에 위험.
- 단점

  레포트 서블릿이 과부하 걸릴경우 업무WAS가 Hang걸릴수 있음. 별도의 Container 분리 고려.

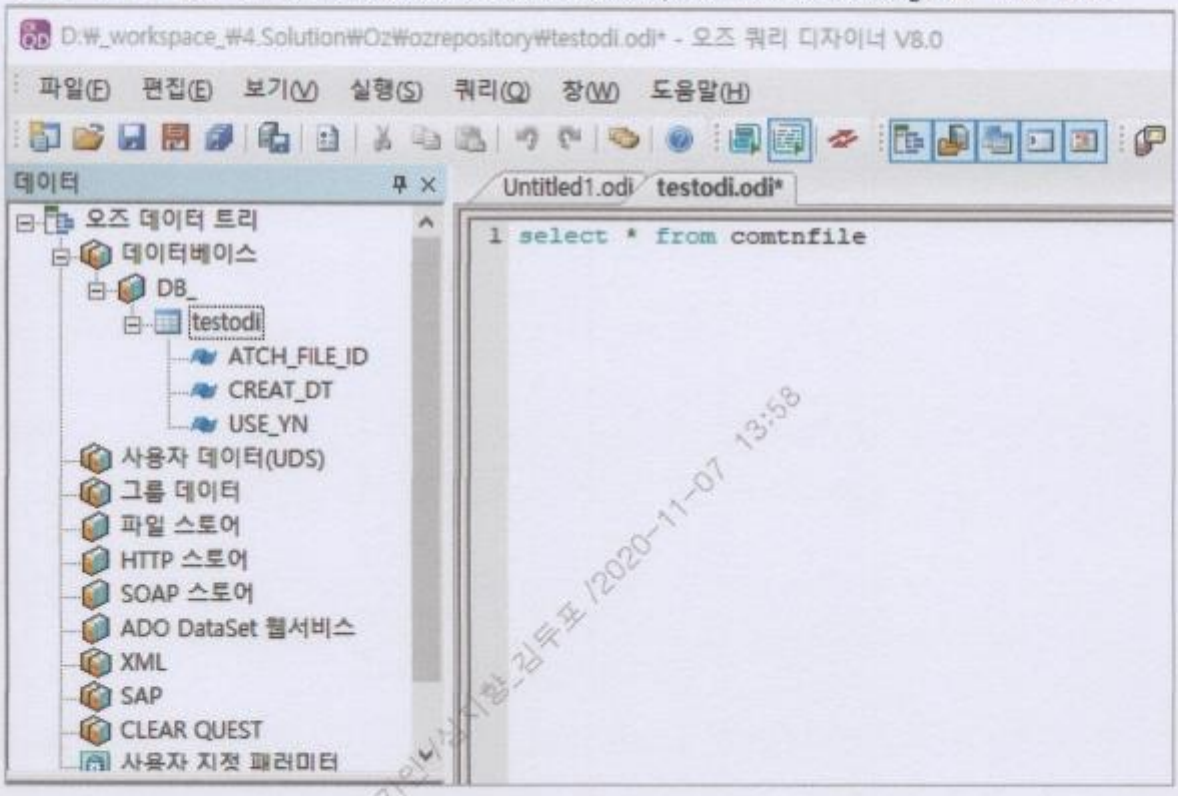


2. ODI 방식

- OZR 파일에 SQL 작성.
- 장점 :

  ASIS 소스를 이용할수 있다함.
- 단점

  OZR 파일 호출시 odi.odinames, odi.testodiodinames.pcount, odi.tesodi.args 등 작성 필요



8.2 MarkAny

- OZ 는 Report 에 Water Mark 인쇄하는 기능
- OZ Report 호출하는 스크립트에 Option 추가하는 형태로 연동됨.

```
<script type="text/javascript" src="/oz80/ozhviewer/OZJSSVGViewer.js" charset="utf-8"></script>
<script type="text/javascript" >
    var param = '{"searchTitle":"test"}';

    function SetOZParamters_OZViewer() {
        var oz;
        oz = document.getElementById("OZViewer");
        oz.sendToActionScript("connection.servlet","http://11.11.0.160:8000/oz80/server");
        oz.sendToActionScript("connection.reportname","/sample.ozr");
        oz.sendToActionScript("toolbar.save", "false");
        oz.sendToActionScript("toolbar.print", "false");
        oz.sendToActionScript("export.applyformat", "xls,xlsx,doc");

        oz.sendToActionScript("print.externalmodule","oz.viewer.export.OZAppletPrintBarcode_MarkAny");
        oz.sendToActionScript("connection.extraparam","PropertiesFilePath=/u01/jeus/webapps/oz80/WEB-
INF/conf/markany.properties,UserPassword=0,PdfCreator=MarkAny");

        return true;
    }

    var opt = [];
    opt["print_exportfrom"] = "scheduler";
    start_ozjs("OZViewer","/oz80/ozhviewer/", opt);
</script>
```

MarkAny 처리

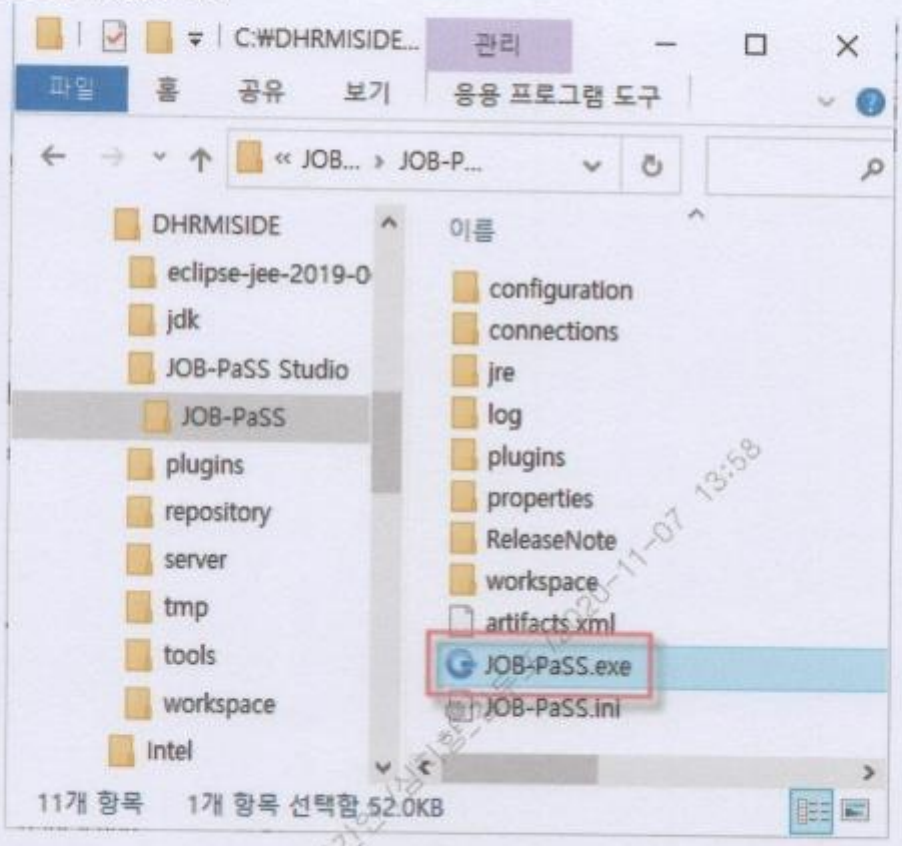
※ 레포트 뷰어에서는 바코드가 안나오고, 인쇄버튼을 눌러 미리보기시 바코드가 출력됨.
솔루션 업체에서 제공하는 바코드스캔 앱을 통해 위변조된 레포트인지 원본인지 확인 가능.

8.3 JOB-PaSS

- Eclipse Framework 기반 솔루션

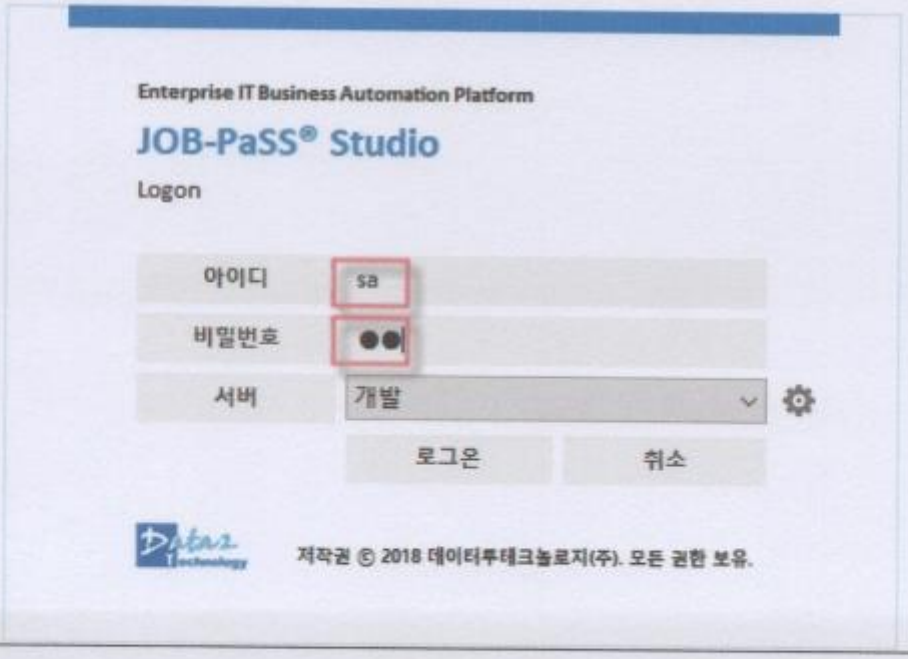
가. 실행

JOB-PsSS.exe 실행



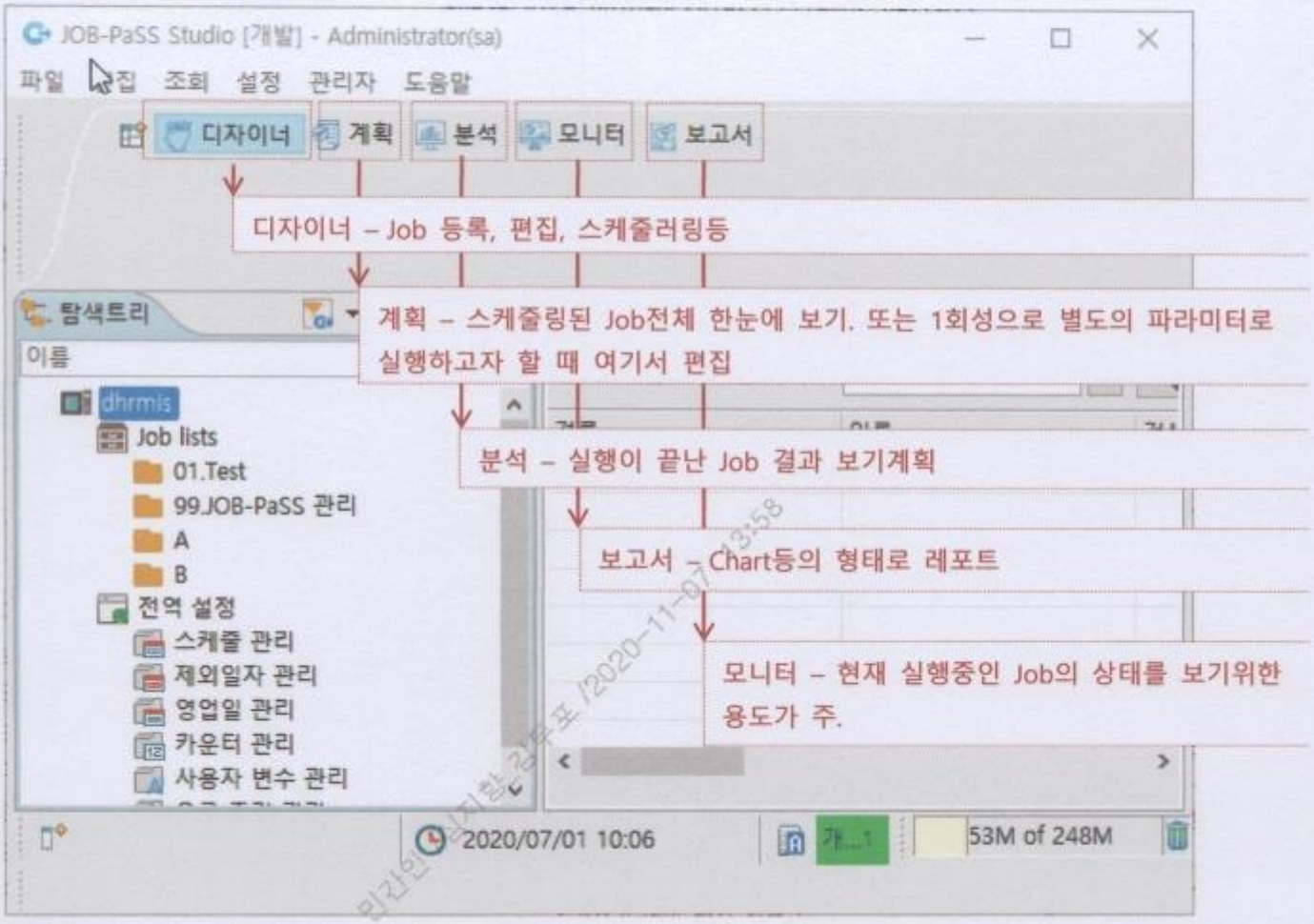
나. 로그인

sa / sa 로 로그인



다. 개요

- Perspective 는 총 가지로 디자이너, 계획, 분석, 모니터, 분석이 있고 이 중 설계자나 개발자들은 주로 디자이너를 사용하게 됨.



※ 이 외에 관리자 기능에

접속정보 설정 – Scheduler Agent 등록/편집/삭제

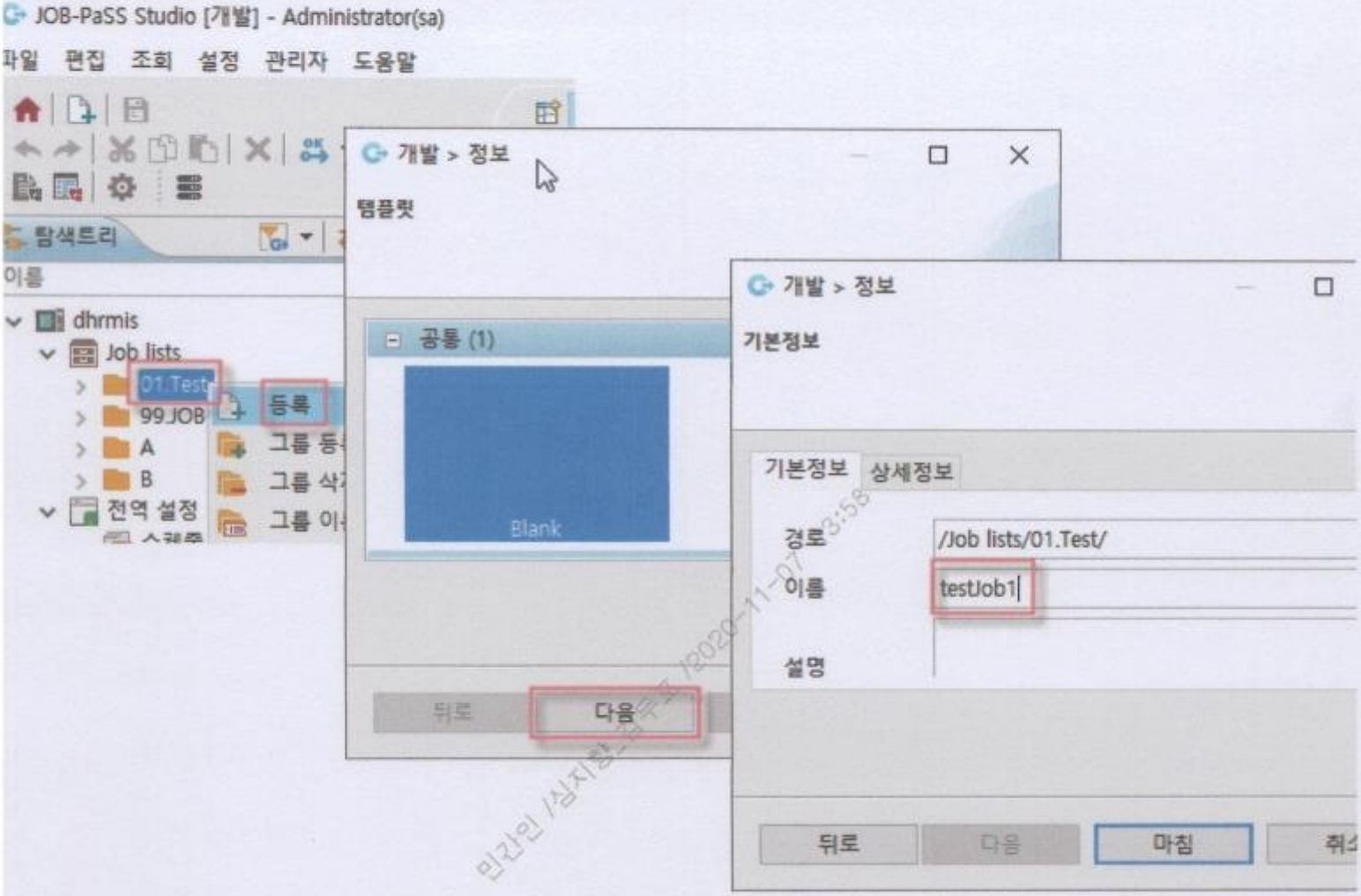
서버설정 - Job 들이 실행후 남기는 로그들 삭제 Rolling 주기 설정등

사용자권한 설정 – 해당 로그인 사용자가 사용할수 있는 컴포넌터 종류 설정등 (Job 종류)

등의 기능이 있으나 공통단에서 처리할 예정임.

라. Job 생성

- job_lists 선택 > 해당 업무그룹 선택 > 등록

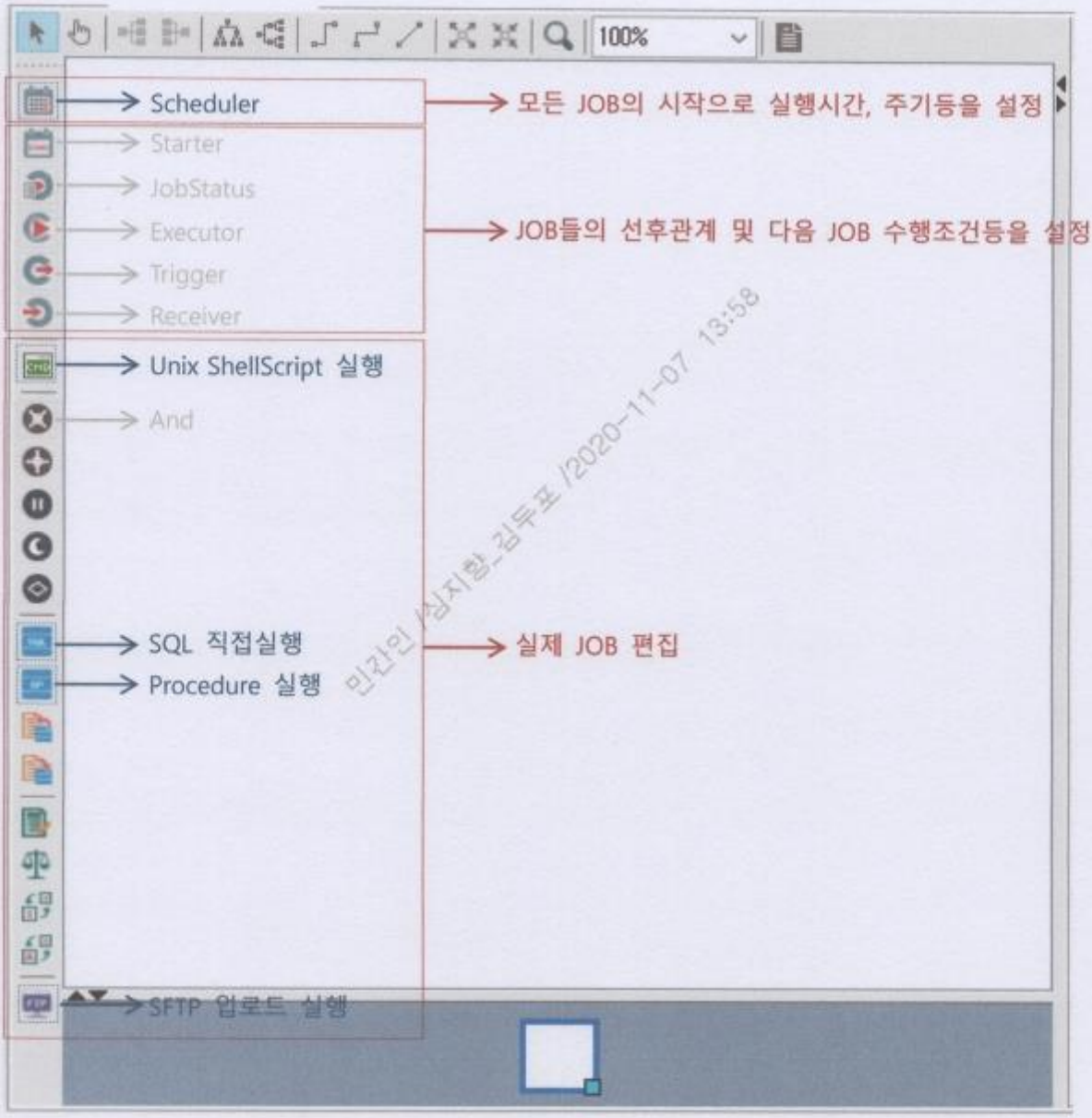


마. Job 편집

1. Job 편집시 시작노드는 **Scheduler, Starter, Receiver** 셋 중 하나인데, 대부분 Scheduler에서 시작.
2. Job의 종류로는 **SQL, StoredProcedure, Unix ShellScript, SFTP** 등이 있다.
3. Starter, JobStatus, Executor, Trigger, Receiver, And는 각 Job들의 선후관계 및 조건등을 걸고자 사용하는 기능으로 필요시 Executor를 사용하기를 권장.

※ Version Up History : Trigger/Receiver > JobStatus > Excutor

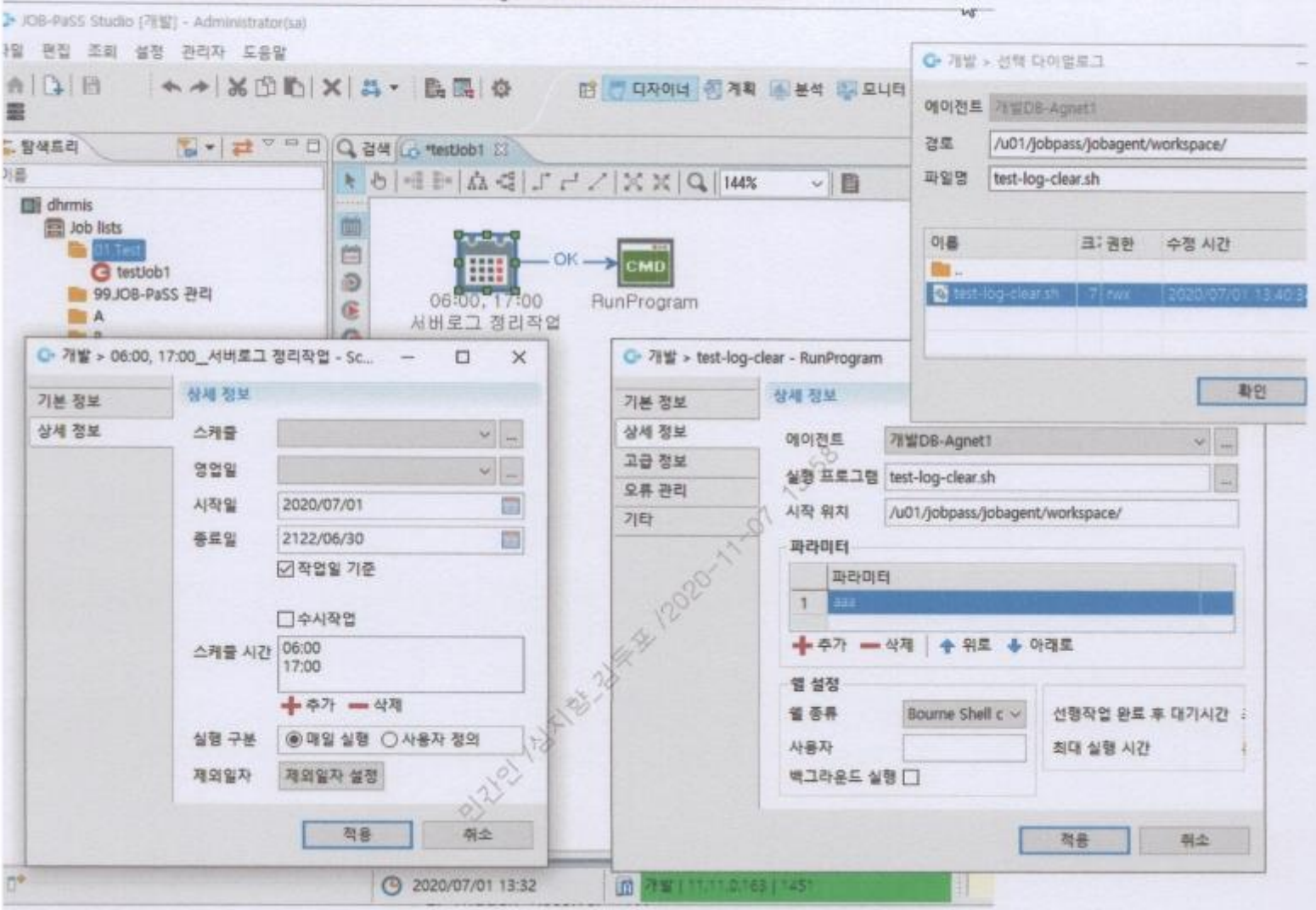
※



바. 예제

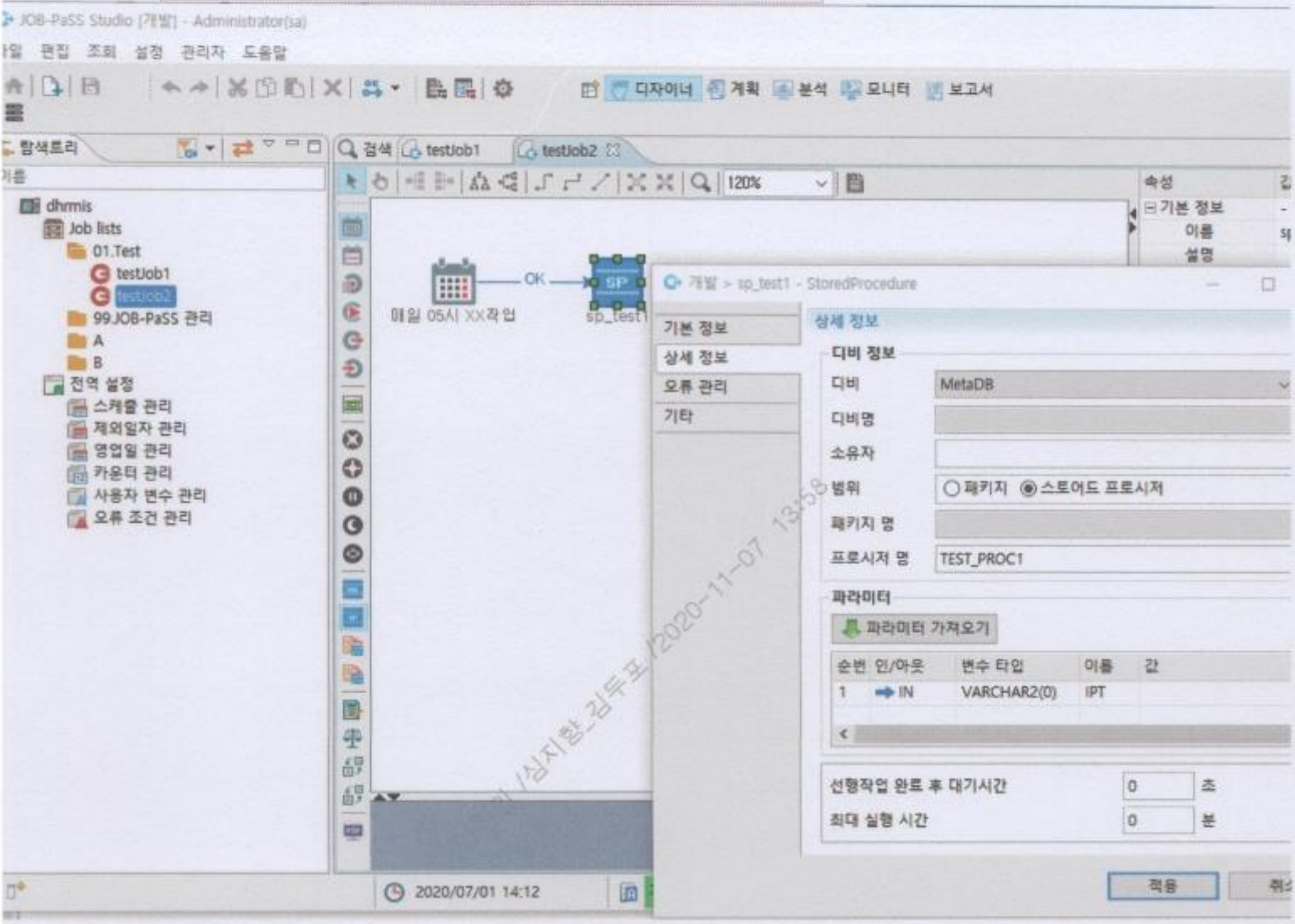
(1) Unix ShellScript 등록

- 스케줄러 등록 > Run Program 등록 > 링크 연결

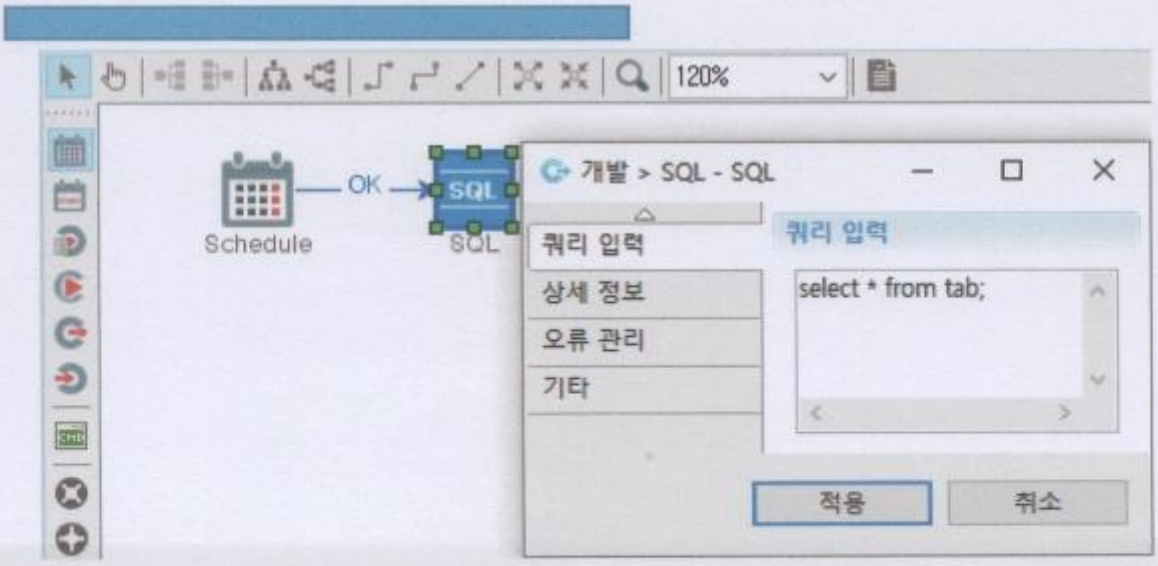


(2) Procedure 실행

- 스케줄러 등록 > StoredProcedure 등록 > 링크 연결

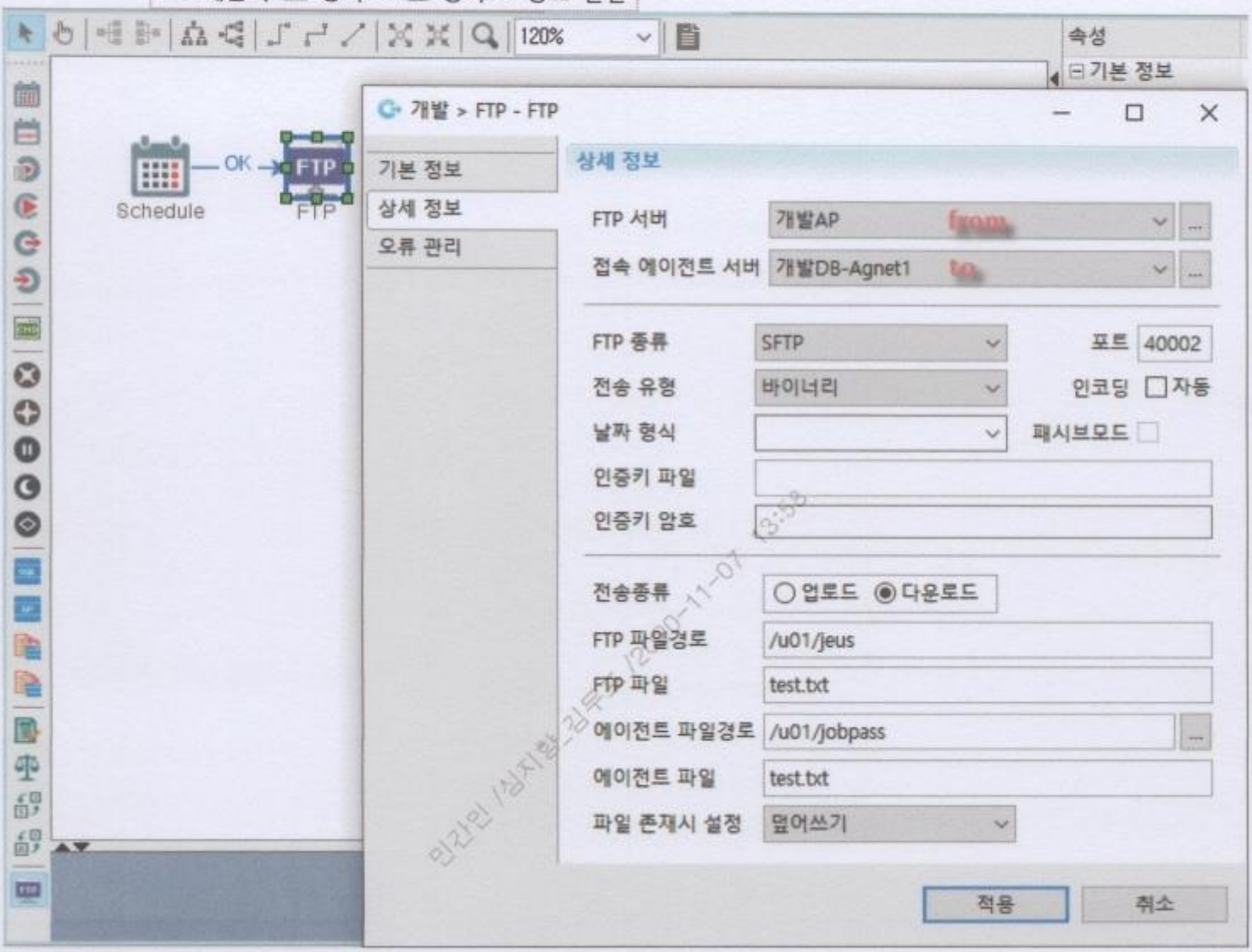


(3) SQL 직접실행



(4) SFTP 실행

- 스케줄러 등록 > FTP 등록 > 링크 연결

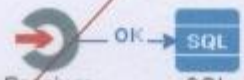


(5) 조건 설정

- 2 개이상 Job 전후관계, 조건등 설정

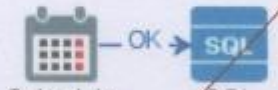
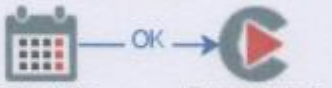
1. Trigger/ Receiver 이용

- Trigger 와 Receiver 가 Set 로 동작한다.
- 선행 작업은 끝에 Trigger 로 끝을 내고, 본작업은 Receiver 로 시작한다.
- 1 대 1 로만 작동하고, 사실상 Deprecaed 된 기능으로 쓰지 않을것을 권장.

선행작업	본 작업
	

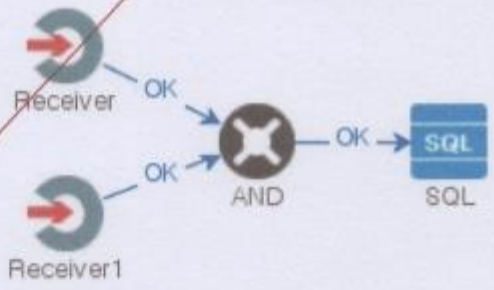
2. Executor 사용

- Executor 와 Starter 가 Set 동작한다.
- 현재 Job 의 Flow 중 한 노드에 다른 Job 을 등록하고자 할 때 Executor 를 사용하는데, 실제 Executor 가 될 Job 을 생성시 앞에 딱히 둘게 없어 Starter 를 둔 형태.

선행작업	본 작업
	

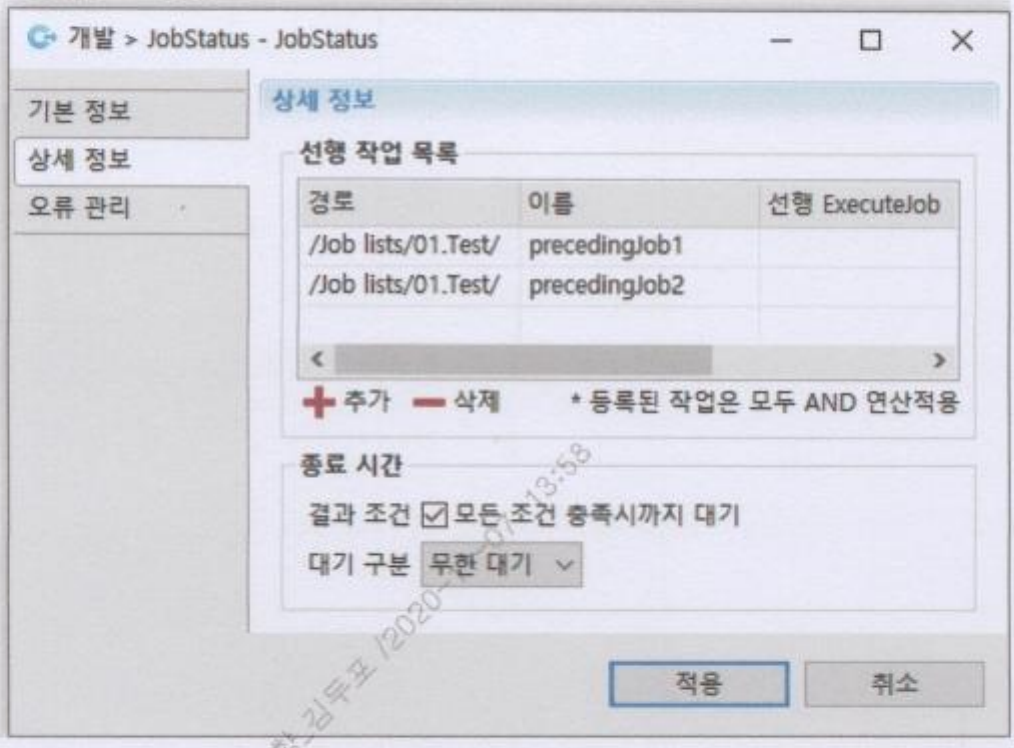
3. AND 이용

- 2 가지 이상의 작업이 끝나야 본작업이 실행되야 할 경우 AND 를 사용



4. JobStatus 사용

- AND 보다 업그레드 된 형태로 JobStatus 안에 선행되야 할 Job 목록을 등록해서 사용.



9. 시큐어 코딩

9.1 전자정부 Code Inspection RuleSet 39 개

No.	Rule 이름	비고
1	EmptyCatchBlock	
2	EmptyIfStmt	
3	EmptyWhileStmt	
4	EmptyTryBlock	
5	EmptyFinallyBlock	
6	UnnecessaryConversionTemporary	
7	EmptyStatementNotInLoop	
8	WhileLoopsMustUseBraces	
9	AssignmentInOperand	
10	UnnecessaryParentheses	
11	SimplifyBooleanExpressions	
12	SwitchStmtsShouldHaveDefault	
13	AvoidReassigningParameters	
14	FinalFieldCouldBeStatic	
15	EqualsNull	
16	SimpleDateFormatNeedsLocale	
17	ImmutableField	
18	AssignmentToNonFinalStatic	
19	AvoidSynchronizedAtMethodLevel	
20	AbstractClassWithoutAbstractMethod	
21	UncommentedEmptyMethod	
22	AvoidConstantsInterface	
23	DuplicateImports	
24	ImportFromSamePackage	
25	SystemPrintln	
26	VariableNamingConventions	
27	MisleadingVariableName	
28	AvoidArrayLoops	
29	UnnecessaryWrapperObjectCreation	
30	AvoidThrowingRawExceptionTypes	
31	AvoidThrowingNullPointerException	
32	StringInstantiation	
33	StringToString	

34	InefficientStringBuffering	
35	InefficientEmptyStringCheck	
36	UselessStringValueOf	
37	UnusedPrivateField	
38	UnusedPrivateMethod	
39	UnusedFormalParameter	

9.2 일반 코드 품질개선 예제

Rule-1	EmptyCatchBlock
설명	내용이 없는 Catch Block이 존재. Exception 처리가 안될 수 있음.
조치방법	catch시의 처리내용을 추가할것

잘못된 예제	public void doSomething() {
	try {
	FileInputStream fis = new FileInputStream("/tmp/bugger");
	} catch (IOException ioe) {
	}
	}

올바른 예제	public void doSomething() {
	try {
	FileInputStream fis = new FileInputStream("/tmp/bugger");
	} catch (IOException ioe) {
	doSomething();
	}
	}

Rule-2	EmptyFinallyBlock
설명	finally block이 비어있음
조치방법	빈 블럭 삭제 (finally 내부 로직 추가)

잘못된 예제	public class Foo {
	public void bar() {
	try {
	int x=2;
	} finally {
	// empty!
	}
	}
	}

올바른 예제	public class Foo {
	public void bar() {
	try {
	int x=2;
	} finally {
	int y = x+3;
	return y;
	}
	}
	}

Rule-3	EmptyIfStmt
설명	빈 if 구문의 사용을 피하도록 함
조치방법	삭제 또는 반대 조건을 이용하여 처리

잘못된 예제	public class Foo {
	void bar(int x) {
	if (x == 0) {
	// empty!
	}
	}
	}

올바른 예제	public class Foo {
	void bar(int x) {
	if (x == 0) {
	doSomething();
	}else if{
	doSomethingOther();
	}
	}
	}

Rule-4	EmptyStatementNotInLoop
설명	필요없는 문장 (";")이 있음
조치방법	삭제

잘못된 예제	public class MyClass {
	public void doit() {
	// the extra semicolon here this is not necessary
	System.out.println("look at the extra semicolon");;
	}
	}

올바른 예제	public class MyClass {

	public void doit() {
	// the extra semicolon here this is not necessary
	System.out.println("look at the extra semicolon");
	}
	}

Rule-5	EmptyTryBlock
설명	내용이 없는 try 블록이 존재함
조치방법	삭제

잘못된 예제	public class Foo {
	public void bar() {
	try {
	} catch (Exception e) {
	e.printStackTrace();
	}
	}
	}

올바른 예제	public class Foo {
	public void bar() {
	
	}
	}

Rule-6	EmptyWhileStmt
설명	빈 while 구문이 사용되었음
조치방법	삭제

잘못된 예제	public class Foo {
	void bar(int a, int b) {
	while (a == b) {
	//
	}
	}
	}

올바른 예제	public class Foo {
	void bar(int a, int b) {
	//소스 구현 없는 경우 메소스,클래스도 삭제
	}
	}

Rule-7	UnnecessaryConversionTemporary
설명	기본 데이터(primitive type)를 String으로 변환할 때 불필요한 임시 변환 작업을

	피하도록 함
조치방법	각 primitive type을 대표하는 클래스들(Integer,Character,Double...)의 static method를 이용하여 변환처리

잘못된 예제	public String convert(int x) {
	// this wastes an object
	String foo = new Integer(x).toString();
	}

올바른 예제	public String convert(int x) {
	return Integer.toString(x);
	}

Rule-8	WhileLoopsMustUseBraces
설명	중괄호({})없이 사용된 while문의 사용은 피하라
조치방법	중괄호 추가

잘못된 예제	public void doSomething() {
	while (true)
	x++;
	}

올바른 예제	public void doSomething() {
	while (true) {
	x++;
	}
	}

9.3 코드 가독성 개선 예제

Rule-9	AssignmentInOperand
설명	피연산자내에 할당문이 사용됨. Code 를 복잡하고 유지보수성이 떨어지게 만듬
조치방법	유지보수성을 위해 각각을 나누어 처리할것

잘못된 예제	public class Foo {
	public void bar() {
	int x = 2;
	if ((x = getX()) == 3) {
	doSomething();
	}
	}
	private int getX() {

	return 3; } }

올바른 예제	public class Foo {
	public void bar() {
	int x = 2;
	x = getX();
	if (x == 3) {
	doSomething();
	}
	}
	private int getX() {
	return 3;
	}
	}

Rule-10	UnnecessaryParentheses
설명	괄호가 없어도 되는 상황에서 불필요한 괄호를 사용할 경우 마치 메소드 호출처럼 보여서 소스 코드의 유지보수성을 떨어뜨릴 수 있음. 이에 불필요한 괄호를 삭제토록 함
조치방법	괄호삭제

잘못된 예제	public class Foo {
	boolean bar() {
	return (true);
	}
	}

올바른 예제	public class Foo {
	boolean bar() {
	return true;
	}
	}

9.4 코드 설계 변경으로 품질개선 예제

Rule-11	SimplifyBooleanExpressions
설명	boolean 사용 시 불필요한 비교 연산을 피하도록 함. 단순한 code 를 복잡하게 만듬.
조치방법	불필요한 비교연산 대신 단순연산처리

잘못된 예제	public class Bar {
	private boolean bar = (isFoo() == true);
	public isFoo() { return false;}
	}

올바른 예제	public class Bar {
	bar = isFoo();
	public isFoo() { return false;}
	}

Rule-12	SwitchStmtsShouldHaveDefault
설명	Switch구문에는 반드시 default label이 있어야 함
조치방법	default label 추가

잘못된 예제	public class Foo {
	public void bar() {
	int x = 2;
	switch (x) {
	case 2: int j = 8;
	}
	}
	}

올바른 예제	public class Foo {
	public void bar() {
	int x = 2;
	switch (x) {
	case 2: int j = 8;
	default: doSomething();
	}
	}
	}

Rule-13	AvoidReassigningParameters
설명	할당을 통해 파라미터 값을 직접 변경금지
조치방법	필요하다면 임시지역변수를 사용함.

잘못된 예제	private void foo(String bar) {
	bar = "something else";
	}

올바른 예제	private void bar(String bar) {
	String tmp = "something else";
	}

Rule-14	FinalFieldCouldBeStatic

설명	final field를 Static 으로 변경하면 overhead 를 줄일 수 있음.
조치방법	static 추가

잘못된 예제	public class Foo {
	// this could be static and save some space
	public final int BAR = 42;
	}

올바른 예제	public class Foo {
	// this could be static and save some space
	public final static int BAR = 42;
	}

Rule-15	EqualsNull
설명	null 값과 비교하기 위해 equals 함수를 사용하였음.
조치방법	null 비교는 (x == null) 형태로 변경

잘못된 예제	class Bar {
	void foo() {
	String x = "foo";
	if (x.equals(null)) {
	doSomething();
	}
	}
	}

올바른 예제	class Bar {
	void foo() {
	String x = "foo";
	if (x == null) {
	doSomething();
	}
	}
	}

Rule-16	SimpleDateFormatNeedsLocale
설명	SimpleDateFormat 인스턴스를 생성할때 Locale 을 지정하는 것이 바람직함
조치방법	Locale 명시

잘못된 예제	public class Foo {
	private SimpleDateFormat sdf = new SimpleDateFormat("E MMM dd HH:mmss");
	}

올바른 예제	public class Foo {
	private SimpleDateFormat sdf =
	new SimpleDateFormat("E MMM dd HH:mmss",Locale.KOREA);

	}
Rule-17	ImmutableField
설명	생성자에서 Assign된 변수를 Final로 선언하지 않았음
조치방법	생성자에서만 할당되는 변수라면 final로 선언

잘못된 예제	public class Foo {
	private int x;
	public Foo() {
	x = 7;
	}
	public void foo() {
	int a = x + 2;
	}
	}

올바른 예제	public class Foo {
	final int x;
	public Foo() {
	x = 7;
	}
	public void foo() {
	int a = x + 2;
	}
	}

Rule-18	AssignmentToNonFinalStatic
설명	static 필드의 안전하지않은 사용 가능성
조치방법	final을 선언하여 막을것

잘못된 예제	public class StaticField {
	static int x;
	public FinalFields(int y) {
	x = y; // unsafe
	}
	}

올바른 예제	public class StaticField {
	static final int x;
	public FinalFields(int y) {
	x = y; // unsafe
	}
	}

Rule-19	AvoidSynchronizedAtMethodLevel

설명	mothod 레벨의 synchronization 보다 block 레벨 synchronization 을 사용하는 것이 바람직함 새로운 코드가 메소드에 추가되었을때, 메소드 레벨 동기화는 역효과를 일으킨다. 블락레벨 동기화는 동기화가 필요한 코드가 동기화될 수 있도록 도울것이다.
조치방법	블럭레벨 sync로 변경한다.

잘못된 예제	public class Foo {
	synchronized void foo() {
	}
	}

올바른 예제	public class Foo {
	void bar() {
	synchronized(this) {
	}
	}
	}

Rule-20	AbstractClassWithoutAbstractMethod
설명	Abstract Class내에 Abstract Method가 존재하지 않음
조치방법	비 추상 메소드를 제거하거나 abstarct임을 명시할것

잘못된 예제	public abstract class Foo {
	void int method1() { ... }
	void int method2() { ... }
	}

올바른 예제	public abstract class Foo {
	abstract void int method1() { ... }
	abstract void int method2() { ... }
	// consider using abstract methods or removing
	// the abstract modifier and adding protected constructors
	}

Rule-21	UncommentedEmptyMethod
설명	Empty Method 가 사용되었음. 의도적으로 Empty Method 를 구현했다면 그에 대항하는 주석을 기재하는 것이 바람직함.
조치방법	불필요할 경우 삭제, 의도적이면 주석추가
잘못된 예제	public void doSomething() { }
올바른 예제	삭제
Rule-22	AvoidConstantsInterface
설명	Interface는 클래스의 behavior 을 구현하는 데에만 사용해야 함. 상수를 담는 컨테이너로 사용하는 것은 바람직하지 않음

	지금 상황에서는 클래스를 사용하는 것이 바람직함
조치방법	interface 내에 상수를 선언하지 말것. (상수클래스를 사용)

잘못된 예제	public interface ConstantsInterface {
	    public static final int CONSTANT1=0;
	    public static final String CONSTANT2="1";
	}

올바른 예제	public class ConstantsClass {
	    public static final int CONSTANT1=0;
	    public static final String CONSTANT2="1";
	}

9.5 기타 코드 표준 개선 예제

Rule-23	DuplicateImports
설명	import문이 중복 선언 되었음
조치방법	중복 import 삭제
잘못된 예제	import java.lang.String; import java.lang.*; public class Foo {}
올바른 예제	import java.lang.*; public class Foo {}
Rule-24	ImportFromSamePackage
설명	동일 패키지에 있을 때는 import문을 사용할 필요가 없음
조치방법	import 삭제
잘못된 예제	package foo; import foo.Buz; import foo.*; public class Bar{}
올바른 예제	package foo; public class Bar{}
Rule-25	SystemPrintln
설명	개발 시 디버그 작업을 위해 사용한 System.out.println()문이 남아 있음. 불필요한 System.out.println()문은 시스템에 부하를 발생 시킨다. (System.out 으로 시작하는 부분을 모두 찾도록 기본PMD룰을 보완함)
조치방법	삭제 또는Logger 사용을 고려하기 바람

잘못된 예제	class Foo{
	    Logger log = Logger.getLogger(Foo.class.getName());
	    public void testA () {

	System.out.println("Entering test"); } }

올바른 예제	class Foo{
	public void testA () {
	// Better use this
	debug("Entering test");
	}
	}

Rule-26	VariableNamingConventions
설명	다음 2가지 경우 중 하나임. - final 변수 이름이 대문자가 아님. - non-final 변수에 underscore 가 포함되어 있음.
조치방법	명명규칙준수 - final 대문자 - 일반변수는 _ 제거

잘못된 예제	public class Foo {
	public static final int my_num = 0;
	public String my_Test = "";
	DataModule dm_Test = new DataModule();
	}

올바른 예제	public class Foo {
	public static final int MY_NUM = 0;
	public String myTest = "";
	DataModule dmTest = new DataModule();
	}

Rule-27	MisleadingVariableName
설명	non-field 이름이 m_ 으로 시작함. M_ 은 field 를 지칭하는 것이므로 혼돈을 줄 수 있음
조치방법	Rule준수 m_ 삭제

잘못된 예제	public class Foo {
	public void bar(String m_baz) { // Bad
	int m_boz = 42; // Bad
	}
	}

올바른 예제	public class Foo {
	public void bar(String baz) { // Bad
	int boz = 42; // Bad

<table>
<tr><td></td><td><pre> }
}</pre></td></tr>
<tr><td>Rule-28</td><td>AvoidArrayLoops</td></tr>
<tr><td>설명</td><td>배열의 값을 루프문을 이용하여 복사하는 것 보다 System.arraycopy() 메소드를 이용하여 복사하는 것이 효율적이며 수행 속도가 빠름</td></tr>
<tr><td>조치방법</td><td>루프를 이용한 배열값 복사 대신 System.arraycopy 메소드를 사용할것</td></tr>
<tr><td>잘못된 예제</td><td><pre>public class Test {
 public void bar() {
  int[] a = new int[10];
  int[] b = new int[10];
  for (int i=0;i<10;i++) {
   b[i]=a[i];
  }
 }
}</pre></td></tr>
<tr><td>올바른 예제</td><td><pre>public class Test {
  public void bar() {
   int[] a = new int[10];
   int[] b = new int[10];
   System.arraycopy(a, 0, b, 0, 10);
  }
 }</pre></td></tr>
<tr><td>Rule-29</td><td>UnnecessaryWrapperObjectCreation</td></tr>
<tr><td>설명</td><td>불필요한 Wrapper Object가 생성되었음.
Parsing 메소드를 직접적 호출하는 것이 바람직함.</td></tr>
<tr><td>조치방법</td><td>parseXXX 형태의 메소드로 변경</td></tr>
<tr><td>잘못된 예제</td><td><pre>public int convert(String s) {
  int i, i2;
  i = Integer.valueOf(s).intValue(); // this wastes an object
  i2 = Integer.valueOf(i).intValue(); // this wastes an object
  return i2;
}</pre></td></tr>
<tr><td>올바른 예제</td><td><pre>public int convert(String s) {
  int i, i2;
  i = Integer.parseInt(s); // this is better
  i2 = i; // this is better
  return i2;
}</pre></td></tr>
</table>

Rule-30	AvoidThrowingRawExceptionTypes
설명	가공되지 않은 Exception을 throw하는 것은 비추천
조치방법	사용자 정의 Exception을 사용하도록 함

잘못된 예제	public class Foo {
	public void bar() throws Exception {
	throw new Exception();
	}
	}

올바른 예제	public class Foo {
	public void bar() throws BizException {
	throw new BizException();
	}
	}

Rule-31	AvoidThrowingNullPointerException
설명	NullPointerException을 throw하는 것은 비추천
조치방법	대신 IllegalArgumentException을 사용할것

잘못된 예제	public class Foo {
	void bar() {
	throw new NullPointerException();
	}
	}

올바른 예제	public class Foo {
	void bar() {
	throw new IllegalArgumentException();
	}
	}

Rule-32	StringInstantiation
설명	필요없는 Instance가 생성되어 있음
조치방법	new로 생성하지 말고, 직접 문자열에 대입 ex) String bar = "bar";

잘못된 예제	public class Foo {
	private String bar = new String("bar");
	}

올바른 예제	public class Foo {
	String bar = "bar";
	}

Rule-33	StringToString
설명	String 객체에서 toString()함수를 사용하는 것은 불필요함
조치방법	toString 삭제

잘못된 예제	public class Foo {
	private String baz() {
	String bar = "howdy";
	return bar.toString();
	}
	}

올바른 예제	public class Foo {
	private String baz() {
	String bar = "howdy";
	return bar;
	}
	}

Rule-34	InefficientStringBuffering
설명	StringBuffer 함수에서 nonliteral 을 직접 concatenate 하지 말 것. Nonliteral conatenation 는 별도로 처리할 것.
조치방법	append를 사용하여 처리할것

잘못된 예제	public class Foo {
	void bar() {
	StringBuffer sb=new StringBuffer("tmp = "+System.getProperty("java.io.tmpdir"));
	}
	}

올바른 예제	public class Foo {
	void bar() {
	StringBuffer sb = new StringBuffer("tmp = ");
	sb.append(System.getProperty("java.io.tmpdir"));
	}
	}

Rule-35	InefficientEmptyStringCheck
설명	Empty String 을 체크하기 위해 String.trim().length() 을 사용하는 것은 피하도록 함. Character.isWhitespace()를 사용하는 것이 바람직함
조치방법	isWhitespace를 이용하여 공백을 체크하는 로직으로 변경처리

잘못된 예제	public class Foo {
	void bar(String string) {
	if (string != null && string.trim().size() > 0) {
	doSomething();
	}
	}
	}

올바른 예제	import com.sli.framework.util.StringUtil;
	public class Foo {
	  void bar(String string) {
	  if (!StringUtil.isEmpty(string)) {
	  doSomething();
	  }
	  }
	}

Rule-36	UselessStringValueOf
설명	String 을 append 할 경우, String.valueOf 함수를 사용할 필요 없음. Valueof() 대신 직접 사용하는 것이 바람직함.
조치방법	valueOf 제거

잘못된 예제	public String convert(int i) {
	  String s;
	  s = "a" + String.valueOf(i);
	  return s;
	}

올바른 예제	public String convert(int i) {
	  String s;
	  s = "a" + i;
	  return s;
	}

Rule-37	UnusedPrivateField
설명	사용되지 않는 Private field가 선언되었음
조치방법	field 삭제

잘못된 예제	public class Something {
	  private static int FOO = 2; // Unused
	  private int i = 5; // Unused
	  private int j = 6;
	  public int addOne() {
	  return j++;
	  }
	}

올바른 예제	public class Something {
	  private int j = 6;
	  public int addOne() {
	  return j++;
	  }
	}

Rule-38	UnusedPrivateMethod
설명	사용되지 않는 Private Method가 선언되었음
조치방법	method 삭제

잘못된 예제	public class Something {
	private void foo() {} // unused
	}

올바른 예제	public class Something { }
Rule-39	UnusedFormalParameter
설명	사용되지 않는 파라미터가 있음
조치방법	파라미터 삭제

잘못된 예제	public class Foo {
	private void bar(String howdy) {
	// howdy is not used
	}

올바른 예제	public class Foo {
	private void bar() {
	// howdy is not used
	}