Spring boot 3 기반 uiadapter

개발자 가이드

TOBESOFT

저작권 및 면책조항

이 문서에 잘못된 정보가 있을 수 있습니다. 투비소프트는 이 문서가 제공하는 정보의 정확성을 유지하기 위해 노력하고 특별한 언급 없이 이 문서를 지속적으로 변경하고 보완할 것입니다. 그러나 이 문서에 잘못된 정보가 포함되어 있지 않다는 것을 보증하지 않습니다. 이 문서에 기술된 정보로 인해 발생할 수 있는 직접적인 또는 간접적인 손해, 데이터, 프로그램, 기타 무형의 재산에 관한 손실, 사용 이익의 손실 등에 대해 비록 이와 같은 손해 가능성에 대해 사전에 알고 있었다고 해도 손해 배상 등 기타 책임을 지지 않습니다.

사용자는 본 문서를 구입하거나, 전자 문서로 내려 받거나, 사용을 시작함으로써, 여기에 명시된 내용을 이해하며, 이에 동의하는 것으로 간주합니다.

발행처 | (주)투비소프트

발행일 | 2024/01/18

주소 | (06083) 서울시 강남구 봉은사로 617 인탑스빌딩 2-5 층

전화 | 02-2140-7700

홈페이지 | www.tobesoft.com

고객지원센터 | support.tobesoft.co.kr

제품기술문의 | 1588-7895 (오전 10 시부터 오후 4 시까지)

문서 개정이력표			
문서명		Spring boot 3 기반 uiadapter 개발자 가이드	
버전	날짜	내용	작성자
0.1	2024.01.18	uiadapter 개발자 가이드 최초 작성	모인환
0.2	2024.08.12	uiadpater 패키지명 및 모듈명 수정	모인환
0.3	2026.01.05	샘플 프로젝트 저장소를 gitlab으로 변경	모인환

목차

1. 개요

1.1 목적

본 문서는 uiadapter 에 관한 가이드문서로서 nexacro UI 와 연계하는 웹어플리케이션 서버(WAS) 시스템 구축 시 개발자가 실제로 작업을 해야 되는 내용에 대해 서술한다. spring boot 3 환경에서 uiadapter 의 구조적 특징과 각 Layer 별 class 의 코딩방식을 설명하여 uiadapter 를 기반으로 개발 및 운영 시에 생산성과 효율성을 높이는 것을 목적으로 한다.

1.2 범위

WAS 의 spring boot 3 기반에서 uiadapter 를 개발하는 process 전반에 적용하는 것을 범위로 한다. 개발자 및 운영자가 일관성이 있는 개발을 하도록 코딩표준 가이드를 포함한다. 선수지식으로 nexacro 에 대한 교육을 받거나 경험한 것으로 가정하고 자세한 nexacro 에 대한 설명은 본 문서의 범주에서 제외한다. 형상관리 및 배포, DB 또한 범주에서 제외한다.

1.3 레퍼런스

본 문서에서 제공하는 프로젝트 소스는 gitlab 저장소(https://gitlab.com/nexacro/spring-boot/jakarta/uiadapter-jakarta.git)를 통해 하고 있다. 소프트웨어의 추가나 업그레이드가 있을 경우, 본 문서를 수정하여 추가 배포, 공지하도록 한다.

2. 환경구성

2.1 시스템 기반 기술

Uiadapter 모듈을 사용하는 샘플 프로젝트는 spring boot 3.5.9 를 기반으로 개발되었다. 개발관 관련하여 요구되는 기술 스택과 적용 버전은 다음과 같다

2.1.1 개발 소프트웨어

항목	소프트웨어	비고
개발 언어	jdk	17 이상
개발툴	sts	5.0.1.RELEASE
db	hsqldb	외 oracle, msSql, mySql, mariaDB, Postgresql 지원
개발자 OS	windows	10 이상

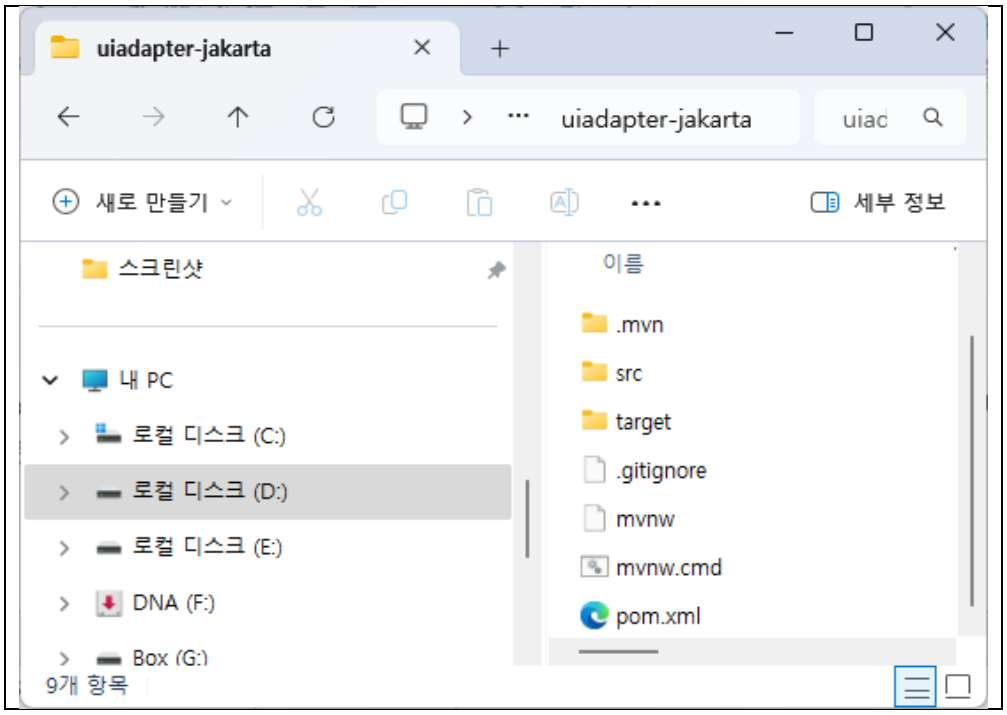
2.1.2 개발 표준 스택

개발 스택	지원 사양	비고
tomcat	9.0.54	Embadding Tomcat in STS
servlet	6.0.0	jakarta.servlet-api
spring framework	6.1.x	
spring boot	3.5.9	
myBatis	2.3.1	mybatis-spring-boot-starter
Lombok	1.18.42	
uiadapter	1.0.24-SNAPSHOT	uiadapter-jakarta-core
x-api	2.0.01	
xeni	1.5.02	poi 5.2.x 지원 버전

2.2 개발환경

2.2.1 개발환경 디렉토리 구조

본 가이드 문서의 경우, uiadapter 모듈을 사용하는 셈플 프로젝트는 아래와 같은 디렉토리 구조를 기반으로 설명한다. 개발 환경의 루트 경로를 D:₩example ₩uiadapter-jakarta 로 한다.



- **jdk :** 개발언어

  Spring Boot 3.0 이상부터 JDK 17 이상 버전을 필요로 한다. Uiadapter 의 개발환경에서는 Oracle 의 OpenJDK 17 을 사용한다

  오라클 JDK 17 이외의 버전을 사용할 경우, 오라클의 공식 웹사이트(https://www.oracle.com/java/technologies/downloads/) 혹은 OpenJDK github(https://github.com/ojdkbuild/ojdkbuild)에서 JDK 를 다운로드 한다.

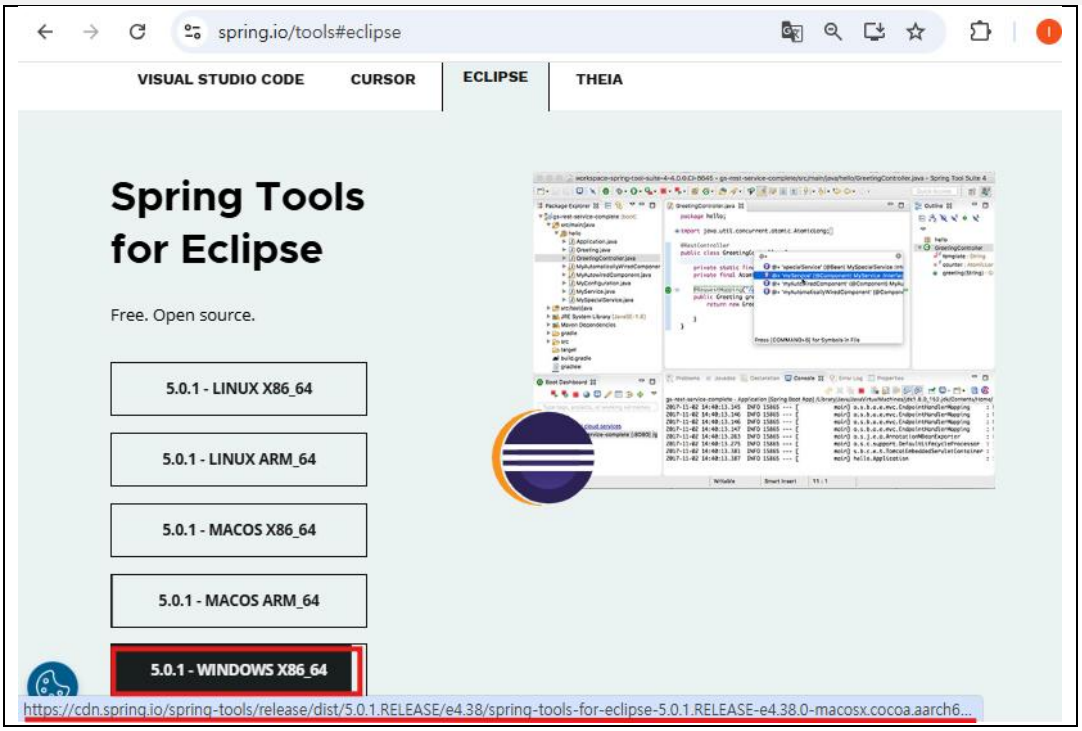
Eclipse Temurin OpenJDK 의 경우 웹사이트 (https://adoptium.net/temurin/releases/?version=17) 에서 다운로드 할 수 있다.

- **IDE : 개발도구 STS5(Spring Tool Suite 5) 설치**

개발툴 중 STS 5.0.1.RELEASE 버전을 기준으로 문서를 작성한다. 만약, spring boot 프로젝트가 아닌 스프링 MVC 프로젝트(Spring MVC Project)나 메이븐(Maven) 등 기본 설정이 미리 준비된 스프링 프로젝트를 사용하려면 Eclipse Marketplace 에서 STS 4 Release 와 Add-on 모듈을 추가로 설치해야 한다.

◆ STS5 다운로드 URL

https://spring.io/tools/



Spring Tools 5.0.1 Windows 버전을 다운로드 한다.

다운로드한 zip 파일을 임의의 경로에 푼다. 문서의 경우 개발툴의 설치 경로와 실행파일은 D:₩eclipse₩sts-5.0.1.RELEASE 경로에 있는 SpringToolsForEclipse.exe 이다.

- repository : **저장소**

  개발가이드에서 소개하는 샘플프로젝트는 Maven 프로젝트이다. 이 프로젝트를 repository 의 default 경로는 ${user.home}/.m2/repository/ 이지만 문서에서는 D:₩repository 경로를 사용한다. 설정은 D:₩repository₩settings.xml 파일에 경로를 수정하고 sts 툴 설정에서 settings.xml 파일을 지정한다.

  ◆ settings.xml 설정

  ```
  <localRepository>D:₩repository</localRepository>
  ```

- server : **tomcat 서버**

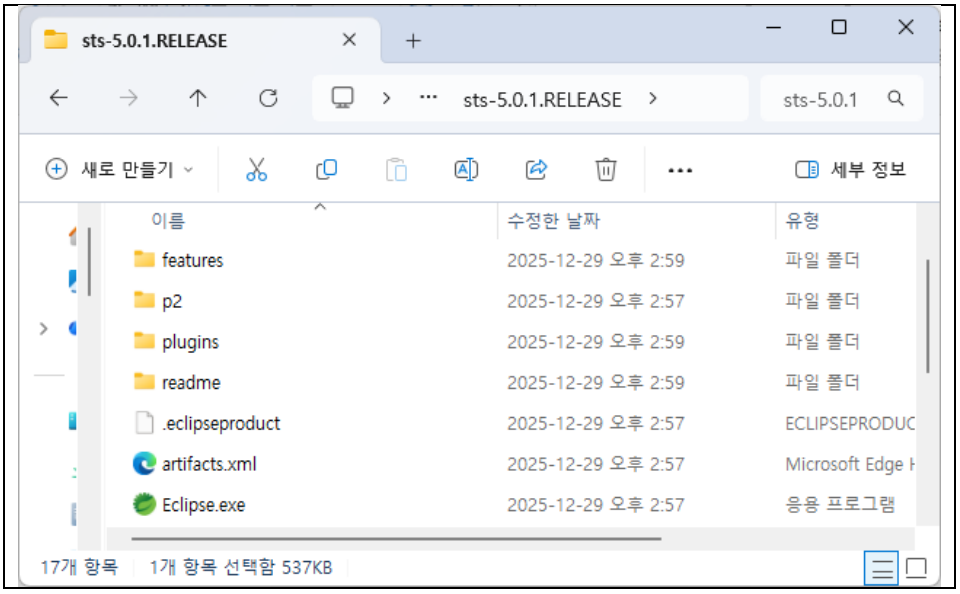
  tomcat 은 STS 에 내장되어 있고 버전은 11.0.x 이다.

- workspaces : **개발 작업공간**

  uiadapter 개발소스가 위치하는 작업공간이다. 샘플 경로는 D:₩example₩uiadapter-jakarta 이다.

2.2.2 개발환경 세팅

- SpringToolSuite5.ini 설정

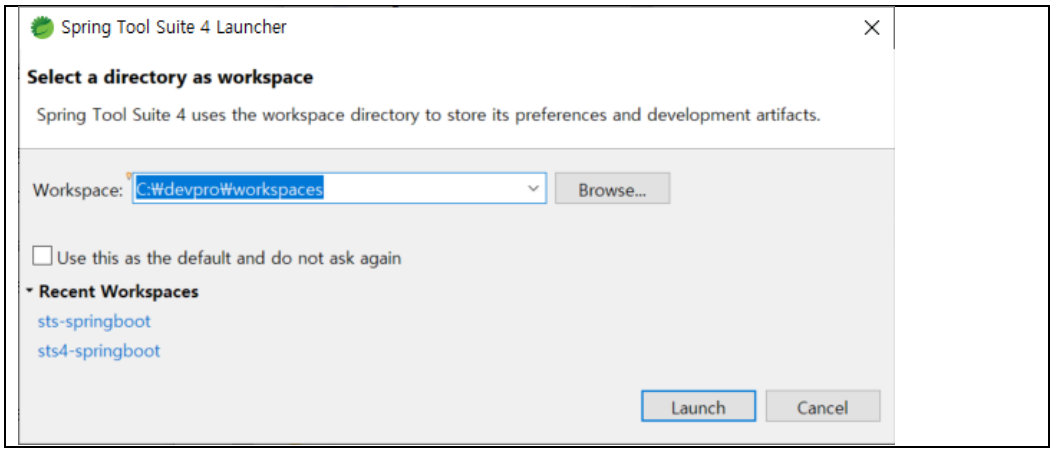


◆ 개발툴 실행파일(SpringToolsForEclipse.exe) 과 같은 경로에 있는 SpringToolsForEclipse.ini 파일을 수정한다. –vmargs 아래 다음과 같이 속성(encoding 및 heap 메모리)을 추가하거나 값을 조정한다.

```
-vmargs
 -Dosgi.requiredJavaVersion=21
 -Dosgi.dataAreaRequiresExplicitInit=true
 -Dsun.java.command=SpringToolsForEclipse
-Dfile.encoding=UTF-8
-Xms1024m
-Xmx2048m
-javaagent:D:₩eclipse₩sts-5.0.1.RELEASE₩lombok.jar
```

- sts workspace 설정

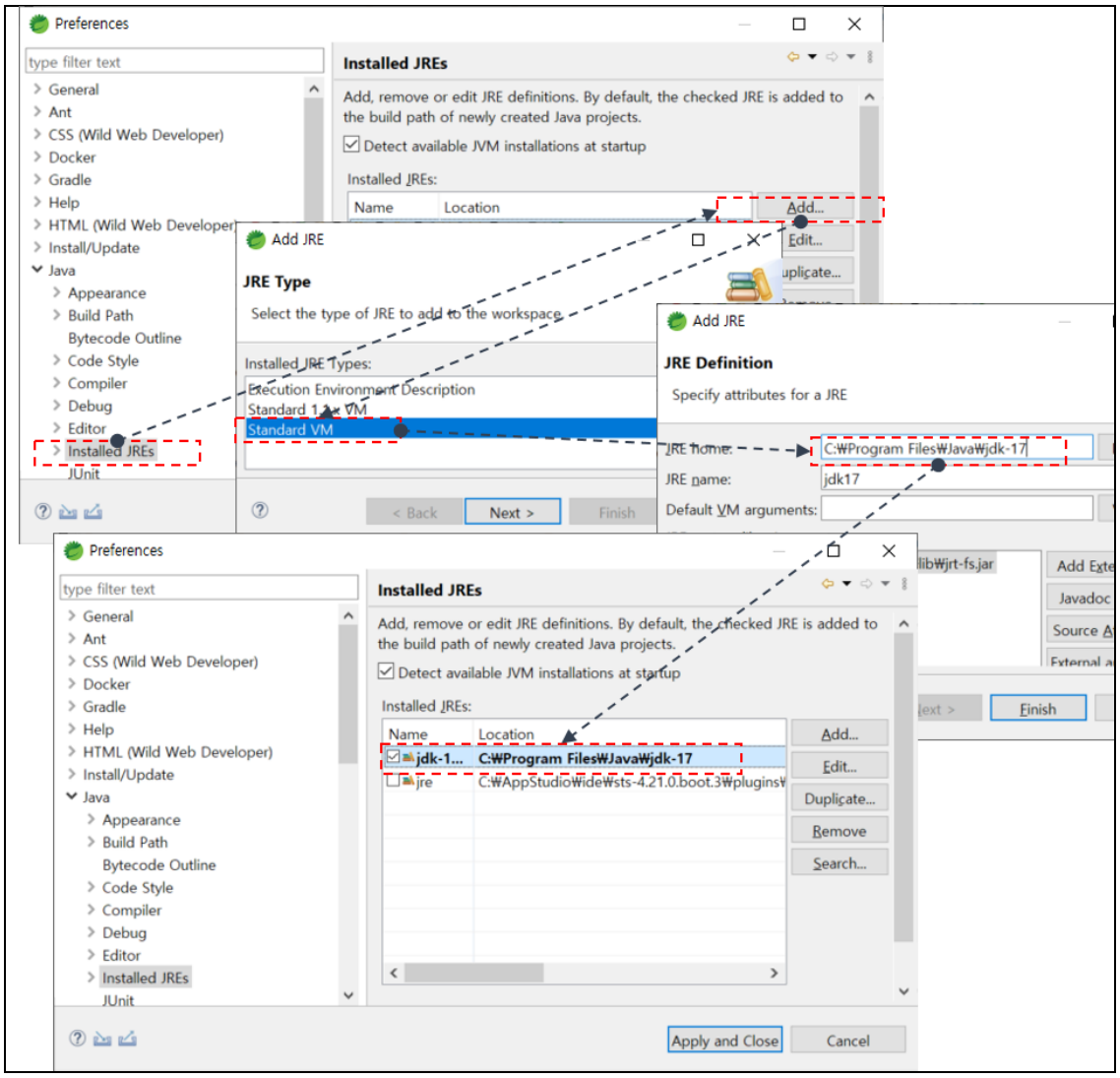
sts 의 workspace 는 D:₩example 로 지정한다.



- JRE 설정

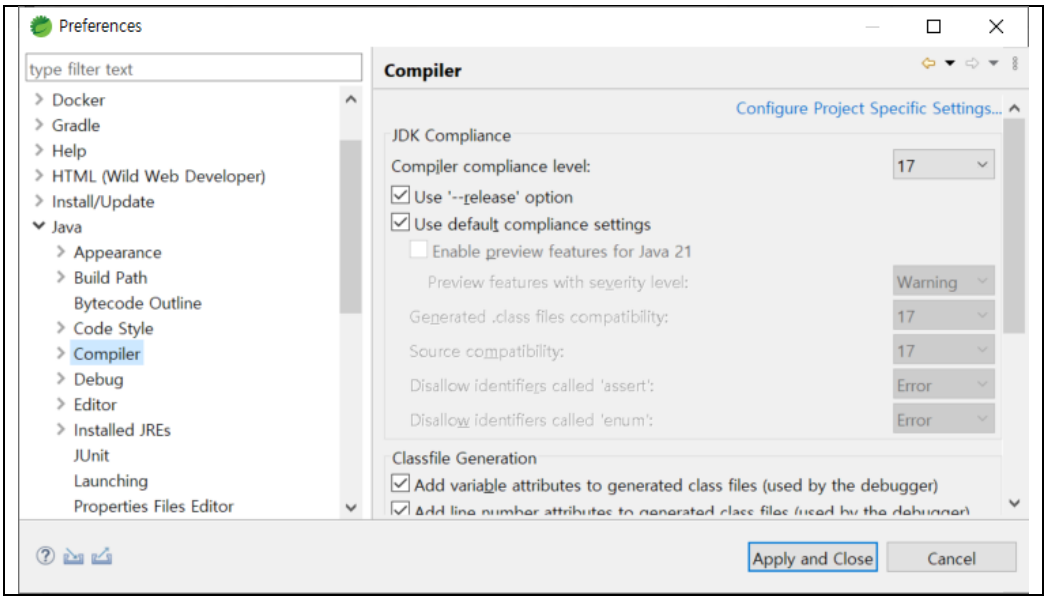
개발툴을 설치하고 jdk 버전이 17 이상인지 확인한다. 사용자 환경이 jdk 17 이하로 지정되 있다면 아래의 예시에 따라 jdk 17 을 지정하도록 한다.

메뉴 > windows > Preferences 선택 후, Java > Installed JRE 를 선택한다. JRE 를 추가하기 위해 [Add..] 클릭하고 "Standard VM" 항목 선택하고 [Next] 클릭한다. JRE home 항목을 [Directory..] 클릭하여 "C:₩Program Files₩Java₩jdk-17" 혹은 다운로드 받아 인스톨한 ".₩jdk₩jdk-17" 경로를 지정하고 마치면 jdk 17 버전이 리스트에 나타난다. 추가한 jdk-17 버전을 체크하여 활성화한다.



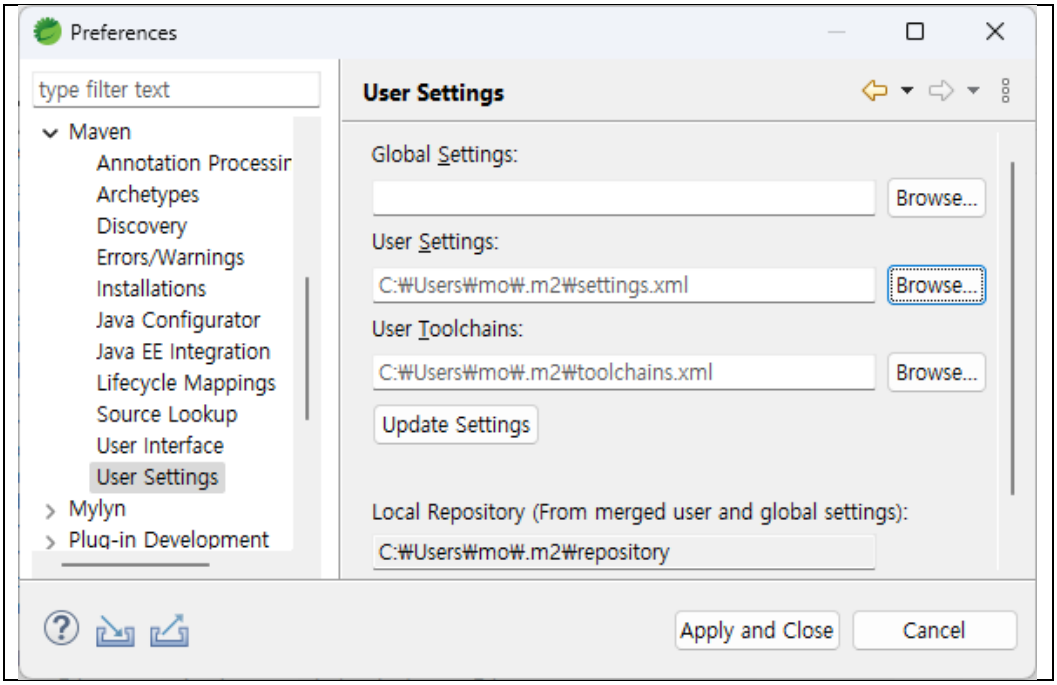
- Compiler 설정

메뉴 > windows > Preferences 선택 후, Java > Compiler 를 선택한다. Compiler level 을 17 버전으로 세팅한다.



- maven local repository 설정

메뉴 > windows > Preferences 선택 후, Maven > User Settings 를 선택한다. "User Settings"의 값을 default 경로 혹은 임의의 설정한 경로로 지정한다.



3. uiadapter 이해

스프링 프레임워크 기반 uiadapter 는 nexacro ui 화면과 스프링 프레임워크 기반 백앤드 시스템을 매끄럽게 이어주기 위한 목적을 가진 미들웨어 모듈이다. 주요 기능은 nexacro ui 에서 보낸 request 를 받아서 고유한 nexacro data 포맷을 bean 혹은 map 으로 변환하고 business layer 의 처리 결과를 다시 nexacro data 포맷으로 변환하여 response 하는 기능을 제공한다.

3.1 스프링프레임워크에서 uiadapter

3.1.1 uiadapter 구성 모듈

- uiadapter 는 3 개의 모듈로 구성된다.

모듈명	설명
uiadapter-spring-core	Nexacro ui 요청 request/response 처리, xapi 기반 데이터 포맷팅 및 어노테이션, 예외 처리
uiadapter-spring-dataaccess	mybatis기반의 Database 별 데이터 포맷팅, 가공 처리
uiadapter-spring-excel	xeni 기반 Excel import, export 처리 기능 지원

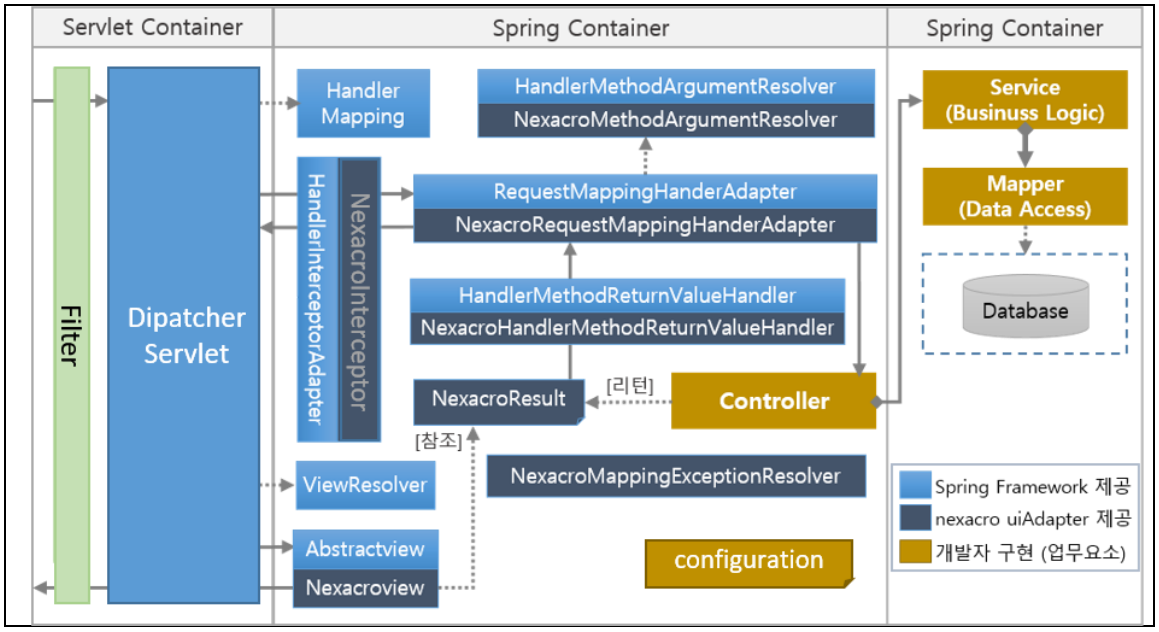
- uiadapter 의 배포 모듈은 nexus 저장소를 통하여 제공한다.
- nexus maven repository 주소는 다음과 같다.
  - https://mangosteen.tobesoft.co.kr/nexus/repository/maven-public/
- uiadapter Maven dependency

모듈명	Maven dependency
uiadapter-spring-core	<dependency> <groupId>com.nexacro</groupId> <artifactId>uiadapter-jakarta-core</artifactId> <version>1.0.27-SNAPSHOT</version> </dependency>
uiadapter-spring-dataaccess	<dependency>

	<groupId>com.nexacro</groupId> <artifactId>uiadapter-jakarta-dataaccess</artifactId> <version>1.0.14-SNAPSHOT</version> </dependency>
uiadapter-spring-excel	<dependency> <groupId>com.nexacro</groupId> <artifactId>uiadapter-jakarta-excel</artifactId> <version>1.5.4.3-SNAPSHOT</version> </dependency>

* 모듈별 버전은 패치가 발생할 경우 올라갑니다.

3.1.2 uiadapter 아키텍처 다이어그램



3.1.3 uiadapter 주요 요소

- NexacroInterceptor

nexacro 으로부터 데이터를 수신 받아 PlatformData 로 변환하는 작업을 수행하는 interceptor 이다. 이 클래스에서 수행하는 기능은 NexacroMethodArgumentResolver 에서도 수행된다.

- NexacroRequestMappingHandlerAdapter

nexacro 화면에서 서버로 보낸 데이터는 고유한 포맷을 가지고 있다. spring framework 가 nexacro 화면에서 보낸 고유한 포맷을 알맞게 변환할 수 있도록 보장하기 위해 NexacroMethodArgumentResolver 가 가장 먼저 처리할 수 있도록 0 순위로 세팅한다.

- NexacroMethodArgumentResolver

Nexacro ui 로부터 전송된 데이터를 자바 객체로 변환(Resolving)해주는 핵심 역할을 한다. 그러면 nexacro 화면에서 보낸 Dataset 및 Variable 데이터를 개발자가 작성하는 Controller 메서드의 아규먼트에 적절한 어노테이션을 사용하여 Map 이나 Bean 객체로 꺼내어 사용할 수 있다.

- 지원 파라미터 목록

클래스 / 어노테이션	설명
@ParamDataSet	특정 명칭의 DataSet을 List<Map>, List<Bean>, DataSet 등으로 바인딩
@ParamVariable	특정 명칭의 Variable을 자바 기본 자료형으로 바인딩
PlatformData	넥사크로 통신의 기본 단위 (DataSetList, VariableList 포함)
DataSetList	전체 DataSet 목록
VariableList	전체 Variable 목록
NexacroFirstRowHandler	대용량 데이터 분할 전송(FirstRow) 처리를 위한 핸들러

ex) DataSet -> java Vo object (Board searchVo)

```
@RequestMapping (value = "select_datalist.do")
public NexacroResult select_datalist(@ParamDataSet(name = "dsSearch") Board searchVo)
```

ex) DataSet -> Map (List<Map<String, Object>> dataList)

```
@RequestMapping (value = "select_datalist_map.do")
public NexacroResult update_datalist(@ParamDataSet(name = "input1") List<Map<String,
Object>> dataList)
```

- NexacroHandlerMethodReturnValueHandler

컨트롤러가 리턴한 자바 객체(List, vo, Map 등)를 nexacro 가 해석할 수 있는 DataSet 이나 Variable 형태로 변환하여 응답 객체에 반환한다.

- 지원하는 Return Type

Return Type	설명
NexacroResult	여러 개의 DataSet과 Variable, 에러 정보를 한 번에 담아 보낼 때 사용되는 전용 객체
PlatformData	넥사크로 XAPI의 기본 데이터 객체로, 이미 구성된 데이터를 그대로 송신할 때 사용
NexacroFileResult	파일 다운로드 처리를 위한 정보를 담고 있는 객체

ex) java bean, map -> DataSet

- NexacroView, NexacroFileView

nexacro 으로 데이터 및 파일을 전달한다. 데이터의 경우 xml, json, svn 포맷을 사용할 수 있다.

- NexacroMappingExceptionResolver

nexacro 의 통신 규약에 맞는 예외 처리를 담당한다.

3.1.4 uiadapter 기반 샘플 프로젝트 main()

샘플 프로젝트는 Spring boot 를 기반으로 한다. main() 함수가 있는 프로젝트의 진입점은 example.nexacro.uiadapter.UiadapterApplication.java 이다. main() 함수는 uiadapterApplication.run()을 호출하여 웹 애플리케이션을 실행하는 역할을 한다.

클래스 레벨에 선언된 @SpringBootAplication 어노테이션은 spring framework 의 dispatcher-servlet.xml 등 xml 에서 지정했던 루틴 한 설정을 자동으로 실행한다. @SpringBootAplication 어노테이션은 3 가지의 어노테이션을 포함한다.

어노테이션	기능
@EnableAutoConfiguration	스프링 부트 개발에 필수적인 설정들을 자동으로 처리

@ComponentScan	스프링 레거시에서는 XML 설정으로 등록했던 ComponentScan 기능을 자동으로 처리
@Configuration	@Configuration 어노테이션이 선언된 클래스는 Java 기반의 설정 파일로 인식하여 처리

3.2 nexacro data format 변환

인터넷 구간에서 nexacro 와의 data 통신은 기본적으로 xml 형태를 가지고 설정에 따라 ssv, json 등의 포맷을 지원한다. nexacro x-api 모듈은 nexacro data 포맷을 java 객체로 서로 전환하는 기능을 제공한다.

3.2.1 nexacro data format

- Xml 포맷 예시

```
<?xml version="1.0" encoding="utf-8"?>
<Root xmlns="http://www.nexacroplatform.com/platform/dataset" ver="4000" >
    <Parameters>
        <Parameter id="ErrorCode">0</Parameter>
        <Parameter id="ErrorMsg">SUCCESS</Parameter>
    </Parameters>
    <Dataset id="output">
        <ColumnInfo>
            <ConstColumn id="market" size="10" type="STRING" value="kr"/>
            <Column id="stockCode" size="5" type="STRING"/>
            <Column id="currentPrice" size="10" type="INT"/>
        </ColumnInfo>
        <Rows>
            <Row>
                <Col id="currentCode">10001</Col>
                <Col id="currentprice">5700</Col>
            </Row>
            <Row>
```

```
                    <Col id="currentCode">10002</Col>
                    <Col id="currentprice">14500</Col>
                </Row>
            </Rows>
        </Dataset>
</Root>
```

※ nexacro xml 데이터 포맷 참조 URL: http://docs.tobesoft.com/advanced_development_guide_nexacro_n_ko/c0f133d58a62e0fd

- ◆ Parameter: id 와 value 쌍으로 값을 전달한다.
- ◆ ErrorCode 와 ErrorMsg 는 예약어이다.
- ◆ Dataset: 데이터의 구조와 값을 함께 가지고 있다. 컬럼의 구조는 Column 에 담고 데이터는 Row 에 담아서 전달한다. Column 의 id 는 Map 의 key 에 해당한다.

- Save Dataset 의 row type 별 포맷

```
<Rows>
  <Row type="insert">
    <Col id="currentCode">10001</Col>
    <Col id="currentprice">13400</Col>
  </Row>
  <Row type="update">
    <Col id="currentCode">10001</Col>
    <Col id="currentprice">13400</Col>
    <OrgRow>
      <Col id="currentCode">10001</Col>
      <Col id="currentprice">13700</Col>
    </OrgRow>
  </Row>
  <Row type="delete">
    <Col id="currentCode">10001</Col>
    <Col id="currentprice">13400</Col>
  </Row>
```

```
</Rows>
```

◆ Row type 별 속성 값

Row type	설명
Insert	원본 Dataset에 추가된 Row
update	원본 Dataset에 변경된 Row. Org_Row는 변경 전의 원본 Row
delete	원본 Dataset에 삭제된 Row

- json 포맷 예시

```
{
    "VERSION" : "1.0",
    "PARAMETERS":
    [
        {"ID":"ERRORCODE", "VALUE":0},
        {"ID":"ERRORMSG", "VALUE":""},
        {"ID":"PARAM", "VALUE":"0", "TYPE":"STRING"}
    ],
    "DATASETS" :
    [
        {
            "ID":"INDATA",
            "COLUMNINFO":
            {
                "CONSTCOLUMN":
                [
                    {"ID":"CONSTCOL1", "VALUE":10},
                    {"ID":"CONSTCOL2", "TYPE":"STRING", "SIZE":"256", "VALUE":10
                ],
                "COLUMN" :
                [
                    {"ID":"COLUMN0"},
                    {"ID":"COLUMN1", "TYPE":"STRING", "SIZE":"256"},
                    {"ID":"COLUMN2", "TYPE":"STRING", "SIZE":"256"}
                ]
            },
            "ROWS":
            [
                {"_ROWTYPE_":"U", "COLUMN0":"", "COLUMN1":"ZZZ", "COLUMN2":""},
                {"_ROWTYPE_":"O", "COLUMN0":"", "COLUMN2":""},
                {"_ROWTYPE_":"N", "COLUMN0":"A", "COLUMN1":"B", "COLUMN2":""},
                {"_ROWTYPE_":"D", "COLUMN0":"A", "COLUMN1":"B", "COLUMN2":"C"},
                {"_ROWTYPE_":"I", "COLUMN0":"", "COLUMN1":"", "COLUMN2":""}
            ]
        }
    ]
 }
```


※ nexacro json 데이터 포맷 참조 URL: http://docs.tobesoft.com/advanced_development_guide_nexacro_n_ko/8518d6570e98b4ad

※ 레코드의 형태(_RowType_)별 속성 값

RowType	설명
N	Normal Record (Default)
I	Inserted Record
U	Updated Record
D	Deleted Record
O	바로 앞 Row의 Original Record (Update Record의 원본 레코드)

3.2.2 x-api 모듈

uiadapter 에서 사용하는 데이터 변환 기능은 xapi 를 기반으로 한다. xapi 의 모듈에 대한 배포는 고객지원센터 > Product > [nexacro] > Download 메뉴 (https://support.tobesoft.co.kr/Support/?menu=Download_N&page=1) 에서 zip 파일 형태로 제공한다. maven repository 가 필요할 경우 아래와 같이 제공한다.

Repository 의 모듈과 버전은 고객지원센터의 Download 의 배포버전과 동일하다.

```
<dependency>
  <groupId>com.nexacro</groupId>
  <artifactId>nexacroN-xapi-jakarta</artifactId>
  <version>1.2.4</version>
</dependency>
```

xapi 의 release note url 은
https://docs.tobesoft.com/product_information_nexacro_n_v24_ko/release_note_XAPI
이다.

- PlatformData

  nexacro platform 에서 사용되는 데이터 형식이다.

  PlatformData 는 데이터 구조의 최상위에 위치하고 있으며, VariableList 와 DataSetList 를 가지고 있다. 데이터는 크게 단일 데이터(Variable)와 2 차원 데이터(DataSet)로 구분한다. 데이터 송수신의 기본 단위로 사용한다.

- DataSetList

Dataset 을 List 형태로 가진다. 2 차원 데이터를 저장하는 DataSet 은 열(column)과 행(row)으로 구성되면 각각의 column 은 Type 및 Size 속성을 가진다.

- VariableList

단일 데이터는 데이터를 구분할 수 있는 식별자(name)와 값(value)을 가지고 있으며, 이는 VariableList 에 저장된다.

자세한 내용은 nexacro platform 의 API 문서를 참조한다

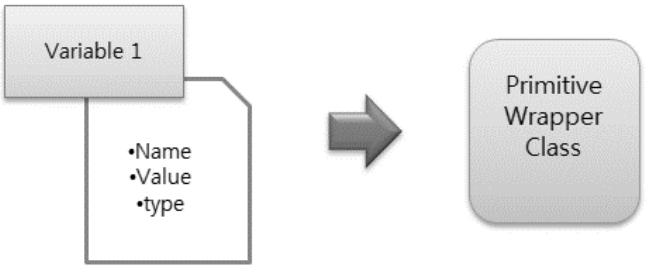
3.2.3 Data 변환

데이터 변환은 크게 2 가지 형태로 데이터 변환을 수행한다. (역관계의 데이터 변환도 지원한다.)

3.2.3.1 Variable 변환

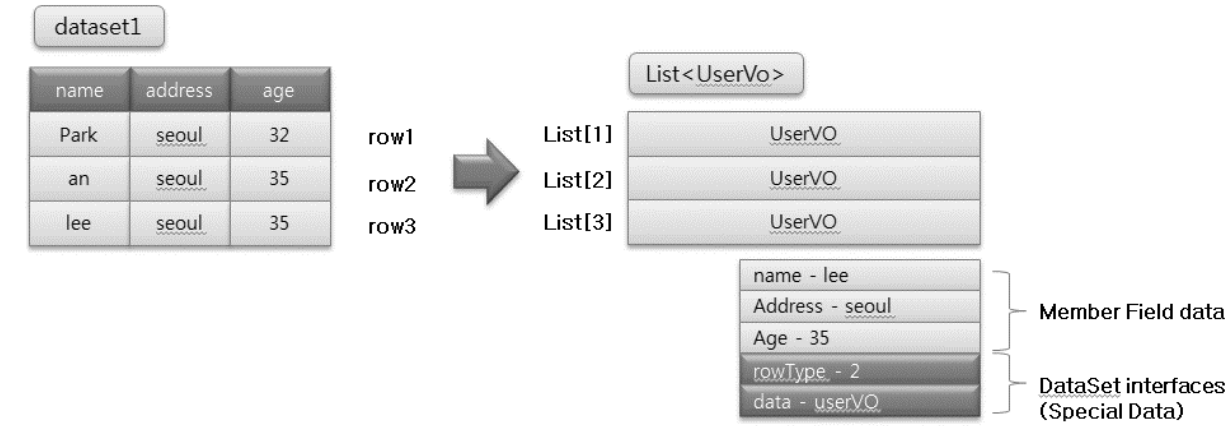
Variable을 primitive Type 혹은 Wrapper class로 변환 한다.

- primitive Type (byte[], int, long, float, double, boolean): Wrapper class 로 변환
- 객체 Type: java.lang.String, java.math.BigDecimal, java.util.Date, java.lang.Object

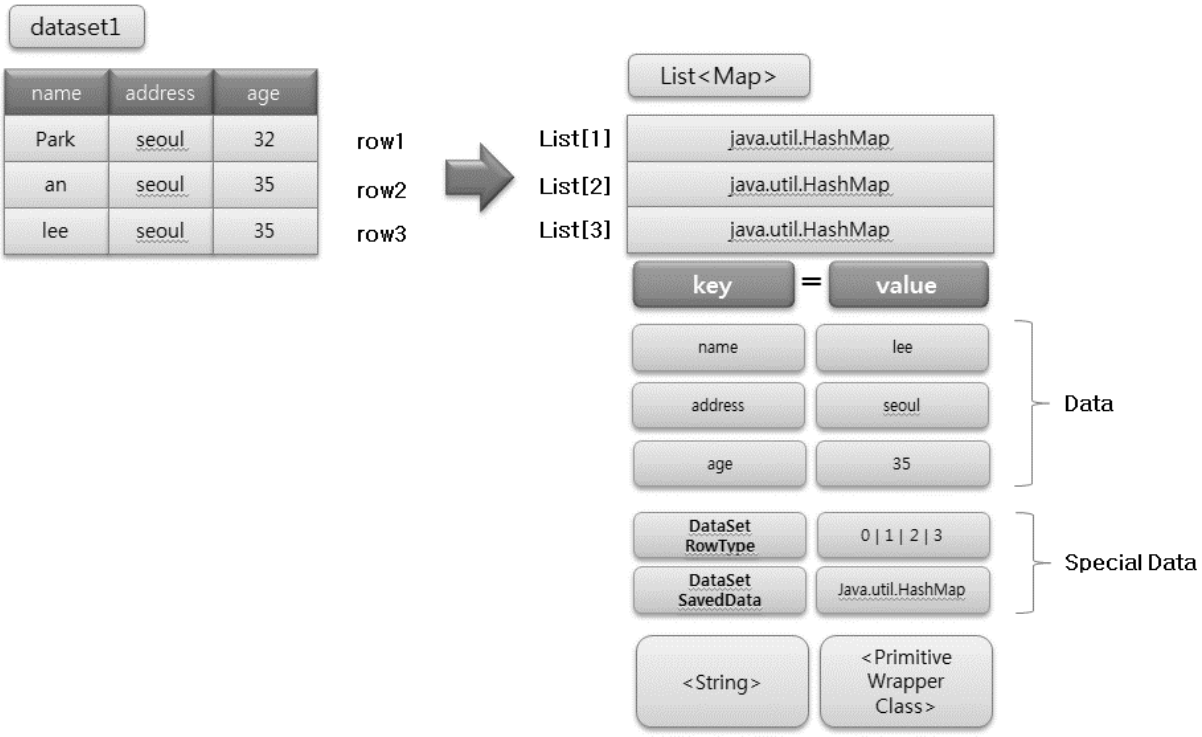


3.2.3.2 DataSet 변환

- ◆ List<VO>: DataSet을 List<VO> 형태로 변환한다.



◆ List<Map> : DataSet을 List<Map> 형태로 변환한다.



3.2.3.3 DataSet ROW_TYPE

DataSet 은 한 번에 입력, 수정, 삭제 처리를 위해 row 별로 상태값(Row Type)을 가지고 있다.

- ROW_TYPE 별 상태값

ROW_TYPE	상태값
com.nexacro.xapi.data.DataSet.*ROW_TYPE_NORMAL*	일반 행 (0)
com.nexacro.xapi.data.DataSet.*ROW_TYPE_INSERTED*	추가 된 행 (1)

com.nexacro.xapi.data.DataSet.*ROW_TYPE_UPDATED*	값이 변경 된 행 (2)
com.nexacro.xapi.data.DataSet.*ROW_TYPE_DELETED*	삭제된 행 (3)

- VO 데이터 객체 RowTypeAccessor

  VO 객체로 데이터를 변환할 경우 DataSetRowTypeAccessor interface 를 구현한다.

  com.nexacro.uiadapter.boot.core.data 패키지에 들어있다.

  ◆ DataSetRowTypeAccessor

```
public interface DataSetRowTypeAccessor {
	public void setRowType (int rowType);
	public int getRowType ();
}
```

  ex) DataSetRowTypeAccessor 구현 VO

```
public class DefaultVO implements DataSetRowTypeAccessor {
```

- Map 데이터 객체의 RowType

  Map 으로 데이터를 변환할 경우 데이터 변환 시 행의 타입을 획득하기 위한 식별자(key)로 DataSetRowTypeAccessor 의 NAME 값을 이용한다.

  ◆ DataSetRowTypeAccessor

```
public interface DataSetRowTypeAccessor {
	static final String NAME = "DataSetRowType";
```

  ex) DataSetRowTypeAccessor 를 이용한 RowType 비교

  ROW_TYPE별 상태값에 따라서 분기처리한다.

```
int dataRowType =
Integer.parseInt(String.valueOf(dataMap.get(DataSetRowTypeAccessor.NAME)));

if (dataRowType == DataSet.ROW_TYPE_INSERTED){
	nexacroSampleDAO.insertSampleMap(dataMap);
}
else if (dataRowType == DataSet.ROW_TYPE_UPDATED){
nexacroSampleDAO.updateSampleMap(dataMap);
}
else if (dataRowType == DataSet.ROW_TYPE_DELETED){
nexacroSampleDAO.deleteSampleMap(dataMap);
}
```

3.2.3.4 Saved Data의 원본 Data

DataSet 의 값이 변경되기 이전의 원본 값에 대한 정보이다.

- VO 로 데이터를 변환할 경우
  com.nexacro.uiadapter.boot.core.data.DataSetSavedDataAccessor<T>를 구현한다.
  Saved Data 를 획득하기 위해선 VO class 와 동일한 class 로 선언해야 한다.

  ◆ DataSetSavedDataAccessor

```
public interface DataSetSavedDataAccessor<T> {
      public void setOrgData (T t);
      public T getOrgData ();
}
```

  ◆ Map 일 경우 Map 내부에 원본데이터가 설정된다.

  Key: com.nexacro.uiadapter.boot.core.data.DataSetSavedDataAccessor.NAME 식별자를 사용한다.

  Value: Map 형태의 데이터가 설정된다.

  ex) Map savedDataMap =

mapData.get(com.nexacro.uiadapter.boot.core.data.DataSetSavedDataAccessor.NAME);

DataSet의 RowType은 nexacro에서 transaction 시 입력 데이터셋의 전송옵션이 :U 혹은 :A 일 때 행의 타입을 확인할 수 있다. :N 옵션일 경우에는 Normal 상태(default) 이다.
DataSet의 SavedData (원본 데이터)의 경우, :A 혹은 :U 옵션으로 데이터를 전송할 경우에만 저장된다. 자세한 내용은 nexacro studio의 help 창의 transaction 옵션을 참고한다.

3.2.4 Data Type 맵핑

nexacro platform 과 java 간의 데이터 타입에 대한 Mapping 정보는 다음과 같다.

Java types	nexacro platform types
byte[], java.lang.Byte[]	PlatformDataType.BLOB
int, java.lang.Integer	PlatformDataType.INT
long, java.lang.Long	PlatformDataType.LONG
float, java.lang.Float	PlatformDataType.FLOAT
double, java.lang.Double	PlatformDataType.DOUBLE
boolean, java.lang.Boolean	PlatformDataType.BOOLEAN
java.lang.String	PlatformDataType.STRING
Java.math.BigDecimal	PlatformDataType.BIG_DECIMAL
java.util.Date	PlatformDataType.DATE
	PlatformDataType.DATE_TIME
	PlatformDataType.TIME
java.lang.Object	PlatformDataType.UNDEFINED

3.3 Controller 데이터 처리

Controller 는 사용자의 요청(request)을 1:1 로 맵핑하여 처리하는 클래스이다. nexacro 에서 보내는 request data 를 받아서 Service 에 처리를 위임하고 그 결과를 리턴한다. Spring boot 는 리턴한 NexacroResult 객체를 반환하면 DispatcherServlet 에 서 ViewResolver 로 NexacroResult 를 처리할 뷰(NexacroView)에 처리를 위임한다.

ex) request 맵핑

```
@RequestMapping (value = "searchSampleList.do")
public NexacroResult searchCodeList(@ParamDataSet(name = "inDs") List<Map<String,
Object>> inDs) throws Exception  {
```

3.3.1 Controller 함수 Arguments

nexacro 에서 보내는 request data 는 Controller 함수 arguments 에서 ParamDataSet 과 ParamVariable 로 받을 수 있다.

3.3.1.1 Controller arguments

Controller 에서 입력 arguments 로 다음과 같은 데이터 형식을 사용할 수 있다.

class 명칭	설명	비고
List<?> list	DataSet 을 List로 converting한 Data (반드시 Generic을 사용해야 한다. Map 혹은 VO class만 지정가능하다.)	Converting된 Data @ParamDataSet과 함께 사용
Java Bean (VO) or Map	DataSet의 첫번째 행의 값을 Java Bean 혹은 Map으로 변환한다.	Converting된 Data @ParamDataSet과 함께 사용
Primitive type or Wrapper	Variable을 Object로 converting한 Data	Converting된 Data @ParamVariable과 함께 사용
PlatformData	nexacro에서 사용되는 모든 데이터	nexacro x-api 지원 객체
DataSetList	Dataset의 list	nexacro x-api 지원 객체
VariableList	Variable 의 List	nexacro x-api 지원 객체
Dataset	2차원의 Data	nexacro x-api 지원 객체
Variable	Key-value 형식의 변수 Data	nexacro x-api 지원 객체
HttpPlatformRequest	http 요청으로 PlatformData를 가진다.	
HttpPlatformResponse	http 응답으로 PlatformData를 가진다.	
NexacroFirstRowHandler	http 응답으로 PlatformData를 분할 전송한다.	대용량 데이터 처리를 위해 사용한다.

- 입력 arguments 사용 예시

```
@RequestMapping (value = "/test.do")
public NexacroResult test (
        @ParamDataSet(name="dsUnit") List<UnitVO> unitList
        , @ParamDataSet(name="dsUnit") List<Map> unitMapList
        , @ParamDataSet(name="dsUnit") DataSet dsUnit

        , @ParamVariable(name="intValue") int iVar1
        , @ParamVariable(name="intValue") Variable iVar2
        , @ParamVariable(name="stringValue") String sVar1
        , @ParamVariable(name="stringValue") Variable sVar2

        , DataSetList dsList
        , VariableList varList
        , PlatformData platformData
        , HttpPlatformRequest platformRequest
        , HttpPlatformResponse platformResponse
        , NexacroFirstRowHandler firstRowHandler
        ){

    // control nexacro result...
    NexacroResult result = new NexacroResult();
    result.addDataSet("dsUnitList", unitList);
    result.addVariable("responseInt", iVar1);
    result.addVariable("responseString", sVar1);

    return result;
}
```

여러가지 arguments 유형중에 원하는 argument 만 사용하면 된다.

3.3.2 NexacroResult

Controller 함수의 리턴 객체로 DataSet 과 Variable 값을 담고 있고 View 단에서 response 하기 위해 사용된다. 데이터 삽입 시 변환은 이루어지지 않으며, NexacroHandlerMethodReturnValueHandler 에서 데이터 변환이 이루어진다.

- NexacroResult 의 주요 내장 객체는 다음과 같다.

내장 객체	설명
dataSetMaps	request의 처리 결과를 dataset 이름을 키와 List<?>를 담는다
variableMaps	request의 처리 상태와 변수를 variable 이름을 key로 담는다
platformData	송신을 위한 객체로 dataSetMaps과 variableMaps 을 담는다.

- 주요 메서드는 다음과 같다.

함수	설명
addDataSet	void addDataSet(String name, List<?> list)
	nexacro으로 전송할 출력 DataSet(List<?>)을 추가한다. Controller의 입력 파라미터 정보와 같이 반드시 List<VO> 혹은 List<Map> 형태의 데이터만 설정가능하다.
addDataSet	void addDataSet(String name, Object bean)
	nexacro으로 전송할 출력 DataSet(Object)을 추가한다. Controller의 입력 파라미터 정보와 같이 반드시 Java Bean 혹은 Map 형태의 데이터만 설정가능하다.
addVariable	void addVariable(String name, Object value)
	nexacro으로 전송할 출력 Variable(Object)를 추가한다. Data 변환의 Data Type Mapping 정보를 참조한다
setErrorCode	void setErrorCode(int errorCode)
	에러코드를 설정한다
setErrorMsg	void setErrorMsg(String errorMsg)
	에러 메시지를 설정한다

3.3.3 NexacroFileResult

nexacro 으로 File 데이터를 송신하기 위한 정보를 가진다.

- 주요 메서드는 다음과 같다.

함수	설명
setContentType	void setContentType(String contentType)
	파일 데이터의 MimeType을 설정한다
setFile	void setFile(File file)

	전송하기 위한 파일 데이터를 설정한다
setOriginalName	void setOriginalName(String originalName)
	전송 시 사용되는 파일의 명칭을 설정한다

3.4 데이터 조회 시 0 건 처리

nexacro 의 경우 서버에서 전송된 DataSet 의 스키마 정보가 존재 할 경우 손쉽게 UI 를 구성할 수 있다. 하지만 DataSet 의 스키마 정보가 전송되지 않을 경우 UI 개발자는 별도로 스키마 정보를 정의해서 사용해야 하는 불편함이 있다. mybatis 를 사용하여 데이터를 조회하였을 때 조회된 데이터가 없을 경우 (Map 형태의 데이터 사용시) 스키마에 대한 정보를 확인할 수 없어 DataSet 으로 변환 시 스키마 정보가 생성되지 않는다. 이를 처리하기 위해 mybatis 를 통해 조회된 데이터가 0 건일 경우 재조회를 통해 ResultSet 의 MetaData 정보를 이용하여 스키마 정보를 획득한다.
아래는 스키마 정보 획득 시 처리되는 형태에 대한 정보이다.

3.4.1 resultType 별 처리 방식

- Java Bean(VO) 사용 시

  추가적인 데이터 조회는 이루어지지 않으며 Java Bean(VO) class 정보만을 획득하여 스키마를 구성한다.

- java.util.Map 사용시 - (java.util.Map)을 사용하지 않을 경우 구현 필요 없음)

  ResultSet 의 MetaData 와 ResultType 에 명시된 정보를 바탕으로 스키마 정보를 획득한다.

ResultSet 으로부터 MetaData 를 획득 할 경우 DBMS 에 따라 컬럼 맵핑 정보가 다를 수 있어서 추가적인 처리가 필요할 수도 있다.

3.4.2 DBMS 유형별 처리

DBMS에 따라 컬럼의 맵핑 방식 차이에 따라 추가적인 처리가 필요할 수도 있다. 타입 변환 처리를 수행할 수 있도록 구현된 DB 는 다음과 같다

DB명	DB 이름
Oracle	"Oracle"
msSQL	"Microsoft SQL Server"
mysql	"MySQL"
mariadb	"MariaDB" or "MySQL"
postgreDB	"PostgreSQL"

아래는 Mssql 의 고유한 데이터 타입 변환을 위해 제공되는 class 인 Mssql 의 코드 내용이다.

```
public class Mssql extends AbstractDbms {
    @Override
    public void handleColumnDataType (final DbColumn column) {
        if (column==null) {
            return;
        }
        if ("xml".equals(column.getVendorsTypeName())) {
            column.setDataType(PlatformDataType.STRING);
        } else if ("image".equals(column.getVendorsTypeName())) {
            column.setDataType(PlatformDataType.BLOB);
        } else if ("int".equals(column.getVendorsTypeName())) {
            column.setDataType(PlatformDataType.INT);
        } else if (column.getVendorsTypeName().endsWith("time")) { //datetime,
smalldatetime
            column.setDataType(PlatformDataType.DATE_TIME);
        } else if (column.getVendorsTypeName().startsWith("time")) { //datetime2,
datetimeoffset
            column.setDataType(PlatformDataType.DATE_TIME);
        } else if (column.getVendorsTypeName().endsWith("money")){ //money, smallmoney
            column.setDataType(PlatformDataType.BIG_DECIMAL);
        } else if (column.getVendorsTypeName().endsWith("text")){ //text, ntext
            column.setDataType(PlatformDataType.STRING);
```

```
        }
    }
}
```

단, DB의 Vendor 별 데이터 타입 처리가 상이할 수 있기 때문에 정상적이지 않을 경우 해당 클래스를 상속받아 수정해야 한다..

3.4.3 Mybatis 설정

Mybatis 는 메타 데이터 정보를 조회하기 위해 설정파일에 Plugin 형태로 등록하여 처리한다. 아래는 sql-mapper-config.xml 에 설정되는 Plugin 항목들이다.

자세한 설정은 "4.3.5 mybatis 설정" 부분을 참조한다.

3.4.4 DbVendorsProvider 지정

nexacro 의 request 처리 결과가 0 건일 때 컬럼 정보를 가져오기 위한 설정이다. uiadapter 에서 0 건 데이터를 처리하기 위해서 DBVendor 별로 컬럼 맵핑을 처리하는 Class 를 제공하도록 하고 있다. 왜냐하면 기본적인 컬럼 맵핑은 java 에서 처리해 주지만 DBMS 에 따라 컬럼 타입이 특이한 경우, 컬럼 타입 맵핑을 위하여 추가적인 처리를 한다. 이 과정을 DbVendorsProvider 를 통해 처리한다.

DbVendorsProvider 설정은 4.3.6 UiadapterWebMvcConfig.java 설정 부분을 참조한다.

3.5 다국어 설정

3.5.1 Locale 설정

다국어 처리를 하기 위해 Spring MVC 에서 제공하고 있는 LocaleResolver 를 이용한다. 설정은 4.3.6 UiadapterWebMvcConfig.java 파일에 있다.

```
@Bean
public SessionLocaleResolver localeResolver () {
    SessionLocaleResolver localeResolver = new SessionLocaleResolver ();
    return localeResolver;
}
```

3.5.2 Message 처리

다국어 처리를 통해 이용하는 기능 중에 하나는 Message 처리가 있다. Message 의 처리는 Spring 에서 지원하는 MessageSource, MessageSourceAccessor 를 이용한다.

```
@Bean
public ReloadableResourceBundleMessageSource messageSource() {
    ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
    messageSource.setBasename("classpath:/messages/message-common");
    messageSource.setDefaultEncoding("UTF-8");
    messageSource.setCacheSeconds(10 * 60); // 리로드 시간
    return messageSource;
}
@Bean(name="messageSourceAccessor")
public MessageSourceAccessor messageSourceAccessor() {
    return new MessageSourceAccessor(messageSource());
}
@Bean(name="message")
public MessageUtils message() {
    MessageUtils messageUtils = new MessageUtils();
    messageUtils.setMessageSourceAccessor(messageSourceAccessor());
    return messageUtils;
}
```

- **MessageSource**

MessageSource 는 스프링에서 제공하는 인터페이스로서 code 값과 arguments 를 통해 .properties, .xml 등에 작성한 메시지를 읽어 동적으로 메시지를 만들어 준다.

- **MessageSourceAccessor**

MessageSourceAccessor 는 MessageSource 내부 변수로 가지고 있으면서 MessageSource 기능을 편리하게 사용할 수 있도록 구현된 클래스다.

- **MessageUtils**

Message 기능을 편리하게 사용하기 위한 Utility 클래스이다.

3.5.3 Message 기능 사용

Message 기능을 사용할 때 @Autowired 로 MessageUtils 를 받고 사용한다.

```
@Autowired
private MessageUtils message;
. . . .
log.debug(message.getMessage("error.minlength", new String[] {"비밀번호", "2"}));
```

- Message 파일

message-common_ko_KR.properties 파일의 `error.minlength` 를 참조한다.

```
error.common=공통 오류가 발생했습니다.
error.minlength=default {0}은 {1}자 이상의 문자를 입력하셔야 합니다.
```

- Message 처리 결과

```
DEBUG - default 비밀번호은 2자 이상의 문자를 입력하셔야 합니다.
```

3.6 예외처리

예외 처리 시 메시지 처리를 하기 위해선 NexacroMappingExceptionResolver 에 MessageSource 를 등록한다. 자세한 설정은 4.3.6 UiadapterWebMvcConfig.java 파트를 참조한다.

3.6.1 NexacroException

uiadpater 는 nexacro 으로 예외를 처리하기 위해서 예외 발생시 NexacroException 을 이용하여 처리한다. com.nexacro.boot.NexacroException 의 주요 메서드는 다음과 같다.

메서드	기능
NexacroException(String message)	입력받은 message를 통해 NexacroException을 생성
NexacroException(String message, Throwable cause)	입력받은 message와 cause를 통해 NexacroException을 생성
NexacroException(String message, int errorCode, String errorMsg)	입력받은 message, 에러코드, 에러메시지를 통해 NexacroException을 생성
NexacroException(String message, Throwable cause, int errorCode, String errorMsg)	입력받은 message와 cause, 에러코드, 에러메시지를 통해 NexacroException을 생성
setErrorCode(int errorCode)	nexacro으로 전송 할 에러코드를 설정. 설정하지 않을 경우 '**-1**' 을 전송
setErrorMsg(String errorMsg)	nexacro으로 전송 할 에러메세지를 설정 설정하지 않을 경우 NexacroException 생성 시 전달 받은 메시지를 전송

3.6.2 예외를 처리하는 예시

```
try {
      .....
} catch(Exception e) {
      throw new NexacroException("error log message", e, -8859, "send message");
}
```

NexacroMappingExceptionResolver 의 shouldSendStackTrace property 설정값에 따라 응답으로 전송되는 에러 정보가 달라진다. shouldSendStackTrace 가 false 로 설정되어 있을 경우 예외의 message 는 전송되지 않으며, errorMsg (에러메시지) 만 응답으로 전송된다. 에러에 대한 정보는 서버의 로그를 통해 확인한다.

nexacro 에서의 기본적인 에러처리 형식은 응답헤더 코드는 200(OK)으로 전송되며, 내부에 Variable (ErrorCode, ErrorMsg)을 통해 처리 한다.

transaction 메서드의 콜백 함수에 처리 된다. 그 외 응답코드(4xx, 5xx 등) 에 대한 부분은 ADL 파일의 onerror 이벤트에서 처리 한다.

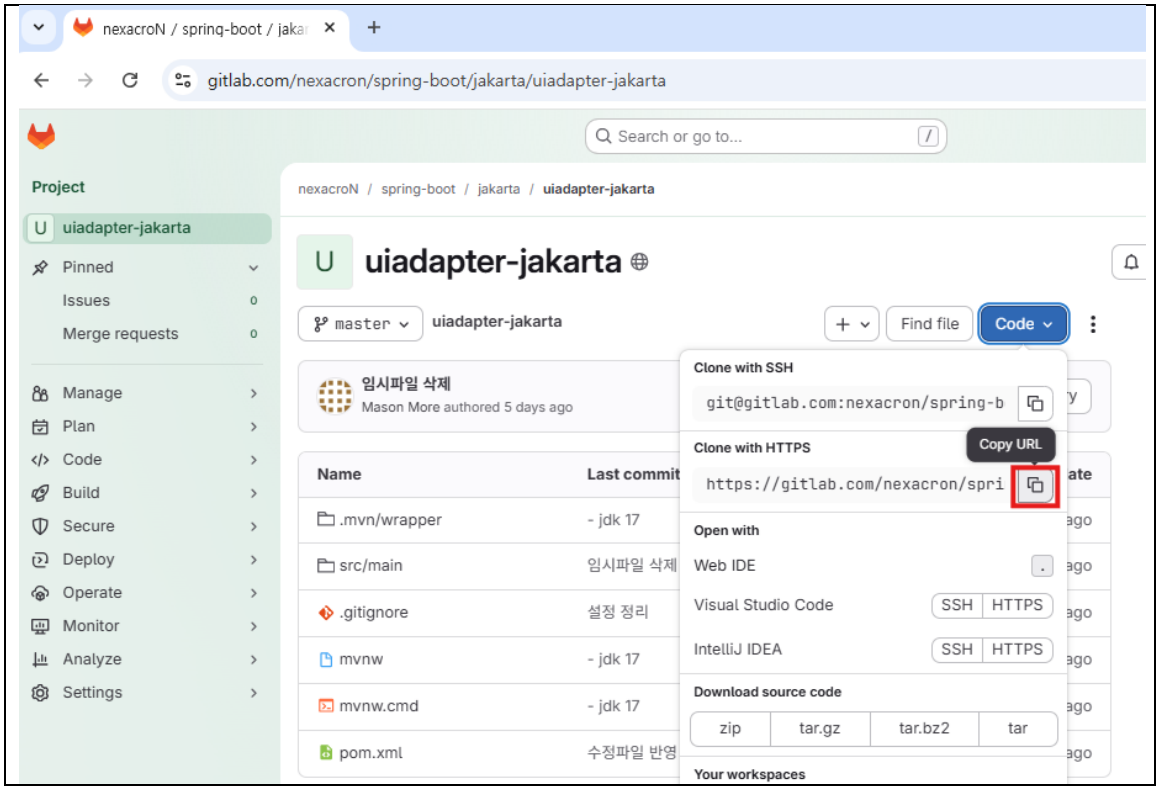
4. uiadapter기반 샘플 프로젝트

4.1 gitlab 프로젝트 clone

샘플 프로젝트는 gitlab Repository 를 통해 제공한다. Gitlab 에 있는 샘플 프로젝트 URL 은 https://gitlab.com/nexacron/spring-boot/jakarta/uiadapter-jakarta 이다. 브라우저의 열린화면에서 [code]를 클릭하면 "Clone with HTTPS" 메뉴의 copy url 아이콘을 선택한다.

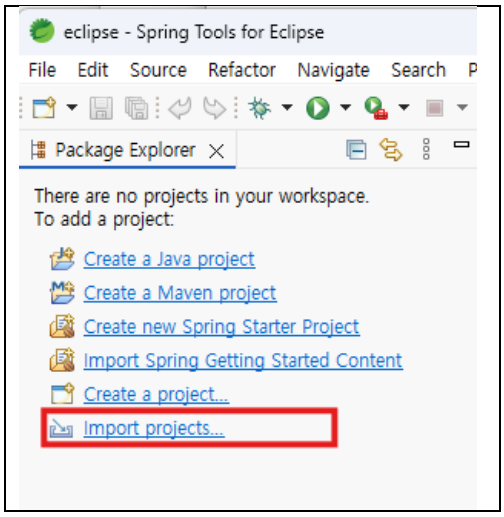
Copy 한 gitlab clone URL 은 다음과 같다.

- https://gitlab.com/nexacron/spring-boot/jakarta/uiadapter-jakarta.git

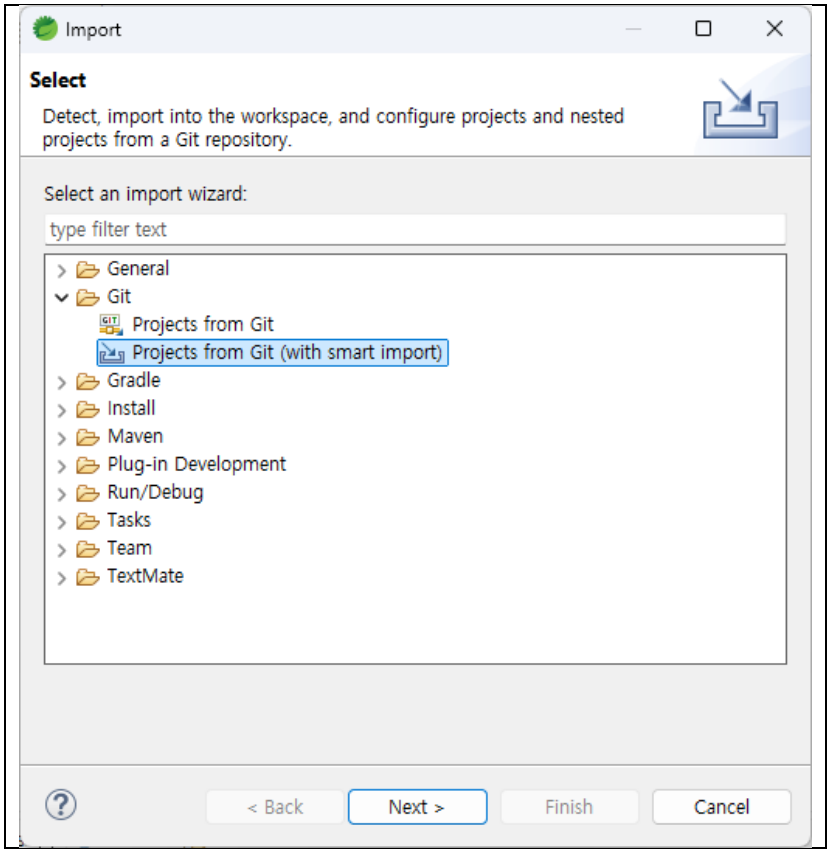


4.1.1 Import project

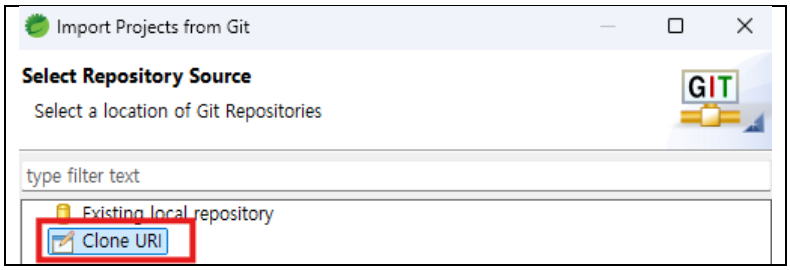
Eclipse 를 실행하고 Package Explorer 에서 Import Projects 를 선택한다.



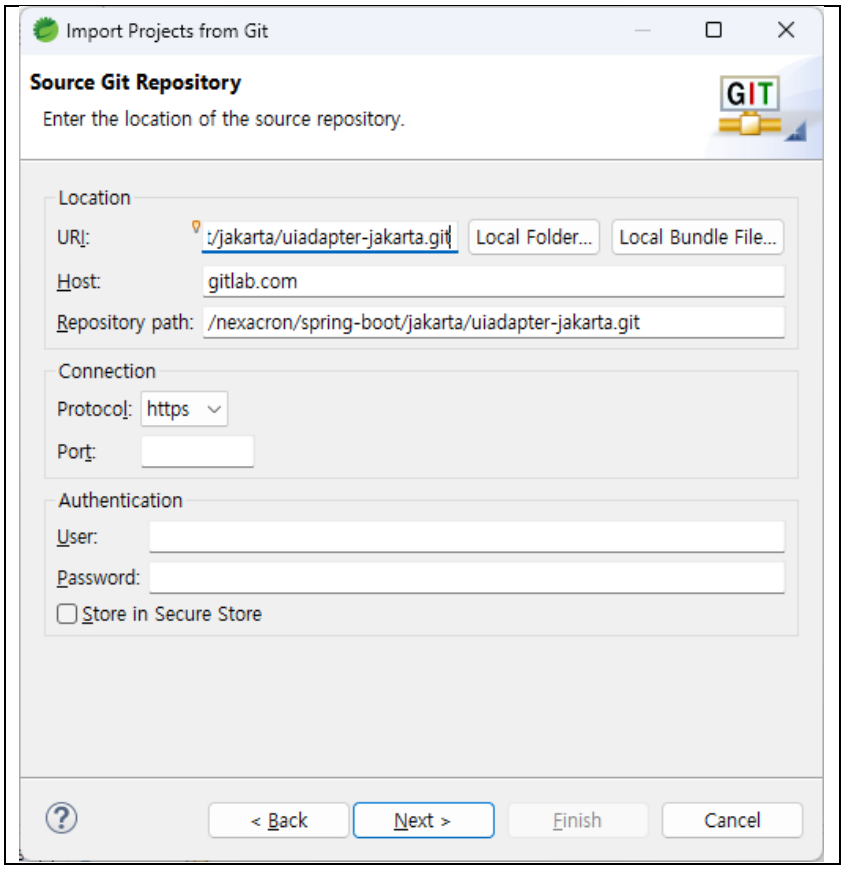
팝업창 Import 가 뜨면 >Git>Projects from Git(with smart import)를 선택한다.



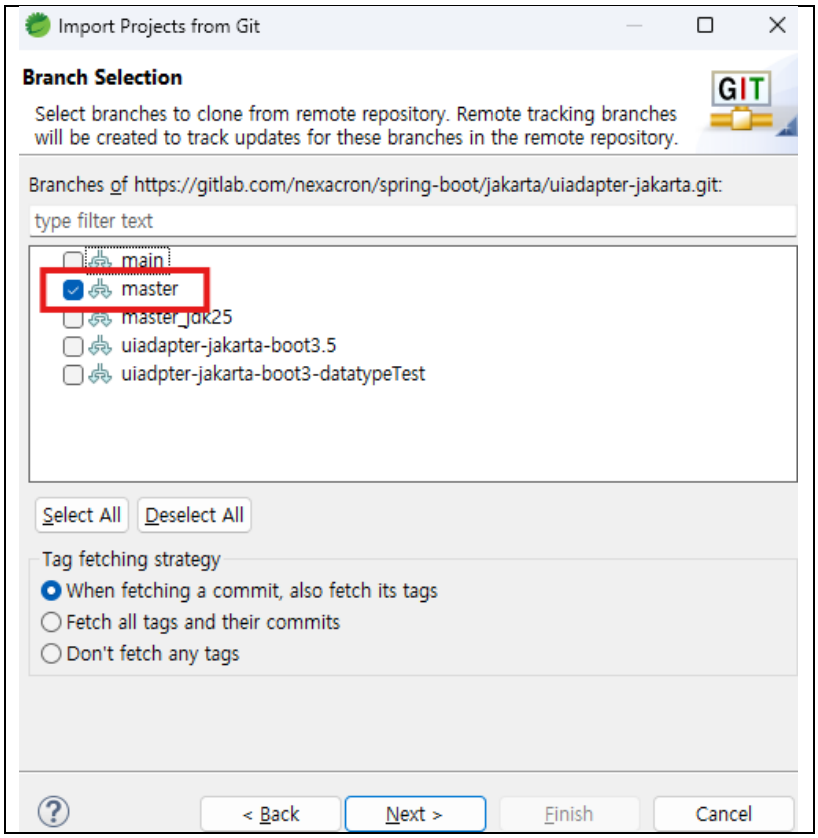
이어서, Repository Source 는 Clone URI 를 선택한다.



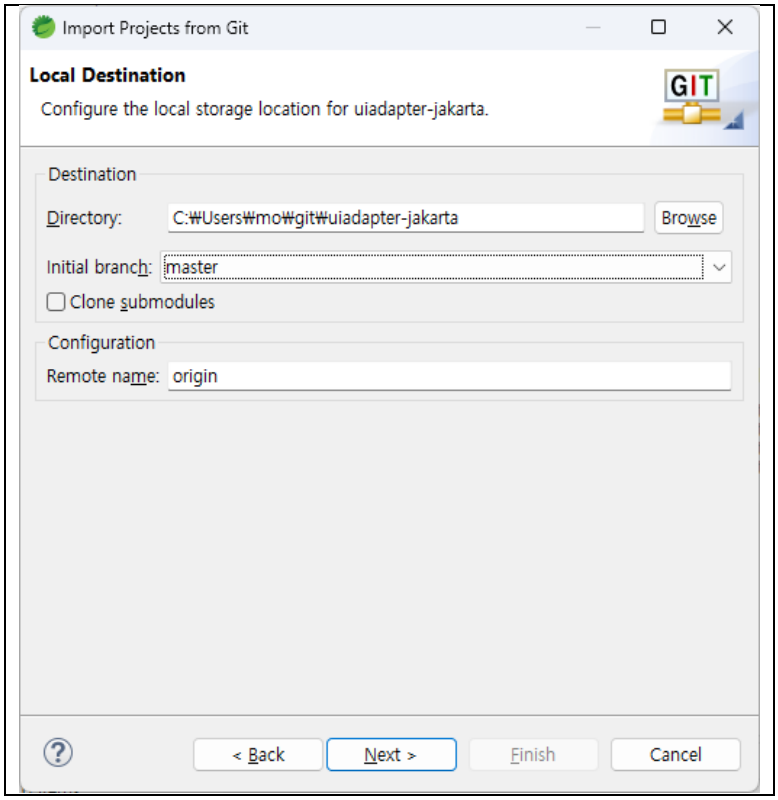
앞서 Copy 한 gitlab clone URL 을 URI 항목에 붙여넣는다. 그러면 Host 와 Repository path 값이 함께 세팅된다.



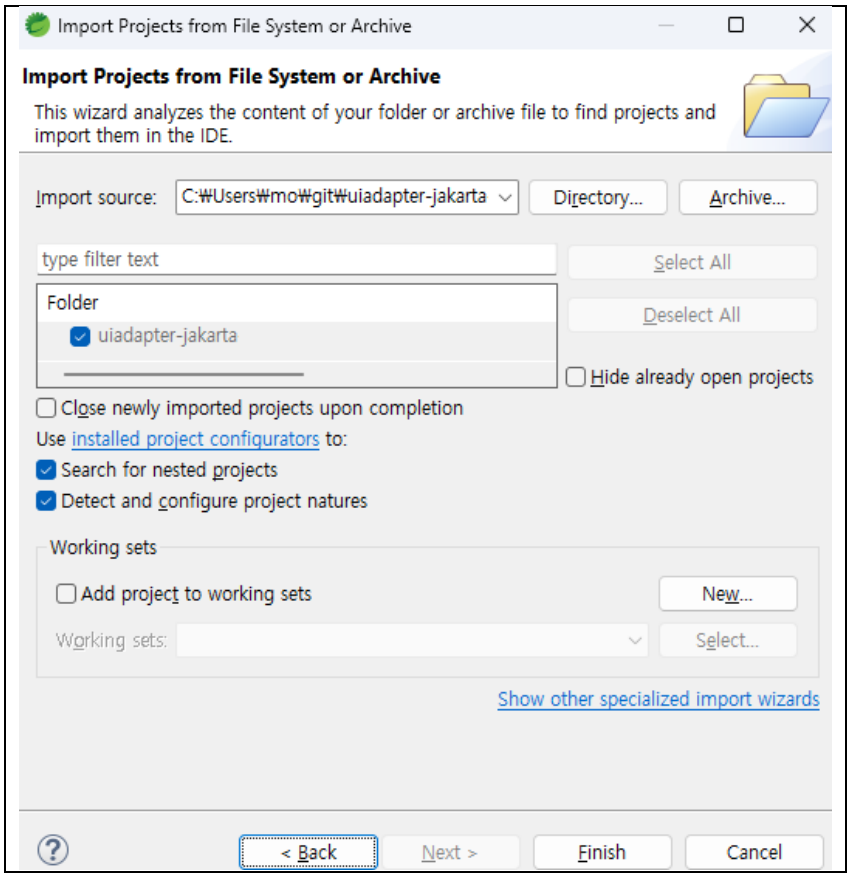
이어서 Branch Selection 화면에서는 master 만 선택하고 넘어간다.



이어서 Local Destination 을 확인한다.



최종 화면은 다음과 같다.

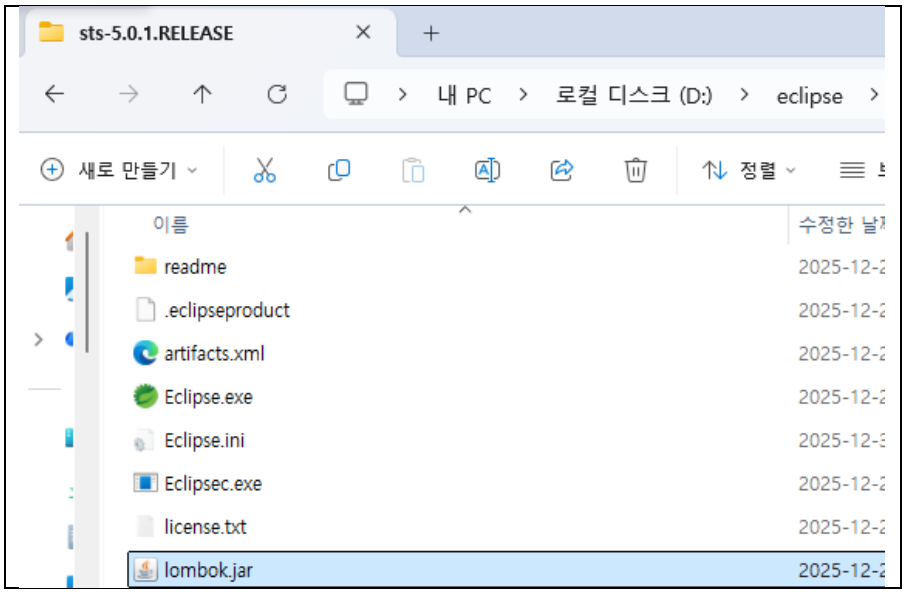


4.1.2 Project 설정 확인

샘플 프로젝트는 lombok 을 사용하였다. 따라서 실행중인 eclipse 에 lombok 을 설치하지 않았다면 프로젝트가 import 된 후 lombok 을 설치한다.

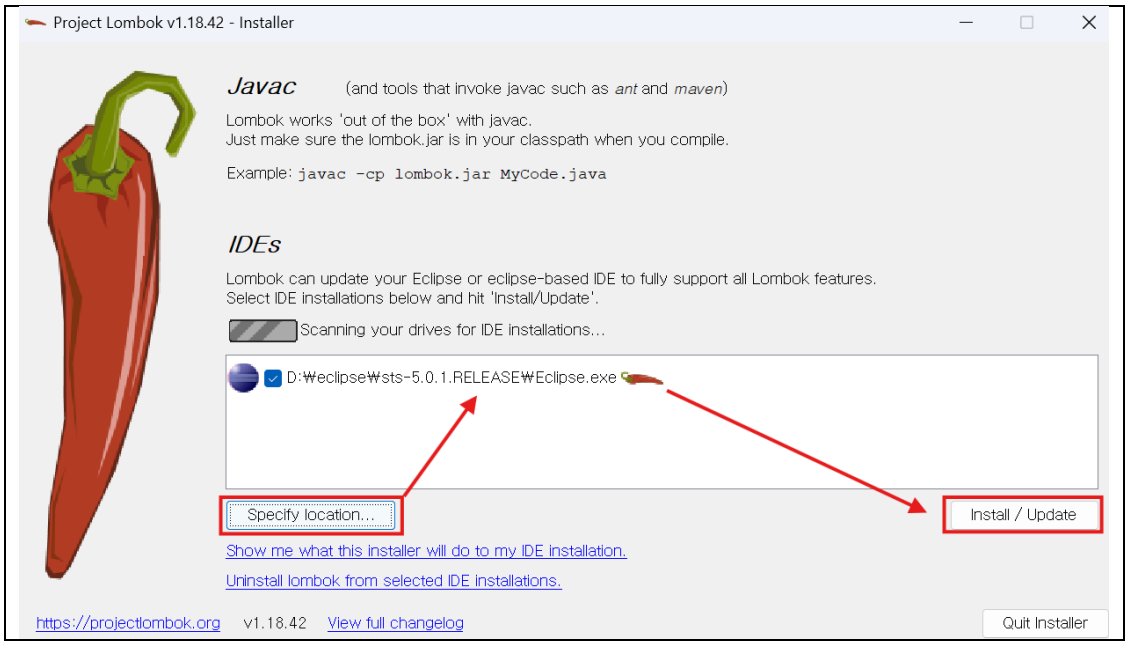
Project Explorer > Maven Dependencies 에서 로딩된 lombok-1.18.42.jar 를 찾는다.

lombok-1.18.42.jar 파일을 eclipse 가 설치된 폴더에 붙여 넣는다.



콘솔창을 관리자권한으로 열고 해당 경로로 이동하고 lombok.jar 를 실행한다.

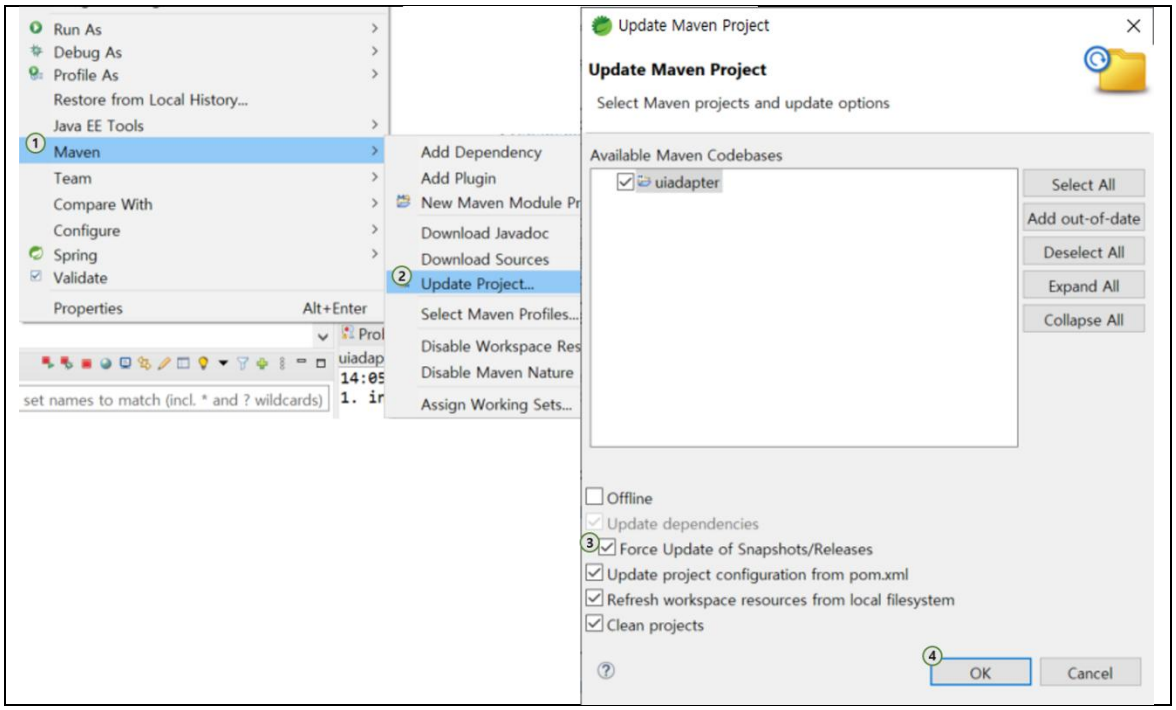
Installer 가 실행되면 [Specify location..]를 클릭하고 팝업된 탐색기에서 이클립스 실행파일(eclipse.exe 혹은 SpringToolsForEclipse.exe)를 선택하고 [install/update]를 클릭한다.



정상적으로 설치가 완료되면 다음과 같은 메시지가 나온다.

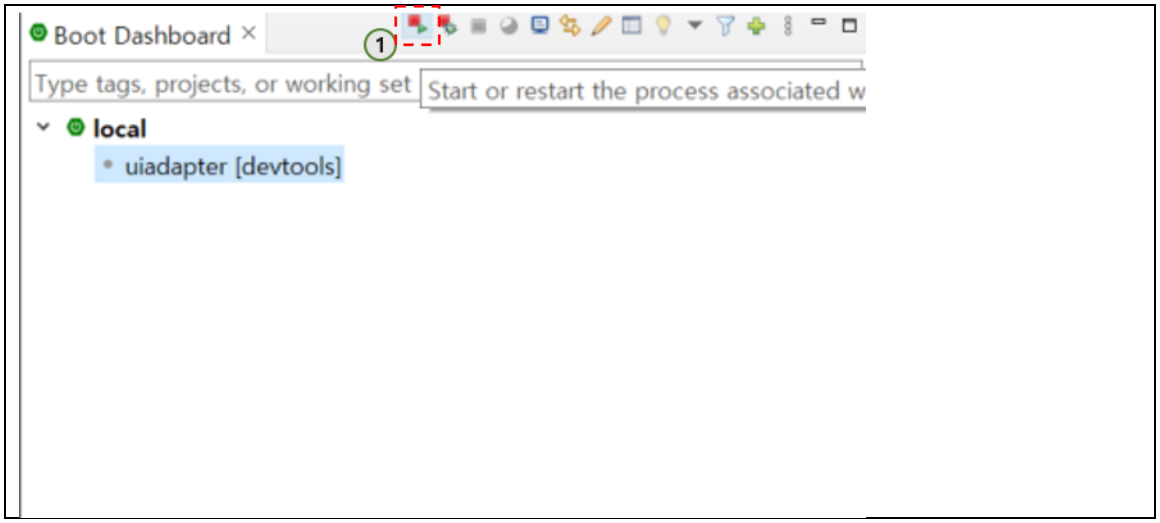
- 메뉴에서 Project > Clean 을 실행한다.
- ProjectExplorer 에서 프로젝트를 선택하고 마우스 우클릭하여 팝업메뉴를 뛰운다.

  Maven > Update Project 를 선택하고 Force Update 옵션을 선택한 후 실행한다.



4.1.3 실행

Boot Dashboard 창에서 start 버튼으로 uiadpater 프로젝트를 실행한다.

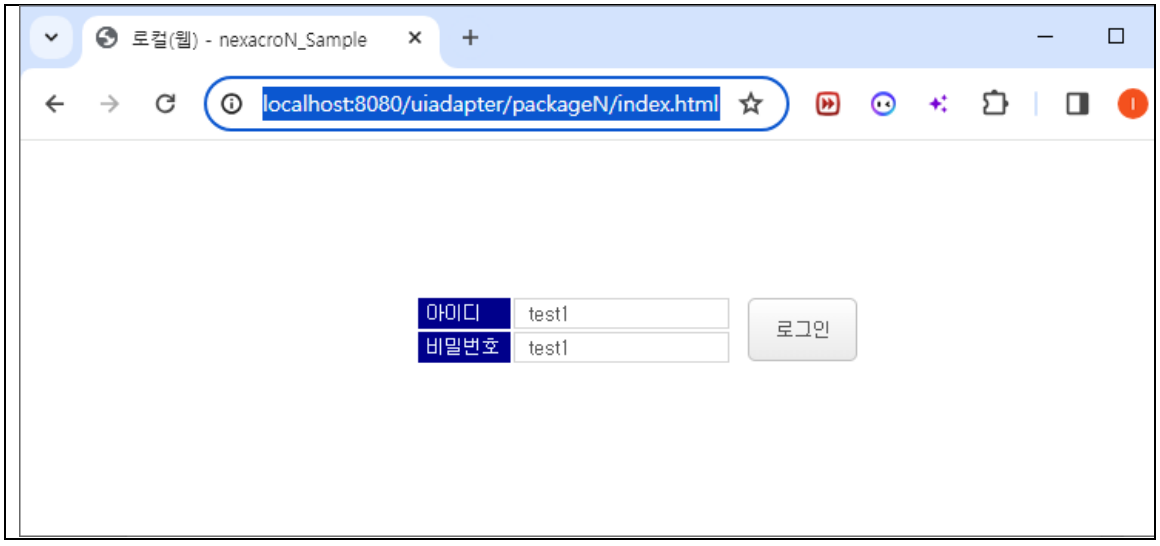


4.1.4 브라우저 호출 확인

브라우저 주소창에 uiadapter 의 실행주소를 입력하고 호출한다.

```
http://localhost:8080/uiadapter/packageN/index.html
```

실행결과는 다음과 같다.



4.2 디렉토리 구조

셈플 프로젝트인 uiadapter 의 디렉토리 구조는 다음과 같다.

업무 파일(Controller, Service, Mapper 파일 등) 의 경로는 다음과 같다.

구분	폴더명
Controller	example.nexacro.uiadapter.web
Service	example.nexacro.uiadapter.service
ServiceImpl	example.nexacro.uiadapter.impl
DAO	example.nexacro.uiadapter.pojo
Mapper	example.nexacro.uiadapter.mapper

4.3 설정파일

Spring Boot 기반의 uiadapter 에서 사용하는 설정 파일은 src/main/resources 경로에 있고 다음과 같다.

구분	파일명	설명
공통 설정	application.yml	Spring boot 환경의 기본 설정
SQL문 실행 로그	log4jdbc.log4j2.properties	쿼리 실행문에 대한 보다 구체적인 log를 보기 위한 설정
로그 레벨 설정	logback.xml	패키지 및 로그 레벨 별 설정
Excel xeni 설정	xeni.properties	excel import를 처리하는 Handler 설정
MyBatis 설정	mybatis/sql-mapper-config.xml	mybatis 설정 세팅 (java config로 대체)
Mapper 설정	mybatis/mappers/*_mapper.xml	Mybatis mapper 쿼리 설정

4.3.1 application.yml

application.properties 대신 사용한다. 주석처리는 #으로 한 줄을 주석으로 처리할 수 있다. 표기법은 가급적 kebab case(ex. user-name)를 사용한다. 주요 설정은 다음과 같다.

```
# 서버 포트 및 context 경로 설정
server:
  port: 8080
  servlet:
    context-path: /uiadapter
# 스프링 환경 설정
```

```yaml
spring:
  main:
    banner-mode: "console"
  web:
    resources:
      static-locations: classpath:/static/
  servlet:
    session:
      timeout: 1800 # 초
    multipart:
      enabled: true
      max-file-size: 20MB
      file-size-threshold: 1MB
      max-request-size: 20MB
      location: C:/Temp
  sql:
    init:
      mode: always
      platform: hsqldb
# DB datasource 설정
  datasource:
    driver-class-name: net.sf.log4jdbc.sql.jdbcapi.DriverSpy
    url: jdbc:log4jdbc:oracle:thin:@mobiletong.tobesoft.com:1521:XE
    username: guest
    password: guest
# mybatis 설정
mybatis:
  config-location: classpath:mybatis/sql-mapper-config.xml
  mapper-locations: classpath:mybatis/mappers/*_mapper.xml
```

- uiadapter 는 5 가지의 DB datasource 를 지원한다. db 별 datasource 설정 예시는 다음과 같다.

DB	Datasource 예시
oracle	datasource: driver-class-name: net.sf.log4jdbc.sql.jdbcapi.DriverSpy

	url: jdbc:log4jdbc:oracle:thin:@dev.uiadapter.com:1521:XE username: guest password: guest
MSSQL	driverClassName: net.sf.log4jdbc.sql.jdbcapi.DriverSpy url: jdbc:log4jdbc:sqlserver://dev.uiadapter.com:1433;databaseName=DB_TEST; username: sa password: sa
MYSQL	driverClassName: net.sf.log4jdbc.sql.jdbcapi.DriverSpy url: jdbc:log4jdbc:mysql://dev.uiadapter.com:3306/DB_TEST?useSSL=true username: sa password: sa
MariaDB	driverClassName: net.sf.log4jdbc.sql.jdbcapi.DriverSpy url: jdbc:log4jdbc:mariadb://dev.uiadapter.com:4406/DB_TEST username: sa password: sa
Postgresql	driverClassName: net.sf.log4jdbc.sql.jdbcapi.DriverSpy url: jdbc:log4jdbc:postgresql://192.168.1.29:5432/DB_TEST username: sa password: sa

- nexacro 옵션별 기능은 다음과 같다.

옵션	기능
client-column-case **값: camel, snake**	Nexacro의 Column 표기법이 DB의 Column 표기법과 다를 경우 맞춰주는 기능으로 client의 컬럼 표기법을 지정한다. 지원되는 Column Case : **camel**(userName), **snake**(user_name)
db-column-case **값: camel, snake**	db의 컬럼 표기법을 지정한다. 지원되는 Column Case : **camel**(userName), **snake**(user_name)
use-request-charset	Nexacro에서 request를 요청할 때 사용한 charset을 response할 때 사용한다.
use-request-contenttype	Nexacro에서 request를 요청할 때 사용한 contenttype을 response할 때 사용한다
trim-paramdataset	@ParamDataSet으로 전달된 dataset의 앞 뒤 공백 제거
trim-paramvariable	@ParamVariable로 전달된 dataset의 앞뒤 공백을 제거한다.
replace-all-empty-variable	variable 값의 앞뒤에 붙은 공백문자를 제거한다.

※ 옵션의 값이 없으면 옵션 기능을 처리하지 않는다.
옵션 기능은 성능 손실이 발생할 수 있으니 필요할 경우만 사용한다.

4.3.2 log4jdbc.log4j2.properties

spring boot 기반의 환경에서 쿼리 실행문에 대한 보다 구체적인 log 를 보기 위해 Log4jdbc2 와 Slf4j 를 함께 사용한다. 이를 위한 설정은 다음과 같다.

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
log4jdbc.dump.sql.maxlinelength=0
```

- Maven Dependency

```
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4.1</artifactId>
    <version>1.12</version>
</dependency>
```

- application.yml 의 datasource 에서 driver-class-name, url 을 log4jdbc 로 지정한다.

```
spring:
 datasource:
    driver-class-name: net.sf.log4jdbc.sql.jdbcapi.DriverSpy
    url: jdbc:log4jdbc:oracle:thin:@dev.uiadapter.com:1521:XE
```

4.3.3 logback.xml

Spring boot 는 기본적으로 Logging, Log4J2 and Logback 기능을 지원한다. default 로그 레벨은 info 이다. 주요 설정은 다음과 같다.

```
<!-- SQL문만을 로그로 남기며, PreparedStatement일 경우 관련된 argument 값으로 대체된 SQL문이 보여진다. -->
    <Logger name="jdbc.sqlonly" level="OFF" />
    <!-- SQL문과 수행된 시간 정보(milliseconds)를 포함한다. -->
```

```
<Logger name="jdbc.sqltiming" level="DEBUG" additivity="false"/>
<!-- x-api의 로그를 모니터링한다.-->
<Logger name="com.nexacro.java" level="ERROR" />
<!-- uiadapter의 로그를 모니터링한다.-->
<Logger name="com.nexacro.uiadapter.boot" level="ERROR" />
<!-- uiadapter의 로그를 모니터링한다.-->
<Logger name="com.uiadapter" level="ERROR" />
```

- log level 은 TRACE, DEBUG, INFO, WARN, ERROR 5 가지가 있다. 사용하지 않을 경우 OFF 를 지정한다.

level of request *p*	effective level *q*					
	TRACE	DEBUG	INFO	WARN	ERROR	OFF
TRACE	YES	NO	NO	NO	NO	NO
DEBUG	YES	YES	NO	NO	NO	NO
INFO	YES	YES	YES	NO	NO	NO
WARN	YES	YES	YES	YES	NO	NO
ERROR	YES	YES	YES	YES	YES	NO

4.3.4 xeni.properties

nexacro 에서 excel import 처리를 위한 설정 및 DRM 처리를 위한 옵션을 설정할 수 있다.

```
xeni.multipart.proc=com.nexacro.uiadapter.boot.excel.servlet.XeniMultipartHandler
#xeni.exportimport.storage=com.extend.userExtendClass
```

- xeni.properties 옵션 기능은 다음과 같다.

옵션	기능
xeni.multipart.proc (필수)	excel import를 처리할 Handler를 지정한다. default값은 uiadapter의 내장 Handler를 사용한다.
xeni.exportimport.storage (옵션)	excel import/export 할 때 DRM을 처리할 클래스를 지정한다. DRM처리가 필요한 경우만 사용한다.
xeni.error.minmessage (옵션)	true : XENI 호출 시 onerror가 발생하는 경우에도 에러메시지를 출력하지 않도록 한다. (1.5.20 버전부터 지원)

※ DRM 처리 참조 URL : http://docs.tobesoft.com/server_setup_guide_nexacro_n_ko/ff4aafb511251833

※ xeni release note : https://docs.tobesoft.com/product_information_nexacro_n_v24_ko/release_note_XENI

- 제니에서 지원하는 파일 포맷은 다음과 같다

파일 형태	확장자	기능
Microsoft Excel 97-2003	xls	export / import
Microsoft Excel 2007/2010	xlsx	export / import
CSV format file	csv	import
HancomOffice Hancell 2010	cell	export
HancomOffice Hancell 2014	cell	export / import

4.3.5 mybatis 설정 파일

application.yml 에서 mybatis 관련 설정에서 지정한 설정파일과 mapper 파일이 있다.

- **sql-mapper-config.xml**

mybatis 에 적용할 설정에 대한 파일이다. Null 값 혹은 column case 변환에 대한 옵션 등을 지정할 수 있고 이 외에도 다양한 옵션을 지원한다.

/resources/mybatis/ 아래 위치하고 주요 설정은 다음과 같다.

```
<configuration>
  <settings>
    <setting name="callSettersOnNulls"   value="true"/>
    <setting name="returnInstanceForEmptyRow" value="true"/>
    <setting name="jdbcTypeForNull" value="VARCHAR"/>
    <setting name="mapUnderscoreToCamelCase" value="false"/>
  </settings>
  <plugins>
    <plugin
interceptor="com.nexacro.uiadapter.boot.dao.mybatis.NexacroMybatisMetaDataProvider" />
    <plugin
interceptor="com.nexacro.uiadapter.boot.dao.mybatis.NexacroMybatisResultSetHandler" />
  </plugins>
</configuration>
```

※ uiadapter 는 Sql 문 조회 후 결과가 0 건일 경우 dataset 의 컬럼정보를 구성하기 위하여 interceptor 를 이용하고 uiAdapter 에서 컬럼정보를 구성한다.

◆ **plugin Interceptor**

파일 형태	설명
NexacroMybatisMetaDataProvider	Executor Plugin – 쿼리를 실행하며, List 형태의 데이터 조회 시 조회된 결과가 0건일 경우 결과 메타데이터 정보를 조회 한다. resultType이 Map일 경우 추가적으로 메타데이터 조회를 위한 쿼리를 실행한다.
NexacroMybatisResultSetHandler	ResultSetHandler Plugin – Executor Plugin에서 메타데이터 조회를 위해 쿼리를 실행하였을 경우에만 동작하며, ResultSetMetaData 와 resultType에 존재하는 필드 정보를 확인하여 메타데이터 정보를 조회 한다.

- Mapper***.xml**

mapper.xml 파일은 mybatis 에서 사용하는 sql 문을 담고 있다. Java serviceImpl 파일에서 SQL 문을 참조할 때 mapper 테그의 namespace 와 쿼리문 테그(select)의 id 를 이용한다.

/resources/mybatis/mappers/*-mapper.xml 아래 위치한다.

ex) mapper.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="example.nexacro.uiadapter.mapper.BoardMapper">

<select id="select_datalist_map" resultType="java.util.Map">
    SELECT
        A.TITLE AS TITLE,
        A.REG_ID AS REG_ID,
        A.REG_DATE AS REG_DATE,
        A.POST_ID AS POST_ID,
        A.CONTENTS AS CONTENTS,
        A.COMMUNITY_ID AS COMMUNITY_ID,
        A.HIT_COUNT AS HIT_COUNT
    FROM TB_BOARD A
```

```
</select>
```

ex) serviceImpl.java

```
public List<Map<String,Object>> select_datalist_map(Map<String,String> search) {
    BoardMapper mapper = sqlSession.getMapper(BoardMapper.class);
    return mapper.select_datalist_map(search);
}
```

4.3.6 UiadapterWebMvcConfig.java

클래스에 @Configuration 어노테이션을 사용하며 Spring Container 에 등록할 Bean 을 설정한다. @Bean 어노테이션이 있는 함수는 Bean 객체를 Spring Container 에 리턴한다. @Value 어노테이션으로 Yml, Properties 를 참조하여 uiadapter 에서 사용할 변수를 설정한다.

uiadapter 의 주요 java 설정은 다음과 같다.

- SessionLocaleResolver

Session 에 담긴 Locale 정보를 통해 다국어를 처리한다

```
@Bean
public SessionLocaleResolver localeResolver() {
    SessionLocaleResolver resolver = new SessionLocaleResolver();
    return resolver;
}
```

- Message **처리**

Application 종료없이 실행 중에 변경한 message 를 반영한다. Property 설정을 통해 Reloading 주기를 설정하면 주기적으로 Message Reloading 을 수행한다.

```
@Bean
public LocalValidatorFactoryBean localValidatorFactoryBean() {
    LocalValidatorFactoryBean localValidatorFactoryBean = new
LocalValidatorFactoryBean();
    localValidatorFactoryBean.setValidationMessageSource(messageSource());
```

```
        return localValidatorFactoryBean;
}
@Bean
public ReloadableResourceBundleMessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
        messageSource.setBasename("classpath:/message/message-common");
        messageSource.setDefaultEncoding("UTF-8");
        messageSource.setCacheSeconds(10 * 60); // 리로드 시간
        return messageSource;
}
```

참조하는 message 파일명은 {basename}_언어코드_국가코드.properties 의 형식을 가진다.

{basename}은 프로젝트의 사용목적에 따라 임의로 지정할 수 있다. uiadapter 에서는 {basename}으로 message-common 을 사용한다.

파일명	설명
{basename}.properties	Default Message로 시스템의 언어 및 지역에 맞는 Property 파일이 존재하지 않을 경우 사용
{basename}_en.properties	시스템 언어 코드가 영어일 때 사용
{basename}_ko_KR.properties	시스템 언어 코드가 한글일 때 사용
{basename}_en_UK.properties	시스템 언어 코드가 영어일 때 영국(국가코드)을 위한 메시지

◆ message-common.propeties

```
error.common=오류가 발생했습니다.
error.minlength={0}은 {1}자 이상의 문자를 입력하셔야 합니다.
```

◆ message 사용 예시

```
log.debug("error.common: "+MessageUtils.getMessage("error.common"));
log.debug("error.minlength: "+MessageUtils.getMessage("error.minlength", new String[]
{"테스트글자", "2"}));
```

◆ **테스트글자**은 **2**자 이상의 문자를 입력하셔야 합니다.

- **파일 처리 모듈 Filter 에 등록**

Excel 파일 import 처리를 위해 "multipartResolver" 이름으로 파일 처리 모듈을 Filter 에 등록한다.

```
@Bean
public MultipartFilter multipartFilter() {
    MultipartFilter multipartFilter = new MultipartFilter();
    multipartFilter.setMultipartResolverBeanName("multipartResolver");
    return multipartFilter;
}
```

- **Excel 처리 servlet 등록**

excel 처리 servlet 은 x-api 에서 제공하는 GridExportImportServlet 클래스를 지정한다.

nexacro ui 에서 호출하는 맵핑 URL 은 "/XExportImport.do" 로 지정한다.

```
@Bean
public ServletRegistrationBean<GridExportImportServlet> gridExportImportServletBean() {
    ServletRegistrationBean<GridExportImportServlet> excelBean = new
ServletRegistrationBean<GridExportImportServlet>(new GridExportImportServlet(),
"/XExportImport.do");
    excelBean.setLoadOnStartup(1);
   MultipartConfigElement multipartConfigElement = new
MultipartConfigElement(System.getProperty("java.io.tmpdir"), 10000000, 10000000 * 2,
10000000 / 2);
   excelBean.setMultipartConfig(multipartConfigElement);
   return excelBean;
}
```

- RequestMappingHandlerAdapter **등록**

Nexacro 의 요청을 컨트롤러 함수와 맵핑 처리하는 클래스를 등록한다. Nexacro 는 고유한 데이터 포맷(xml, json, ssv)을 가지고 있어서 이를 처리할 수 있는 Handler 를 등록해야 한다. 내부적으로 nexacro 에서 request 로 보낸 데이터(dataset, variable)를 컨트롤러 함수의 매개변수와 매핑할 수 있도록 NexacroMethodArgumentResolver 를 지정하고 처리를 위임한다.

```
@Override
public RequestMappingHandlerAdapter getRequestMappingHandlerAdapter() {
    return new NexacroRequestMappingHandlerAdapter();
}
```

- NexacroMethodArgumentResolver

nexacro 에서 request 할 때 보낸 dataset, variable 을 컨트롤러 함수의 매개변수에 바인딩하고, 객체 변환을 수행한다. 이 처리과정 후에 컨트롤러 함수에서 매개변수를 사용하여 로직을 처리하게 된다.

```
@Override
public void addArgumentResolvers(List<HandlerMethodArgumentResolver> resolvers) {
    NexacroMethodArgumentResolver nexacroResolver = new
    NexacroMethodArgumentResolver();
    resolvers.add(nexacroResolver);
    WebMvcConfigurer.super.addArgumentResolvers(resolvers);
}
```

- ReturnValueHandler **등록**

nexacro 에서 요청에 대한 처리를 마치고 결과를 다시 response 할 수 있도록 ReturnValueHandler 를 지정한다.

```
@Override
public void addReturnValueHandlers(List<HandlerMethodReturnValueHandler> handlers) {
    NexacroHandlerMethodReturnValueHandler returnValueHandler = new
NexacroHandlerMethodReturnValueHandler();

    NexacroFileView nexacroFileView = new NexacroFileView();
    NexacroView     nexacroView     = new NexacroView();
    nexacroView.setDefaultContentType(PlatformType.CONTENT_TYPE_XML);
    nexacroView.setDefaultCharset("UTF-8");

    returnValueHandler.setView(nexacroView);
    returnValueHandler.setFileView(nexacroFileView);
    handlers.add(returnValueHandler);
```

```
        WebMvcConfigurer.super.addReturnValueHandlers(handlers);
}
```

ReturnValueHandler 는 다음과 같은 속성을 지정한다.

속성	설명
NexacroFileView	파일 다운로드를 처리할 객체
NexacroView	nexacro에 response할 view 객체
ContentType	response할 content-type은 3가지가 있다. ● PlatformType.***CONTENT_TYPE_XML*** ● PlatformType.***CONTENT_TYPE_JSON*** ● PlatformType.***CONTENT_TYPE_SSV***
Charset	response할 Charset을 지정한다. ● PlatformType.***DEFAULT_CHAR_SET*** // "UTF-8"

- ExceptionResolver **등록**

nexacro 의 요청에 대해 정상적으로 처리될 경우 res 코드는 200(OK)으로 전송되며, 내부에 Variable (ErrorCode, ErrorMsg)을 통해 처리 한다 nexacro request 처리중 발생하는 에러를 nexacro 포맷에 맞게 예외를 생성한다.

```
@Override
public void configureHandlerExceptionResolvers(List<HandlerExceptionResolver> resolvers)
{
        NexacroView nexacroView = new NexacroView();
        nexacroView.setDefaultContentType(PlatformType.CONTENT_TYPE_XML);
        nexacroView.setDefaultCharset("UTF-8");

        NexacroMappingExceptionResolver nexacroException = new
NexacroMappingExceptionResolver();

        nexacroException.setView(nexacroView);
        nexacroException.setShouldLogStackTrace(true);
        nexacroException.setShouldSendStackTrace(true);
        nexacroException.setDefaultErrorMsg("fail.common.msg");
        resolvers.add(nexacroException);
        WebMvcConfigurer.super.configureHandlerExceptionResolvers(resolvers);
}
```

nexacro 는 에러 유무에 따른 상태 값은 다음과 같다.

구분	ErrorCode	ErrorMsg
정상	0	공백
Exception 발생	-1	예외의 실제 메시지 혹은 'An Error Occured. check the ErrorCode for detail of error infomation'
Exception 사용자정의	-100 이하	사용자정의 Exception 메시지

NexacroMappingExceptionResolver 는 다음과 같은 속성을 지정한다

속성	설명
view	예외 발생시 사용되는 View
shouldLogStackTrace	에러정보를 로깅할지 여부 (true | false)
shouldSendStackTrace	에러정보를 응답으로 전송할지 여부 (true | false), 예외의 실제 메시지를 응답으로 전송할 경우 보안상의 문제가 있을 수 있다. 보안상의 이유로 운영시에는 false로 설정한다. 단, false로 설정 되어 있을 경우 예외의 message는 전송되지 않으며, errorMsg(에러메시지) 만 응답으로 전송된다.
defaultErrorMsg	shouldSendStackTrace 가 false 일 경우 nexacro platform으로 전송되는 에러메시지, 정의된 메시지 명칭 혹은 값을 입력한다.
messageSource	메시지를 처리하기 위한 org.springframework.context.MessageSource를 설정한다.

NexacroException 의 주요 함수는 다음과 같다.

함수	설명
NexacroException(String message)	입력받은 message를 통해 NexacroException을 생성한다.
NexacroException(String message, Throwable cause)	입력받은 message와 cause를 통해 NexacroException을 생성한다.
setErrorCode(int errorCode)	nexacro platform으로 전송 할 에러코드를 설정한다. 설정하지 않을 경우 '**-1**' 을 전송한다.
setErrorMsg(String errorMsg)	nexacro platform으로 전송 할 에러메세지를 설정한다. 설정하지 않을 경우 NexacroException 생성 시 전달 받은 메시지를 전송한다.

ex) NexacroException 사용 예시

```
if(info == null) {
    throw new NexacroException("exceptionName", -999, "error.session");
}
```

- **excel 처리 속성 등록**

nexacro 에서 excel 을 처리할 때 참조하는 속성을 등록한다.

```
@Bean
    public ServletContextInitializer initializer() {
        return new ServletContextInitializer() {

    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {
        servletContext.setInitParameter("export-path"          , "/excel");
        servletContext.setInitParameter("import-path"          , "/excel");
        servletContext.setInitParameter("monitor-enabled"     , "true");
        servletContext.setInitParameter("monitor-cycle-time", "30");
        servletContext.setInitParameter("file-storage-time" , "10 );
      };
  };
}
```

excel 관련 속성과 값은 다음과 같다.

속성	값	내용
export-path	/excel	엑셀 export 임시폴더 생성 기준 디렉터리
import-path	/excel	엑셀 import 임시폴더 생성 기준 디렉터리
monitor-enabled	true	임시파일 삭제를 위한 모니터링 여부
monitor-cycle-time	30	임시파일 삭제를 위한 모니터링 주기(default:분)
file-storage-time	10	임시파일 생성 디렉터리 모니터링 주기 (default:분)

- **Resource 경로 지정**

uiadapter 에서 사용하는 resource 경로를 지정한다.

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
                    registry.addResourceHandler("/**")
                                .addResourceLocations(contextRoot);
}
```

- **DBVendorsProvider 설정**

nexacro 의 request 처리 결과가 0 건일 때 컬럼 정보를 가져오기 위한 설정이다. 기본적인 컬럼 맵핑은 java 에서 처리해 주지만 DBMS 에 따라 컬럼 타입이 특이한 경우, 컬럼 타입 맵핑을 위하여 추가적인 처리가 필요할 수도 있다.

```
@Bean
public DbVendorsProvider dbmsProvider() {

    DbVendorsProvider dbmsProvider = new DbVendorsProvider();
    Map<String,Dbms> dbvendors = new HashMap<String,Dbms>();

    dbvendors.put("Oracle", new Oracle()); // key(DB 이름) 값은  아래 표 참조.
    dbmsProvider.setDbvendors(dbvendors);

    return dbmsProvider;
}
```

DB 별로 고유한 DB 이름을 key 로 DB 객체를 `DbVendorsProvider` 에 담는다.
DB 이름은 DatabaseMetaData 객체의 getDatabaseProductName() 함수 호출 결과 혹은 mybatis DatabaseIdProvider 의 key 값과 일치해야 한다.

각 DB 별 DB 이름과 사용 예시는 다음과 같다.

DB명	DB 이름	사용예시
Oracle	"Oracle"	dbvendors.put("Oracle", new Oracle());
MsSQL	"Microsoft SQL Server"	dbvendors.put("Microsoft SQL Server", new Mssql());
mysql	"MySQL"	dbvendors.put("MySQL", new Mysql());
mariadb	"MariaDB" or "MySQL"	dbvendors.put("MariaDB", new Mariadb());
postgreDB	"PostgreSQL"	dbvendors.put("PostgreSQL", new Postgre());

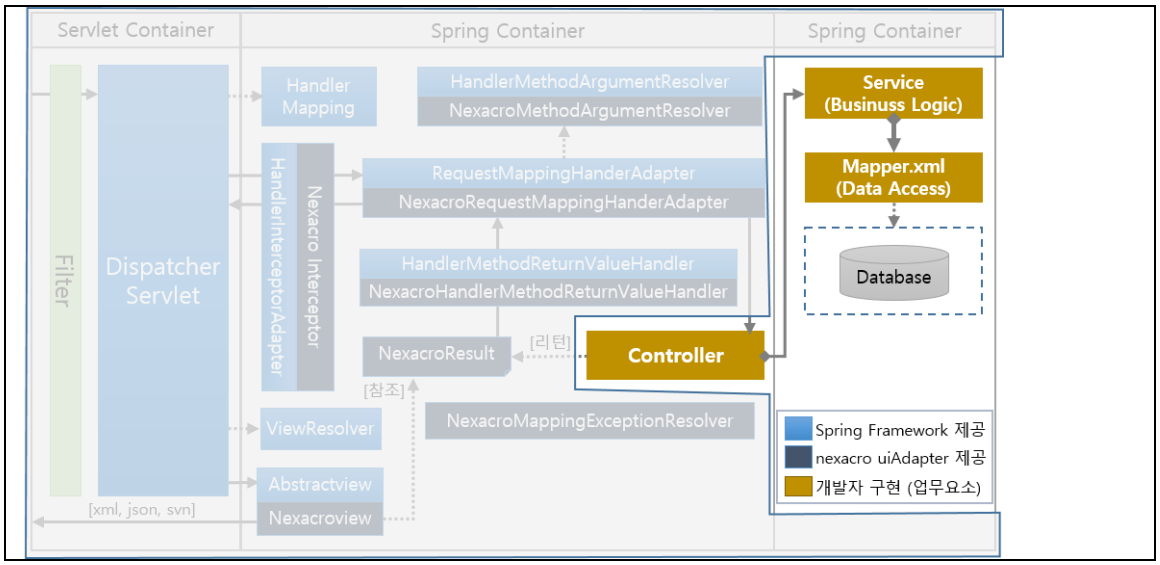
4.3.7 Nexacro_server_license.xml

nexacro 서버 라이선스 파일을 class-path 경로에 위치시킨다. 개발용 라이선스는 2 개월 단위로 기한이 만료된다. 개발용 라이선스는 고객지원사이트에서 신청한다.

- 고객지원 URL :
  http://support.tobesoft.co.kr/Support/?menu=License_Info&page=1

4.4 Nexacro transaction 처리 리뷰

uiadapter 는 spring boot 아키텍처 기반의 프레임워크이다. 게시판 업무에 대한 request 를 받고 처리하는 프레임워크 프로세스 중에 업무개발자의 구현(업무요소) 단계인 Controller, Service, Mapper.xml 의 코드를 리뷰한다.



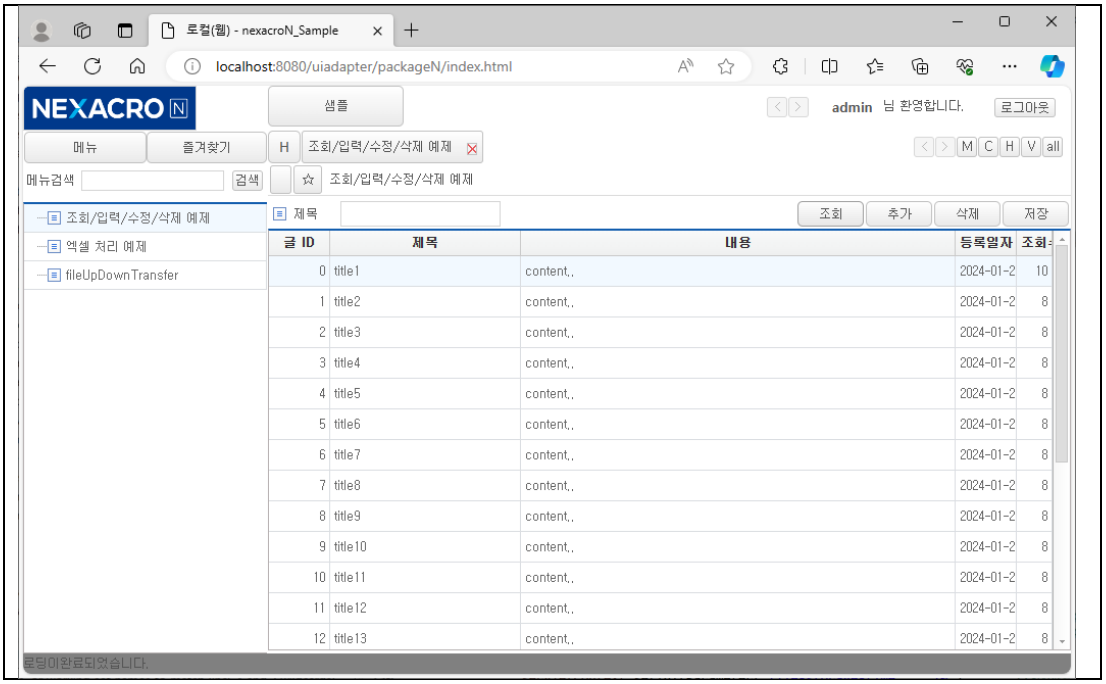
- nexacro ui 소스의 경로는 ./src/main/nxui/packageN 이다. ui 소스의 개발가이드는 개발도구가이드(https://docs.tobesoft.com/development_tools_guide_nexacro_n_v24_ko)를 참조한다.

4.4.1 게시판 조회 예제

구분	설명
화면명	게시판 목록 조회
nexacro 메뉴명	조회/입력/수정/삭제 예제
nexacro 파일명	pattern::pattern01.xfdl
Request URL	select_datalist_map.do

4.4.1.1 nexacro 화면 transaction

- 화면



- 소스

```
this.fnSearch = function ()
{
        var strSvcId      = "search";
        var strSvcUrl     = "select_datalist_map.do";
        var inData        = "dsSearch=dsSearch";
        var outData       = "dsList=output1";
        var strArg        = "";
        var callBackFnc = "fnCallback";
        var isAsync     = true;
        this.gfnTransaction(strSvcId, strSvcUrl, inData, outData, strArg, callBackFnc,
isAsync);
};
```

4.4.1.2 Controller

controller 위치는 [프로젝트 package].web 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 목록 조회
처리 단계	
Controller 파일명	example.nexacro.uiadapter.web.BoardController.java
@RequestMapping	/select_datalist_map.do
맵핑 메서드	NexacroResult select_datalist_map()

- 소스

```
@RequestMapping(value = "/select_datalist_map.do")
    public NexacroResult select_datalist_map(@ParamDataSet(name="dsSearch", required
= false) Map<String,String> searchMap) throws NexacroException{

    List<Map<String,Object>> sampleList = boardService.select_datalist_map(searchMap);
    NexacroResult result = new NexacroResult();
    result.addDataSet("output1", sampleList);

    return result;
    }
```

- 소스 단계별 주요 기능은 다음과 같다.

소스	설명
@RequestMapping	Repuest URL을 메서드에 맵핑
@ParamDataSet	nexacro 화면에서 Transaction 함수에서 보낸 3번째 아규먼트 중 좌변에 해당하는 dataset 이름 (name = "dsSearch")을 메서드 아규먼트 dsSearch와 맵핑된다.
List<Map<String, Object>> sampleList = boardService.select_datalist_map(searchMap);	사용자 정보 조회 Service 구현체를 호출하고 그 결과를 List 형태로 받는다.
result.addDataSet("output1", sampleList);	List 형태의 조회 결과를 NexacroResult 객체에 담는다.

4.4.1.3 Service

service interface 위치는 [프로젝트 package].service 경로를 default 로 한다.

service 구현체 위치는 [프로젝트 package].impl 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 목록 조회
처리 단계	Controller → Service (ServiceImpl) → Mapper Interface → Mapper.xml
Service 구현체 파일명	example.nexacro.uiadapter.impl.BoardServiceImpl.java
Service interface	BoardService.java
처리 메서드	List<Map<String, Object>> searchSampleList()

- 소스

```
@Override
public List<Map<String,Object>> select_datalist_map(Map<String,String>
search) {
    BoardMapper mapper = sqlSession.getMapper(BoardMapper.class);
   return mapper.select_datalist_map(search);
}
```

◆ 소스 단계별 주요 기능은 다음과 같다.

소스	설명
@Override	Interface에 정의된 함수를 구현해야하는 어노테이션
BoardMapper mapper = sqlSession.getMapper(BoardMapper.class);	BoardMapper interface 로 sqlSession의 Mapper 설정
return mapper.select_datalist_map(search);	mapper interface의 함수명을 id로 하는 query를 실행하고 그 결과를 받음.

4.4.1.4 Mapper interface

Mapper interface 위치는 [프로젝트 package].mapper 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 목록조회
처리 단계	Controller → Service (ServiceImpl) → Mapper Interface → Mapper.xml
Mapper Interface 파일명	example.nexacro.uiadapter.impl.BoardMapper.java
처리 메서드	public List<Map<String,Object>> select_datalist_map(Map<String,String> board);

● 소스

```
public List<Map<String,Object>> select_datalist_map(Map<String,String> board);
```

4.4.1.5 mapper.xml

mapper 쿼리 파일 위치는 src/main/resource/mybatis/mappers/ 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 목록 조회
처리 단계	Controller → Service (ServiceImpl) → Mapper Interface → Mapper.xml
Mapper 파일명	mybatis/mappers/board_mapper.xml
mapper namespace	"example.nexacro.uiadapter.mapper.BoardMapper"
쿼리 ID	<select id="select_datalist_map" ...

● 소스

```
<mapper namespace="example.nexacro.uiadapter.mapper.BoardMapper">

<select id="select_datalist_map" parameterType="java.util.Map" resultType="java.util.Map">
        ... <중략>
```

◆ 소스 단계별 주요 기능은 다음과 같다.

소스	설명
<mapper namespace=" example.nexacro.uiadapter.mapper.BoardMapper ">	Mapper Interface 경로명으로 한다. Mapper.xml의 고유한 명칭으로 중복되지 않아야 한다.
<select id="select_datalist_map "	쿼리문의 고유한 명칭으로 중복되지 않아야 한다.
resultType="java.util.Map"	쿼리문 수행결과 리턴 형태를 지정한다.

4.4.2 게시판 저장(등록, 수정, 삭제)

nexacro 에서 보내는 dataset 은 insert, update, delete 를 모두 보낼 수 있다.

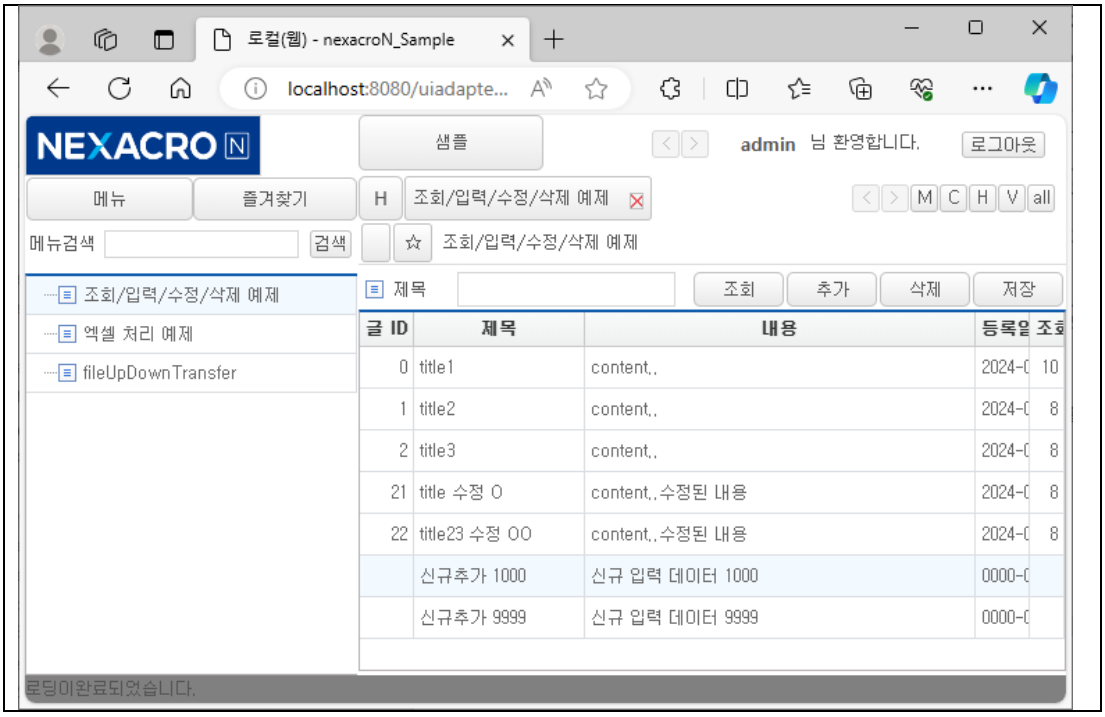
구분	설명

업무명	게시판 저장 (입력/수정/삭제)
nexacro 메뉴명	저장(입력/수정/삭제) 예제
nexacro 파일명	pattern::pattern01.xfdl
Request URL	update_datalist_map.do

4.4.2.1 nexacro 화면 transaction

- 화면

조회된 목록 화면에서 수정, 삭제, 추가 내용을 작성하고 저장한다.



- 소스

```
this.fnSave = function ()
{
    var strSvcId     = "save";
    var strSvcUrl    = "save_datalist_map.do";
    var inData       = "dataList=dsList:A";
    var outData      = "";

        this.gfnTransaction( strSvcId ,      // transaction을 구분하기 위한 svc id값
```

```
                        strSvcUrl ,     // trabsaction을 요청할 주소
                        inData ,        // 입력값으로 보낼 dataset id , a=b형태
                        outData );
};
```

4.4.2.2 Controller

controller 위치는 [프로젝트 package].web 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 저장
처리 단계	
Controller 파일명	example.nexacro.uiadapter.web.BoardController.java
@RequestMapping	/save_datalist_map.do
맵핑 메서드	NexacroResult save_datalist_map()

- 소스

```
@RequestMapping(value = "/save_datalist_map.do")
public NexacroResult save_datalist_map(@ParamDataSet(name = "dataList")
List<Map<String,Object>> dataList) throws NexacroException{

    boardService.save_datalist_map(dataList);
    NexacroResult result = new NexacroResult();
    return result;
}
```

- 소스 단계별 주요 기능은 다음과 같다.

소스	설명
@RequestMapping	Repuest URL을 메서드에 맵핑
@ParamDataSet	nexacro 화면에서 Transaction 함수에서 보낸 3번째 아규먼트(name = "dataList=dsList:A") 중 좌변에 해당하는 변수명 ("dataList")이 메서드 아규먼트 dataList와 맵핑된다.
boardService.save_datalist_map(dataList);	사용자 정보 조회 Service 구현체를 호출하고 그 결과를 List 형태로 받는다.
NexacroResult result = new	정상적으로 처리가 되면 신규로 NexacroResult 객

NexacroResult();	체를 생성하여 리턴한다.

4.4.2.3 Service

service interface 위치는 [프로젝트 package].service 경로를 default 로 한다.

service 구현체 위치는 [프로젝트 package].impl 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 저장
처리 단계	
Service 구현체 파일명	example.nexacro.uiadapter.impl.BoardServiceImpl.java
Service interface	BoardService.java
처리 메서드	List<Map<String, Object>> save_datalist_map()

- 소스

nexacro 에서 보내는 dataset 은 insert, update, delete 를 모두 보낼 수 있고 상태값에 따라 각각 처리한다. nexacro 화면 소스에서 상태값을 위한 별도의 작업이 필요하지 않다.

```
@Override
public void save_datalist_map(List<Map<String,Object>> dataList) {
      BoardMapper mapper = sqlSession.getMapper(BoardMapper.class);

     int size = dataList.size();
     for (int i=0; i<size; i++) {
          Map<String,Object> board = dataList.get(i);

          int rowType =
Integer.parseInt(String.valueOf(board.get(DataSetRowTypeAccessor.NAME)));
          if (rowType == DataSet.ROW_TYPE_INSERTED){
           mapper.insert_board_map(board);
          }else if (rowType == DataSet.ROW_TYPE_UPDATED){
           mapper.save_board_map(board);
          }else if (rowType == DataSet.ROW_TYPE_DELETED){
           mapper.delete_board_map(board);
          }
```

```
    }
}
```

◆ 소스 단계별 주요 기능은 다음과 같다.

소스	설명
rowType	Nexacro 에서 보낸 dataset의 row는 모두 각각의 상태값을 가지고 있다. 각각의 data 상태는 **3.2.3.3 DataSet ROW_TYPE** 을 참조한다.

mapper.insert_board_map(board);
mapper.save_board_map(board);
mapper.delete_board_map(board);
	* Data의 rowType에 따라 insert(), update(), delete()를 호출한다.

4.4.2.4 Mapper Interface

Mapper interface 위치는 [프로젝트 package].mapper 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 저장
처리 단계	Controller → Service (ServiceImpl) → Mapper Interface → Mapper.xml
Mapper Interface 파일명	example.nexacro.uiadapter.impl.BoardMapper.java
처리 메서드	**public void** insert_board_map(Map<String,Object> board); **public void** save_board_map(Map<String,Object> board); **public void** delete_board_map(Map<String,Object> board);

4.4.2.5 mapper.xml

mapper 쿼리 파일 위치는 src/main/resource/mybatis/mappers/ 경로를 default 로 한다.

구분	설명
UI 업무명	게시판 저장
처리 단계	Controller → Service (ServiceImpl) → Mapper Interface → Mapper.xml
Mapper 파일명	mybatis/mappers/board_mapper.xml
mapper namespace	"example.nexacro.uiadapter.mapper.BoardMapper"
쿼리 ID	<insert id="insert_board_map" parameterType="java.util.Map"> <update id="update_board_map" parameterType="java.util.Map">

	<delete id="delete_board_map" parameterType="java.util.Map">

- 소스

```
<insert id="insert_board_map" parameterType="java.util.Map">
    INSERT INTO TB_BOARD (
        TITLE,
        REG_ID,
        CONTENTS,
        COMMUNITY_ID
    ) VALUES (
        #{TITLE},
        #{REG_ID},
        #{CONTENTS},
        #{COMMUNITY_ID}
    )
</insert>

<update id="update_board_map" parameterType="java.util.Map">
    UPDATE TB_BOARD
    SET TITLE=#{TITLE}
        , REG_ID=#{REG_ID}
        , CONTENTS=#{CONTENTS}
        , COMMUNITY_ID=#{COMMUNITY_ID}
    WHERE POST_ID=#{POST_ID}
</update>

<delete id="delete_board_map" parameterType="java.util.Map">
    DELETE FROM TB_BOARD
    WHERE POST_ID=#{POST_ID}
</delete>
</mapper>
```

◆ 쿼리 id 별 주요 기능은 다음과 같다.

소스	설명
<mapper namespace="sample">	Mapper.xml의 고유한 명칭으로 중복되지 않아야

	한다.
<insert id="insert_board_map" parameterType="java.util.Map">	입력은 <insert 로 시작하고 id는 유니크한 값으로 중복되지 않아야 한다.
<update id="update_board_map" parameterType="java.util.Map">	수정은 <update 로 시작하고 id는 유니크한 값으로 중복되지 않아야 한다.
<delete id="delete_board_map" parameterType="java.util.Map">	삭제는 <delete 로 시작하고 id는 유니크한 값으로 중복되지 않아야 한다.

4.4.3 파일 업로드

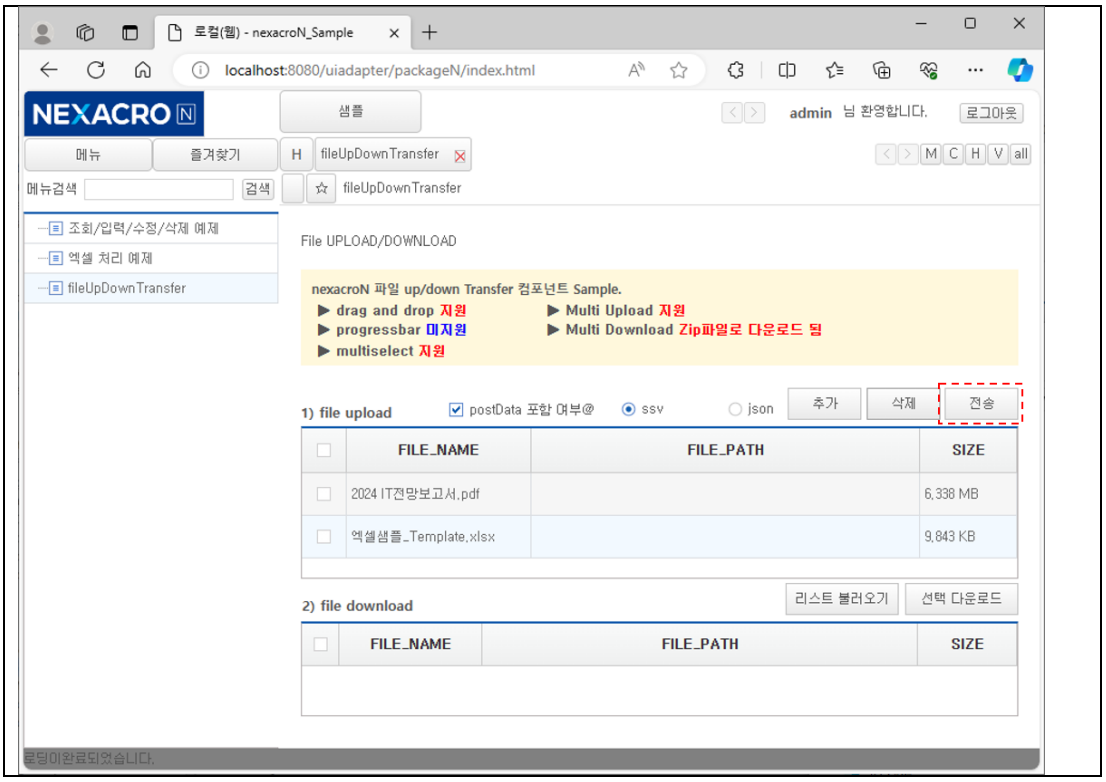
메뉴에서 fileUpDownTransfer 메뉴를 실행한다. 파일 데이터를 업로드 하는 경우 SpringBoot 의 MultipartHttpServletRequest 를 이용하여 처리 한다.

구분	설명
업무명	File Upload
nexacro 메뉴명	fileUpDownTransfer > 전송 버튼
nexacro 파일명	pattern/pattern03-FileUpDownloadTrans.xfdl
Request URL	advancedUploadFiles.do

4.4.3.1 nexacro 화면 transaction

- 화면

  파일 업로드 화면에서 업로드할 파일을 Drag&Drop 하거나 [추가]버튼으로 찾은 후 [전송]으로 업로드한다.



- 소스

```
this.fnUploadFile = function()
{
	this.setUploadInfo(this.ds_inputparam);

	var uploadUrl = this.serverUrl + "advancedUploadFiles.do?subFolder=" +
this.folderName;
	this.fileUpTrans.upload(uploadUrl);
}
```

4.4.3.2 Controller

controller 위치는 [프로젝트 package].web 경로를 default 로 한다.

SpringBoot 에서 Multipart 파일을 처리하기 위해서는 UiadapterWebMvcConfig.java 설정에서 Filter 를 등록해야 한다. 설정은 4.3.6 UiadapterWebMvcConfig.java 의 "파일처리 모듈 Filter 등록" 부분을 참조한다.

구분	설명

UI 업무명	File Upload
처리 단계	Controller
Controller 파일명	example.nexacro.uiadapter.web.FileController.java
@RequestMapping	/advancedUploadFiles.do
맵핑 메서드	NexacroResult uploadFiles()

- 소스

파일 업로드를 처리하는 함수는 다음과 같다. 파일 처리에 대한 간단한 셈플로 실제 업무에 맞게 정교하게 구현하여야 한다.

```
@RequestMapping(value = "/advancedUploadFiles.do" )
public NexacroResult uploadFiles(HttpServletRequest request, HttpServletResponse
response) throws Exception {

    if(!(request instanceof MultipartHttpServletRequest)) {
        if(logger.isDebugEnabled()) {
         logger.debug("Request is not a MultipartHttpServletRequest");
        }
        return new NexacroResult();
    }

     String characterEncoding = request.getCharacterEncoding();
    if(characterEncoding == null) {
        characterEncoding = PlatformType.DEFAULT_CHAR_SET;
    }
    String charsetOfRequest = CharsetUtil.getCharsetOfRequest(request,
characterEncoding);
    String queryString         = request.getQueryString();
    Map<String, String> queryMap = getQueryMap(queryString, charsetOfRequest);
    String filefolder = queryMap.get("filefolder");

    DataSet resultDs = createDataSet4UploadResult();

    MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;

    uploadParameters(multipartRequest);
    uploadMultipartFiles(multipartRequest, resultDs, filefolder);
```

```
        NexacroResult nexacroResult = new NexacroResult();
        nexacroResult.addDataSet(resultDs);

        return nexacroResult;
    }
```

◆ 주요 기능은 다음과 같다.

소스	설명
createDataSet4UploadResult()	처리결과를 리턴할 Dataset 생성
uploadParameters(multipartRequest)	파일업로드 외 추가로 처리할 정보가 있을 경우 처리할 수 있는 함수로 목적에 맞게 추가로 기능을 구현해야 함.
uploadMultipartFiles(multipartRequest, resultDs, filefolder);	파일업로드를 처리하는 함수를 호출하고 2번째 argument resultDs 에 처리결과를 기록함
nexacroResult.addDataSet(resultDs)	처리 결과를 nexacroResult 에 담음

4.4.4 파일 다운로드

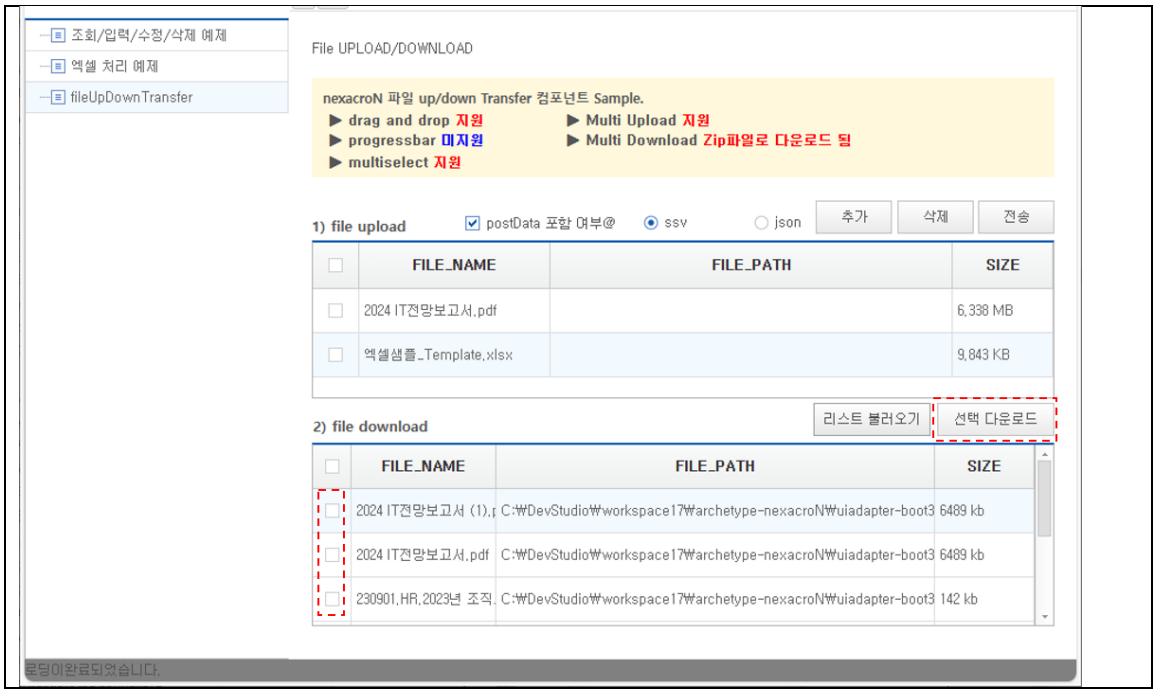
예제에서 파일 다운로드는 단건 다운로드와 다건 다운로드 예제를 포함하고 있다.

구분	설명
업무명	File Download
nexacro 메뉴명	파일 다운로드
nexacro 파일명	example.nexacro.uiadapter.web.FileController.java
단건 다운로드 URL	/advancedDownloadFile.do
다건 다운로드 URL	/multiDownloadFiles.do

4.4.4.1 nexacro 화면 transaction

- 화면

  파일리스트에서 첨부파일이 있을 경우 첨부파일 목록에 나타나고 원하는 파일을 체크박스로 선택한 후 [선택다운로드] 한다. 한 건을 체크하면 단건 다운로드 로직이 수행되고 하나 이상의 파일을 체크하면 다건 다운로드 로직이 수행된다.



- 소스

```
// 단건처리
this.fnDownloadFile = function(i)
{
	this.fileDownTrans.clearPostDataList();
	this.fileDownTrans.set_downloadfilename(this.dsDownload.getColumn(i, "FILE_NAME"));
	//runtime 전용 프로퍼티

	this.fileDownTrans.setPostData(    "filepath",this.dsDownload.getColumn(i, "FILE_URL") );
	this.fileDownTrans.setPostData( "filename", this.dsDownload.getColumn(i,
"FILE_NAME") );
	this.fileDownTrans.setPostData( "subFolder", "fileSample" );
	var queryStr = "&file="+this.dsDownload.getColumn(i, "FILE_NAME");
	var downloadUrl = this.serverUrl+"advancedDownloadFile.do?subFolder=" +
this.folderName + queryStr;
	this.fileDownTrans.download(downloadUrl);
}

// 다건처리
this.fnDownloadFileAll = function()
{
```

```
        this.fileDownTrans.clearPostDataList();
        var arrNameList = new Array();

        for(var i=0; i < this.dsDownload.getRowCount(); i++)
        {
            if(this.dsDownload.getColumn(i, "CHK") == 1) {
                arrNameList.push(this.dsDownload.getColumn(i, "FILE_NAME"));
            }
        }
        this.fileDownTrans.set_downloadfilename("fileSample.zip");      //runtime 전용 프로퍼
티
        this.fileDownTrans.setPostData( "filenamelist", arrNameList );

        this.fileDownTrans.download(this.serverUrl+"multiDownloadFiles.do?subFolder=" +
this.folderName);
}
```

4.4.4.2 Controller

controller 위치는 [프로젝트 package].controller 경로를 default 로 한다.

- Case 1 > 단건 처리

구분	설명
UI 업무명	첨부파일 다운로드
처리 단계	Controller 독립 처리
Controller 파일명	example.nexacro.uiadapter.web.FileController.java
@RequestMapping	/advancedDownloadFile.do
맵핑 메서드	NexacroFileResult downloadFile()

◆ 소스

단건 파일 다운로드 소스는 다음과 같다. 단건 파일 다운로드의 경우 return type 이 NexacroFileResult 이다.

```
@RequestMapping(value = "/advancedDownloadFile.do")
public NexacroFileResult downloadFile(HttpServletRequest request) throws Exception {
```

```
        String characterEncoding = request.getCharacterEncoding();
        if(characterEncoding == null) {
            characterEncoding = PlatformType.DEFAULT_CHAR_SET;
        }
        String charsetOfRequest = CharsetUtil.getCharsetOfRequest(request,
characterEncoding);
        String queryString          = request.getQueryString();
        Map<String, String> queryMap = getQueryMap(queryString, charsetOfRequest);

        String filefolder = queryMap.get("filefolder");

        String fileName = queryMap.get("file");
        if(fileName == null) {
            throw new NexacroException("No input fileName specified.");
        }

        fileName = URLDecoder.decode(fileName, charsetOfRequest);
        fileName = removedPathTraversal(fileName);

        String filePath ;
        if(filefolder == null) {
         filePath        = getFilePath();
        } else {
         filePath        = getFilePath() + SP + filefolder;
        }
        String realFileName = filePath + SP + fileName;
        File file = new File(realFileName);

        NexacroFileResult result = new NexacroFileResult(file);
        return result;
}
```

- Case 2 > 다건 처리

구분	설명
UI 업무명	첨부파일 다운로드

처리 단계	Controller 독립 처리
Controller 파일명	example.nexacro.uiadapter.web.FileController.java
@RequestMapping	/multiDownloadFiles.do
맵핑 메서드	NexacroFileResult multiDownloadFiles()

◆ 소스

다건 파일 다운로드 소스는 다음과 같다. 다건 파일 다운로드의 경우 return type 이 NexacroMultiFileResult 이다.

```
@RequestMapping(value = "/multiDownloadFiles.do")
public NexacroFileResult multiDownloadFiles(HttpServletRequest request,
HttpServletResponse response) throws Exception {

    String characterEncoding = request.getCharacterEncoding();
    if(characterEncoding == null) {
        characterEncoding = PlatformType.DEFAULT_CHAR_SET;
    }
    String charsetOfRequest = CharsetUtil.getCharsetOfRequest(request,
characterEncoding);
    String queryString        = request.getQueryString();
    Map<String, String> queryMap = getQueryMap(queryString, charsetOfRequest);

    String filePath ;

    String filefolder = queryMap.get("filefolder");
    if(filefolder==null) {
     filePath = getFilePath();
     filefolder = "download";
    } else {
     filePath = getFilePath() + SP + filefolder;
    }

    String filenamelist = request.getParameter("filenamelist");
    if(filenamelist == null) {
        throw new NexacroException("No input fileName specified.");
    }
```

```
        filenamelist = URLDecoder.decode(filenamelist, charsetOfRequest);
        filenamelist = removedPathTraversal(filenamelist);

        NexacroMultiFileResult result = new NexacroMultiFileResult(filefolder, filePath,
filenamelist);
        return result;
}
```