

# Anyframe MiP Query Plugin



Version 1.6.0

저작권 © 2007-2014 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

I. Introduction .....	1
II. MiPlatform Integration .....	2
1. MiPController .....	3
2. MiPQuery .....	5
2.1. MiPQueryService 활용 .....	5
2.1.1. MiPQueryService 속성 정의 파일 Sample .....	5
2.1.2. 매핑 XML 파일 샘플 .....	6
2.1.3. 테스트 코드 Sample .....	7
III. MiPlatform Simplification .....	10
3. MiPlatform Service .....	11
3.1. Controller .....	11
3.1.1. MiPController .....	11
3.2. Service .....	12
3.2.1. MiPService .....	12
3.2.2. MiPServiceImpl .....	13
3.3. MiPDao .....	14
3.4. Extension of MiPServiceImpl .....	14
3.4.1. [참고] MiPActionCommand .....	15
3.5. Testcase .....	15
IV. MiPlatform UI Sample .....	17
4. Architecture .....	18
4.1. Presentation Layer .....	18
4.2. Business Layer .....	19
5. Sample UI .....	20
5.1. Introduction .....	20
5.2. Set of Sample UI .....	20
5.2.1. 01GRD – 샘플1 (검색 + Grid + Paging Control) .....	20
5.2.2. 02GRDFRM – 샘플2 (검색 + Grid + 입력Form + Paging Control) .....	20
5.2.3. 03GRDPOP – 샘플3 (검색 + Grid + 입력Form 팝업 + Paging Control).....	21
5.2.4. 04GRDTAB – 샘플4 (검색 + 상단 Grid + 하단 Tab 입력 Form) .....	22
5.2.5. 05GRDTAB – 샘플5 (검색 + 좌측 Grid + 우측 Tab 입력Form) .....	22
5.2.6. 06GRDGRD – 샘플6 (좌/우Grid 간 항목 이동) .....	23
5.2.7. 07GRDGRD – 샘플7 (검색 + 상단 Master Grid + 하단 Sub Grid + Validation) .....	23
5.2.8. 08TRVGRD – 샘플8 (검색 + Tree + Grid + Tab 입력 Form 팝업 + Validation) .....	24
5.2.9. 09TRVFRM – 샘플9 (검색 + Tree + 입력 Form) .....	25
5.2.10. 10CTGGRD – 샘플10 (카테고리구분 + 검색 + Grid) .....	25
5.2.11. 11CALMTLY – 샘플11 (Grid를 이용한 월간 Calendar) .....	26
5.2.12. 12CALWKLY – 샘플12 (Grid를 이용한 주간 Calendar) .....	27
5.2.13. 13FILEATT - 샘플13 (파일 첨부) .....	27
6. Standards .....	29
6.1. Naming Rules of Form .....	29
6.2. Naming Rules of UI Component .....	29
6.3. Naming Rules of Variable .....	30
6.3.1. Global Variable .....	30
6.3.2. Common Script Variable .....	31
6.3.3. Local Variable .....	31
6.4. Naming Rules of Function .....	32
6.4.1. Global Function .....	32
6.4.2. Common Script Function .....	32
6.4.3. Local Function .....	32
7. Working with Common Flow .....	34
7.1. Common Script .....	34
7.1.1. Service Call .....	34

7.1.2. Callback .....	35
7.2. Common Dataset .....	35
7.2.1. Dataset for Service .....	35
7.3. Example .....	36
8. Validation .....	41
8.1. Using UI Component .....	41
8.1.1. Size Validation .....	41
8.1.2. Type Validation .....	41
8.2. Using Script .....	42
8.2.1. Check Validity .....	42
8.2.2. Check List for Validation .....	42
9. Internationalization (i18n) .....	45
9.1. Domain .....	45

---

# I.Introduction

MiPlatform은 X-internet 기반의 RIA 개발 플랫폼으로 국내에서 각광을 받고 있다. 이에 Anyframe에서는 MiPlatform을 기반으로 어플리케이션 개발 시에 MiPlatform 전용 데이터 타입 처리를 위해서 MiPQueryService와 MiPController를 제공하고 있다. mip-query plugin은 Anyframe에서 제공하는 MiPlatform 연계 모듈인 MiPQueryService와 MiPController의 기본적인 활용 방법을 가이드하기 위한 샘플 코드와 이를 위해 필요한 참조 라이브러리들로 구성되어 있다. Anyframe에서는 MiPlatform을 기반으로 어플리케이션 개발 시에 서버측 코딩을 전혀 하지 않고도 기본 CRUD 기능을 구현할 수 있도록 도와주는 공통 서버 모듈을 제공한다. mip-query plugin은 이러한 공통 서버 모듈을 활용하여 현장에서 사용 빈도가 높은 MiPlatform 기반의 UI를 유형별로 구현한 샘플 어플리케이션이 포함되어 있다.

## Installation

Command 창에서 다음과 같이 명령어를 입력하여 mip-query plugin을 설치한다.

```
mvn anyframe:install -Dname=mip-query
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

## Dependent Plugins

Plugin Name	Version Range
query-ria [ <a href="http://dev.anyframejava.org/docs/anyframe/plugin/optional/query-ria/1.6.0/reference/htmlsingle/query-ria.html">http://dev.anyframejava.org/docs/anyframe/plugin/optional/query-ria/1.6.0/reference/htmlsingle/query-ria.html</a> ]	2.0.0 > * > 1.4.0

## MySQL 사용 시 유의사항

본 샘플 어플리케이션은 ID 채번을 위해 Database의 Sequence/Function 기능을 이용 하고있다. 샘플 어플리케이션을 MySQL DB를 사용하는 환경에서 설치할 때, Function을 사용자가 MySQL Client 프로그램을 이용하여 직접 등록해야 한다. Function 생성 구문은 [프로젝트 폴더]/db/scripts/mip-query-insert-data-mysql.sql 파일에 작성 되어있다.

---

## II.MiPlatform Integration

Anyframe에서는 RIA 개발 플랫폼인 MiPlatform과 쉽고 편하게 연계할 수 있도록 MiPQueryService와 AbstractMiPController를 제공하고 있다.

---

# 1.MiPController

MiPlatform을 사용하여 개발 시, Client UI Component에서 조회/저장 이벤트가 발생하면 호출하는 Controller 클래스는 Business Service를 실행하여 결과값을 XML로 변환하여 전송한다. Anyframe은 개발자 편의를 위하여 AbstractMipController를 제공하며 개발자는 복잡한 변환로직을 신경쓰지 않고 개발이 가능하다.

```
public abstract class AbstractMipController extends AbstractController {
<!-- 중략 -->
    public ModelAndView handleRequestInternal(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        VariableList inVl = null;
        DatasetList inDl = null;
        VariableList outVl = null;
        DatasetList outDl = null;

        PlatformRequest platformRequest =
            new PlatformRequest(request, defaultCharset);
        PlatformResponse platformResponse =
            new PlatformResponse(response, defaultEncodeMethod, defaultCharset);

        try {
            platformRequest.receiveData();

            inVl = platformRequest.getVariableList();
            inDl = platformRequest.getDatasetList();
            outVl = new VariableList();
            outDl = new DatasetList();

            getLogger().debug(this.getClass().getName() + "." + "operate()" + " started");

            operate(request, inVl, inDl, outVl, outDl);
            if(logger.isDebugEnabled()){
                logger.debug("{} operate() ended", new Object[] { this.getClass()
                    .getName() });
                logger.debug("Output variableList");
                outVl.printVariables();
                logger.debug("Output DatasetList");
                outDl.printDatasets();
            }

            setResultMessage(outVl, 0, "Request has been processed successfully");

            String isFirstRow = inVl.getValueAsString("isFR");

            // in case of Firstrow
            if("y".equalsIgnoreCase(isFirstRow)){
                sendFirstrowData(response, inVl, outVl, outDl);
            } else {
                // general PlatformResponse
                sendPlatformData(response, outVl, outDl);
            }
        } catch (Exception e) {
            String msg = e.getMessage();

            if (msg == null)
                msg = "Fail to process client request.";

            logger.error(msg);
        }
    }
}
```

```

        setResultMessage(outVI, -1, msg);

        sendPlatformData(response, outVI, outDI);
    }

    logger.debug(this.getClass().getName() + " process() end!");
    return null;
}

    public abstract void operate(HttpServletRequest request, VariableList inVI,
                                DatasetList inDI, VariableList outVI, DatasetList outDI)
                                throws Exception;
<!-- 종략 -->
}

```

AbstractMiPController는 AbstractController의 handleRequestInternal 메소드를 구현하고 있고 operate 메소드를 호출한다. 개발자가 AbstractMiPController를 상속하여 User Defined Controller를 개발할 경우 실제 코딩은 operate 메소드 내부에 구현하면 된다.

- operate 메소드 내 이용변수 설명

변수 타입	변수명	설명
VariableList	inVI	Client에서 GET방식으로 전송한 parameter를 포함
VariableList	outVI	Client로 전송하는 VariableList
DatasetList	inDI	Client에서 POST방식으로 전송한 Dataset XML를 포함
DatasetList	outDI	Client로 전송하는 DatasetList

- Firstrow 방식 응답처리

MiPlatform에서는 서버에서 대용량 데이터를 클라이언트로 전송할 때 전체 데이터를 분할해서 전송하는 Firstrow 방식을 제공하고, 그를 위한 PlatformFRResponse 객체를 지원한다. 따라서, AbstractMiPController에서는 Dataset에 담긴 대량의 데이터를 사용자가 정의한 값(nextDataSize) 만큼의 Row 개수로 분할하여 응답 처리를 수행하도록 구현하고 있다. (\*대용량 데이터가 VariableList의 형태로 저장된 경우에는 Firstrow 방식을 사용할 수 없음을 유의하도록 한다.)

아래는 Firstrow 처리를 위해 클라이언트에서 전송해야 하는 파라미터에 대한 설명이다.

변수명	설명
isFR	Firstrow 방식으로 전송할지 여부 (Default : N)
nextDataSize	Firstrow 방식으로 전송할 때, DataSet row 분할 기준이 되는 DataSize값 (Default : 1000)

- Page navigation

MiPlatform이용 시 화면이동은 발생하지 않으며 조회/저장 이벤트에 해당하는 결과인 Dataset XML만 전송한다. 개발자가 User Defined Controller에서 operate 메소드 구현 시 화면 이동을 위한 View Name값은 null로 설정한다. 그리고 개발자가 AbstractMiPController를 상속하여 Controller를 개발할 때는 operate 메소드 내부에 Business Service를 실행하여 결과값을 반환하도록 구현한다.

## 2. MiPQuery

프리젠테이션 레이어 개발 시 X-Internet 제품인 MiPlatform 또는 Gauge 등을 기반으로 할 경우 각 제품은 사용자 입력 사항을 제품 고유의 데이터 형태에 저장하여 전달한다. 따라서 Query 서비스를 이용하여 DB 데이터를 처리하기 위해서는 "제품 고유의 데이터 전달 형태 <-> Map 또는 VO 간의 변환"을 위한 추가 작업이 필요하며, 이로 인해 대량의 데이터를 다루는 경우 성능 저하가 발생할 가능성이 크다. Query 서비스에서는 기본 QueryService를 확장하여, 특정 X-Internet 제품에 최적화된 형태의 구현체를 제공함으로써 개발 편의성과 응답 속도 향상을 도모하고자 한다. 다음에서는 MiPlatform에 최적화된 MiPQueryService 사용 방법에 대해서 살펴보도록 한다.

### 2.1. MiPQueryService 활용

MiPQueryService는 MiPlatform 고유의 데이터 전달 형태로부터 사용자가 입력한 데이터를 추출하여 해당 DB에 반영하는 역할을 수행한다. 그러므로 투비소프트사의 X-Internet 제품인 MiPlatform 기반으로 프리젠테이션 레이어를 개발하는 경우, MiPQueryService를 통해 MiPlatform 고유의 데이터 전달 형태인 `com.tobesoft.platform.data.Dataset`, `com.tobesoft.platform.data.VariableList`를 그대로 이용할 수 있게 된다.



#### Pagination시 유의 사항

PagingJdbcTemplate 속성 정의시에는 반드시 DBMS에 적합한 PagingSQLGenerator를 셋팅해 주어야 한다. 적절한 PagingSQLGenerator가 존재하지 않는 경우에는 QueryService에서 제공하는 `org.anyframe.query.impl.jdbc.generator.DefaultPagingSQLGenerator`를 사용할 수 있으나, DefaultPagingSQLGenerator를 이용하여 `findXXX()` 메소드를 실행하면 QueryService 내부적으로 조건에 해당하는 모든 데이터를 fetch한 이후 ResultSet Cursor의 위치를 이동시키는 방식으로 특정 페이지에 속한 데이터를 걸러낸다. 이 때 ResultSet Cursor를 움직이는 로직에서 상당한 시간이 소요되어 다량의 데이터 조회시 성능에 심각한 영향을 끼칠 수 있다. 따라서, DefaultPagingSQLGenerator 사용은 권장하지 않는다.

#### 2.1.1. MiPQueryService 속성 정의 파일 Sample

다음은 MiPQueryService를 정의한 `context-miplatform-query.xml` 파일의 일부이다. MiPQueryService는 내부적으로 RiaQueryService를 통해 데이터 접근 처리를 수행하므로 RiaQueryService에 대한 참조 관계 설정이 필요하다.

```
<bean id="mipQueryService" class="org.anyframe.mip.query.impl.MiPQueryServiceImpl">
  <property name="namedParamJdbcTemplate" ref="pagingNamedParamJdbcTemplate"/>
  <property name="lobHandler" ref="lobHandler"/>
  <property name="sqlRepository" ref="sqlLoader"/>
  <lookup-method name="getRowCallbackHandler" bean="rowCallbackHandler"/>
  <lookup-method name="getCallableStatementCallbackHandler"
    bean="callableStatementCallbackHandler"/>
  <lookup-method name="getPrintWriterRowCallbackHandler"
    bean="printWriterRowCallbackHandler"/>
</bean>

<bean id="callableStatementCallbackHandler"
  class="org.anyframe.mip.query.impl.jdbc.mapper.MiCallableStatementCallbackHandler"
  scope="prototype"/>

<bean id="rowCallbackHandler"
  class="org.anyframe.mip.query.impl.jdbc.mapper.MiDataSetCallbackHandler"
  scope="prototype"/>
```



```

<bean id="printwriterRowCallbackHandler"
  class="org.anyframe.mip.query.impl.jdbc.mapper.MiPPrintwriterCallbackHandler"
  scope="prototype"/>

<!-- The original JdbcTemplate definition -->
<bean id="pagingNamedParamJdbcTemplate"
  class="org.anyframe.query.impl.jdbc.PagingNamedParamJdbcTemplate">
  <constructor-arg index="0" ref="pagingJdbcTemplate"/>
  <constructor-arg index="1" ref="dataSource"/>
</bean>

<bean id="pagingJdbcTemplate" class="org.anyframe.query.impl.jdbc.PagingJdbcTemplate">
  <constructor-arg ref="dataSource"/>
  <property name="exceptionTranslator" ref="exceptionTranslator"/>
  <property name="paginationSQLGetter" ref="paginationSQLGenerator"/>
</bean>

<bean id="paginationSQLGenerator"
  class="org.anyframe.query.impl.jdbc.generator.OraclePaginationSQLGenerator"/>

```

## 2.1.2.매핑 XML 파일 샘플

다음은 앞서 정의한 MiPQueryService가 참조하는 RiaQueryService Bean을 통해 로드된 mapping-mip-query-movie.xml 파일의 일부로, Named Parameter를 이용한 쿼리문들을 포함하고 있다.

```

<query id="createMovie" isDynamic="true" mappingStyle="upper">
  <statement>
    INSERT INTO
      MOVIE (MOVIE_ID, TITLE, DIRECTOR, GENRE_ID, ACTORS, RUNTIME, RELEASE_DATE, TICKET_PRICE,
      NOW_PLAYING, POSTER_FILE)
    VALUES
      (:MOVIE_ID, :TITLE, :DIRECTOR, :GENRE_ID, :ACTORS, :RUNTIME, :RELEASE_DATE, :TICKET_PRICE, :NOW_PLAYING, :POSTER_FILE)
  </statement>
</query>

<query id="updateMovie" isDynamic="true" mappingStyle="upper">
  <statement>
    UPDATE MOVIE
    SET
      TITLE = :TITLE,
      DIRECTOR = :DIRECTOR,
      GENRE_ID = :GENRE_ID,
      ACTORS = :ACTORS,
      RUNTIME = :RUNTIME,
      RELEASE_DATE = :RELEASE_DATE,
      TICKET_PRICE = :TICKET_PRICE,
      NOW_PLAYING = :NOW_PLAYING
    WHERE
      MOVIE_ID = :MOVIE_ID
  </statement>
</query>

<query id="removeMovie" isDynamic="true" mappingStyle="upper">
  <statement>
    DELETE FROM MOVIE
    WHERE
      MOVIE_ID = :MOVIE_ID
  </statement>
</query>

```

```

<query id="findMovieByPk" isDynamic="true" mappingStyle="upper">
  <statement>
    SELECT
      MOVIE_ID, TITLE, DIRECTOR, GENRE_ID, ACTORS, RUNTIME, RELEASE_DATE, TICKET_PRICE,
      NOW_PLAYING, POSTER_FILE
    FROM MOVIE
    WHERE
      MOVIE_ID = :MOVIE_ID
  </statement>
</query>

<query id="findMovieList" isDynamic="true" mappingStyle="upper">
  <statement>
    SELECT
      MOVIE_ID, TITLE, DIRECTOR, GENRE_ID, ACTORS, RUNTIME, RELEASE_DATE, TICKET_PRICE,
      NOW_PLAYING, POSTER_FILE
    FROM MOVIE
    WHERE
      TITLE like '%' || :SEARCH_TITLE || '%'
      AND NOW_PLAYING = :SEARCH_NOW_PLAYING
    ORDER BY RELEASE_DATE DESC
  </statement>
</query>

```

## 2.1.3.테스트 코드 Sample

다음에서는 MiPQueryService를 이용하여 앞서 언급한 매핑 XML 파일에 정의된 INSERT, SELECT, UPDATE, DELETE 쿼리문을 실행하는 테스트 코드의 일부이다.

```

/**
 * Test code that enters new data in DB through Query service for MiPlatform
 */
public void insertDataSet() {
    MiPQueryService mipQueryService = (MiPQueryService) context.getBean("mipQueryService");

    Map<String, String> queryMap = new HashMap<String, String>();
    queryMap.put(MiPQueryService.QUERY_INSERT, "createProduct");

    // Transfer Dataset that should be reflected in map and DB which defines query id that
    // will be used by INSERT, UPDATE, DELETE type.
    // A large amount of data can be saved in Dataset.
    int resultInsert =
        mipQueryService.update(queryMap, makeProduct());
}

/**
 * Test code that modifies data in DB through Query service for MiPlatform
 */
public void updateDataSet() {
    MiPQueryService mipQueryService =
        (MiPQueryService) context.getBean("mipQueryService");

    Map<String, String> queryMap = new HashMap<String, String>();
    queryMap.put(MiPQueryService.QUERY_UPDATE, "updateProduct");

    // Transfer Dataset that should be reflected in map and DB which defines query id that
    // will be used by INSERT, UPDATE, DELETE type.
    // A large amount of data can be saved in Dataset.
    int resultUpdate =
        mipQueryService.update(queryMap, makeProduct());
}

```

```

}

/**
 * Test code for INSERT, UPDATE, DELETE of data in DB through query service for MiPlatform
 */
public void processAllDataSet() {
    MiPQueryService mipQueryService =
        (MiPQueryService) context.getBean("mipQueryService");

    Map<String, String> queryMap = new HashMap<String, String>();
    queryMap.put(MiPQueryService.QUERY_UPDATE, "updateProduct");
    queryMap.put(MiPQueryService.QUERY_INSERT, "createProduct");
    queryMap.put(MiPQueryService.QUERY_DELETE, "removeProduct");

    // Transfer Dataset that should be reflected in map and DB which defines query id that
    // will be used by INSERT, UPDATE, DELETE type.
    // A large amount of data can be saved in Dataset.
    int resultUpdate = mipQueryService.update(queryMap, makeProducts());
}

/**
 * Test code for INSERT, UPDATE, DELETE of data in DB through query service for MiPlatform
 * In this case, ActionCommand pre-post processing is executed.
 */
public void processAllDataSetWithActionCommand() {
    MiPQueryService mipQueryService =
        (MiPQueryService) context.getBean("mipQueryService");

    Map<String, String> queryMap = new HashMap<String, String>();
    queryMap.put(MiPQueryService.QUERY_UPDATE, "updateProduct");
    queryMap.put(MiPQueryService.QUERY_INSERT, "createProduct");
    queryMap.put(MiPQueryService.QUERY_DELETE, "removeProduct");

    // Transfer ActionCommand which defines process that should be done before and after
    // accessing Dataset and DB
    // that should be reflected in Map, DB which defines query id used according to INSERT,
    // UPDATE, DELETE type.
    // A large amount of data can be saved in Dataset.
    int resultUpdate =
        mipQueryService.update(queryMap, makeProducts(),
            new ProductActionCommand());
}

/**
 * Test code for searching data in DB through query service for MiPlatform
 * In this case, search conditions are in VariableList form.
 */
public void findDataSetWithVariant() {
    MiPQueryService mipQueryService =
        (MiPQueryService) context.getBean("mipQueryService");

    // Transfer query id to be executed, and search conditions in VariableList form
    Dataset resultDataSet =
        mipQueryService.search("findProductByPk", makeVariantList());
}

/**
 * Test code to search data in DB through query service for MiPlatform
 * In this case, search conditions are in Dataset form.
 */
public void findListDataSet(int expected) {
    MiPQueryService mipQueryService =

```

```
(MiPQueryService) context.getBean("mipQueryService");

// Transfer query id to be executed and search conditions in Dataset form.
Dataset resultDataSet =
    mipQueryService.search("findProductList",
        makeSearchCondition());
}
```

위 소스 코드 중 `testProcessAllDataSetWithActionCommand()` 메소드에서는 `ActionCommand`를 이용하여 DB에 데이터를 입력하기 전에 특정 칼럼의 값을 변경하고 있다. 이와 같이 `MiPQueryService`는 `org.anyframe.mip.query.MiPActionCommand`를 구현한 별도 `ActionCommand`를 인자로 함께 전달하는 경우 입력 데이터를 DB에 반영하기 전/후에 대한 공통 처리를 수행할 수 있도록 지원한다. 예를 들어 입력받은 개별 Row를 DB에 신규 등록하기 전에 신규 식별자 값이 셋팅되어야 한다면, Loop을 돌면서 각 Row를 추출한 뒤 식별자를 셋팅하는 별도 로직없이 `preInsert()` 로직 내에 식별자 생성 구문이 추가된 `ActionCommand` 객체만 전달하면 되는 것이다.

---

## III. MiPlatform Simplification

MiPlatform에서는 데이터를 전달하기 위해서 MiPlatform만의 전용 데이터 타입을 사용한다. 주로 많이 사용되는 타입이 Dataset 객체이다. Dataset이라는 전용 데이터 타입을 사용하기 때문에 단순 CRUD 기능에 대해서는 Java의 Generics 처럼 서버 로직을 공통화하여 사용할 수 있다. 그래서 Anyframe 에서는 MiPlatform을 기반으로 어플리케이션을 개발 시, 복잡하고 특수한 업무로직이 없는 CRUD 기능 구현에 대해서는 개발자들이 Service나 DAO를 전혀 개발하지 않아도 되도록 공통 서버 모듈을 제공한다.

---

## 3.MiPlatform Service

어플리케이션의 UI를 MiPlatform을 사용해 개발 할 경우, MiPlatform 고유의 데이터 형태를 DB에 반영하기에는 많은 어려움이 있다.

예를 들어 Dataset에 10개의 컬럼과 10개의 Insert할 Record가 있을 경우 개발자가 일반적인 JDBC코딩을 하기 위해서는 Dataset의 10개의 컬럼 값을 일일이 꺼내야 하고 10번의 루프를 돌면서 Insert문을 실행하는 로직을 작성해야 한다.

또 DB에서 조회를 하고자 할 경우에는 ResultSet의 메타 정보를 이용해 Dataset의 컬럼을 셋팅하고 루프를 돌면서 ResultSet의 값들을 Dataset에 추가하는 로직을 작성해야 한다.

**Anyframe Ria MiP는 MiPlatform의 고유 데이터 형태를 사용해 DB에 CRUD하기 위한 공통 비즈니스 서비스와 Controller를 제공한다.**

Anyframe Ria MiPlatform의 장점은 아래와 같다.

- Dataset, VariableList와 같은 MiPlatform 고유의 데이터 형태를 변환하지 않고 비즈니스 서비스 개발을 할 수 있다.
- 추가적인 비즈니스 로직이 필요 없는 CRUD에 대해서는 비즈니스 서비스 개발 없이 Query Mapping File에 필요한 Query만 작성하면 된다.
- 확장이 필요한 부분만 오버라이드 해서 사용할 수 있기 때문에 비즈니스 서비스 개발이 쉽다.
- 기능이 중복되거나 불필요한 클래스를 생성하지 않기 때문에 전체 클래스 수가 줄어 들고 유지보수 또한 용이하다.

Anyframe Ria MiPlatform은 크게 Controller, Service, Dao로 구성되어 있다.

- **Controller** – MiPController : AbstractMiPController의 operate()를 확장한 클래스로 사용자 요청에 따라 비즈니스 서비스의 메소드를 호출하고 결과값을 화면으로 전달한다.
- **Service 인터페이스** – MiPService : DatasetList, VariableList를 이용해 DB에서 데이터를 조회, 추가, 삭제, 수정 등을 할 수 있는 API를 제공한다.
- **Service 구현 클래스** – MiPServiceImpl : MiPService의 구현 클래스로 Dataset과 실행 하고자 하는 Query Id를 짝지은 후 MiPDao의 메소드를 호출하고 Query실행 결과를 DatasetList에 추가한다.
- **Dao 클래스** – MiPDao : 파라미터의 형태에 따라 적절한 MiPQueryService의 메소드를 호출해 쿼리를 실행한다.

### 3.1.Controller

Anyframe에서는 MiPlatform 기반의 UI를 통한 사용자 요청을 처리할 수 있도록 Spring MVC의 AbstractController를 구현한 AbstractMiPController, AbstractMiPDispatchController를 제공하고 있다. 이 두 클래스를 상속받아 Controller 클래스를 구현하려면 사용자의 요청 별로 Controller 클래스를 구현해야 하므로 개발해야 할 Controller 클래스 수가 많아지고 유지보수 또한 어려워지는 단점이 있다. 이러한 단점을 보완하기 위해서 Anyframe Ria MiPlatform에서는 MiPController를 제공한다.

#### 3.1.1.MiPController

JSP 기반의 UI일 경우, 사용자 요청에 따라 Controller가 호출되고, Controller에서는 비즈니스 서비스 호출 결과 값을 결과 페이지에 전달하는 로직이 필요하다. 그러나 MiPlatform 기반의 UI에서는 화면과 서버간의 주고받는 데이터의 유형(DatasetList, VariableList)이 동일하고, 요청 화면과 결과 화면이 같으므로 공통화 처리가 가능해진다. 따라서, 비즈니스 서비스 호출 외에 별도 로직이 없을 때는 MiPController를 공통 Controller로 사용할 수 있다.

아래는 MiPController의 operate()의 일부로, 화면에서 전달받은 비즈니스 서비스의 Bean Id 를 이용해 WebApplicationContext에서 비즈니스 서비스 객체를 얻어온다. 실행할 비즈니스 서비스의 Bean Id와 메소드 이름은 dsService의 SERVICE(예: boardService.getPagingList)의 값에 의해 결정된다.

```
public class MiPController extends AbstractMiPController {

    public void operate(PlatformRequest platformRequest, VariableList inVl,
        DatasetList inDl, VariableList outVl, DatasetList outDl)
        throws Exception {

        String serviceName = inVl.getValueAsString("service");

        Object bean = getWebApplicationContext().getBean(serviceName);

        Method method = getMethod(bean, inVl.getValueAsString("method"));

        try {
            method.invoke(bean, new Object[] { inVl, inDl, outVl, outDl });
        } catch (Exception e) {
            Throwable te = e.getCause();
            logger.error("Can not invoke a dispatch method name", te);
            throw new Exception("Fail to process client request.", te);
        }
    }
}
..중략
```

만약 아래 그림의 설정처럼 SERVICE의 값이 없을 경우에는 비즈니스 서비스의 Bean Id는 mipService이고 메소드 이름은 dsService의 SVC\_ID값의 prefix로 결정된다.

No	SVC_ID	QUERY_LIST	SERVICE
1	saveAllBoard	querySet1=createBoard,updateBoard,removeBoard	
2	getPagingListBoard	querySet1=findBoardList	
3	getListCommunity	querySet1=findCommunityList	

prefix로는 get, getList, getPagingList, create, update, remove, saveAll이 올 수 있다. prefix가 getList일 경우에는 MiPService의 getList()가 실행된다.

## 3.2.Service

Anyframe Ria MiPlatform의 Service는 Interface인 MiPService와 구현 클래스인 MiPServiceImpl로 구성 되어 있다.

### 3.2.1.MiPService

MiPService는 MiPlatform의 고유 데이터 형태인 VariableList와 Dataset을 이용하여 외부에 제공할 수 있는 일반적인 기능을 정의하고 있는 인터페이스 클래스이다. 아래는 MiPService 소스 코드의 일부로 모든 메소드의 입력 파라미터는 (VariableList inVl, DatasetList inDl, VariableList outVl, DatasetList outDl)이며, Return Type는 void이다.

```
public interface MiPService {
    ..중략
    //리스트 조회
    void getList(VariableList inVl, DatasetList inDl,
        VariableList outVl, DatasetList outDl) throws Exception;
    //리스트 조회(페이징 처리)
    void getPagingList(VariableList inVl, DatasetList inDl,
        VariableList outVl, DatasetList outDl) throws Exception;
    //추가
    void create(VariableList inVl, DatasetList inDl,
```

```

        VariableList outVl, DatasetList outDl) throws Exception;
    ..중략
}

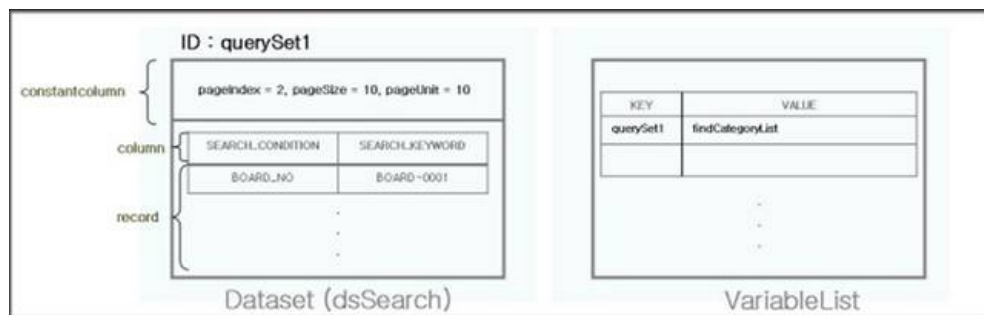
```

## 3.2.2.MiPServiceImpl

MiPServiceImpl은 MiPService의 구현 클래스로써 dsService에 설정된 정보를 기반으로 MiPDao의 메소드를 호출한다.

No	SVC_ID	QUERY_LIST	SERVICE
1	saveAllBoard	querySet1=createBoard,updateBoard,removeBoard	
2	getPagingListBoard	querySet1=findBoardList	
3	getListCommunity	querySet1=findCommunityList	

위의 2번 Row와 같이 dsService를 설정 했다면 VariableList에 아래 그림처럼 값이 셋팅되어 서버 측에 전달된다.



아래는 MiPServiceImpl의 getPagingList()의 일부로써, 특정 페이지에 속한 데이터를 조회하는 기능을 제공한다.

getPagingList()에서는 위의 그림에서와 같이 입력 파라미터로 전달된 DatasetList에 ID가 "querySet1"인 Dataset이 포함되어 있는 경우, VariableList로부터 "querySet1"이라는 KEY에 해당하는 값을 변수 queryId에 할당한다. 또한 해당 DatasetList로부터 ID가 "querySet1"인 Dataset을 추출하여 inDs라는 변수에 할당한다. 그 후 queryId와 inDs를 이용해 MiPDao의 메소드를 호출한다.

```

public void getPagingList(VariableList inVl, DatasetList inDl,
        VariableList outVl, DatasetList outDl) throws QueryException {

    int querySetCount = getQuerySetCount(inVl, outVl);
    String queryId = null;
    Dataset inDs = null;
    Dataset outDs = null;
    for( int i = 1 ; i <= querySetCount ; i++) {
        queryId = inVl.getValueAsString("querySet"+i);
        inDs = inDl.get("querySet"+i);
        try{
            if(inDs != null) {
                outDs = mipDao.getPagingList(queryId, inDs);
            }
            outDl.addDataset("querySet"+i, outDs);
        }
    }
}

```

MiPServiceImpl의 다른 메소드들도 이와 같이 입력 파라미터로부터 추출한 Dataset과 Query ID를 이용하여 사용자의 요청을 처리한다.

아래 그림처럼 dsService SVC\_ID의 prefix가 get, getList, getPagingList, create, update, remove일 경우에는 querySet에 한 개의 Query Id가 셋팅되어야 하고



No	SVC_ID	QUERY_LIST	SERVICE
1	saveAllBoard	querySet1=createBoard,updateBoard,removeBoard	
2	getPagingListBoard	querySet1=findBoardList	
3	getListCommunity	querySet1=findCommunityList	

saveAll이면 QUERY\_LIST에 “querySet1=createBoard,updateBoard,removeBoard”와 같이 추가, 수정, 삭제를 위한 3개의 Query Id가 셋팅되어 있어야 한다. 조회(get, getList, getPagingList)의 경우에는 결과 Dataset의 Id는 조회 시 검색 조건으로 사용했던 Dataset의 ID(“querySet+번호”)이다.

### 3.3.MiPDao

MiPDao는 MiPQueryService를 이용해 Query를 실행한다.

아래는 MiPDao는 Dataset의 Record를 DB Table에 저장(추가, 수정, 삭제)하는 saveAll()이다.

```
public int saveAll(Map queryMap, Dataset inDs,
    MiPActionCommand actionCommand) throws QueryException {

    if(actionCommand == null){
        return miPQueryService.update(queryMap, inDs);
    }else{
        return miPQueryService.update(queryMap, inDs, actionCommand);
    }
}
```

insert, update, delete를 위한 Map형태의 Query Id와 Dataset을 이용해 MiPQueryService의 update()를 호출하고 있음을 알 수 있다.

MiPDao는 Anyframe Ria MiPlatform에서 제공한 구현체를 사용할 것을 추천하며, 꼭 필요한 경우에 한해 확장해서 사용한다.

### 3.4.Extension of MiPServiceImpl

MiPService에서 제공하는 기능 외에 추가적인 기능이 필요할 경우에는 API를 추가로 정의하거나 해당 메소드를 오버라이드 할 수 있다.

아래는 Dataset의 Record를 DB에 Insert하기 전 'PROD\_NO' 컬럼에 유일한 아이디를 셋팅하기 위해 saveAll()을 오버라이드 해 기능을 확장한 예이다.

```
@Service("mipQueryMovieService")
@Transactional(rollbackFor = { Exception.class }, propagation = Propagation.REQUIRED)
public class MovieServiceImpl extends MiPServiceImpl implements MovieService {

    @Inject
    public MovieServiceImpl(MiPDao mipDao){
        super.mipDao = mipDao;
    }

    ..중략
    public void saveAll(VariableList inVl, DatasetList inDl,
        VariableList outVl, DatasetList outDl) throws Exception {

        Map<String, String> sqlMap = new HashMap<String, String>();
        sqlMap.put(MiPQueryServiceImpl.QUERY_INSERT, "createMovie");
        sqlMap.put(MiPQueryServiceImpl.QUERY_UPDATE, "updateMovie");
        sqlMap.put(MiPQueryServiceImpl.QUERY_DELETE, "removeMovie");

        mipDao.saveAll(sqlMap, inDl.get("dsSave"), new MovieActionCommand());
    }
}
```

```

    }

}

```

MiPDao의 saveAll()의 파라미터에 Anyframe에서 제공하는 MiPActionCommand를 구현한 MovieActionCommand를 전달하고 있다. MiPActionCommand를 활용하면 특정 쿼리문을 수행하기 전/후에 필요한 비즈니스 로직을 수행 할 수 있다.

### 3.4.1.[참고] MiPActionCommand

MiPActionCommand는 MiPQueryService의 save() 가 호출 됐을 때 Insert, Update, Delete Query를 실행 하기 전, 후 필요한 비즈니스 로직을 추가 할 수 있도록 하기 위해 제공되는 인터페이스이다. MiPActionCommand 인터페이스를 구현한 별도의 클래스를 정의하고 해당 메소드에 비즈니스 로직을 추가하면 된다.

아래는 앞서 언급한 MovieActionCommand 클래스의 preInsert()의 일부로써, Dataset을 특정 Table에 Insert하기 전에 Primary Key에 해당하는 PROD\_NO 컬럼에 유일한 값을 셋팅하고 있음을 알 수 있다.

```

public class MovieActionCommand implements MiPActionCommand{

    public void preInsert(Dataset ds, int index) {
        String id = "MV-" + System.currentTimeMillis();
        Variant variant = new Variant();
        variant.setObject(id);
        ds.setColumn(index, "MOVIE_ID", variant);
    }

}

```

따라서, MiPQueryService의save()에서는 Dataset의 Status가 'insert'인 Record를 DB에 Insert하기 전 MovieActionCommand의 preInsert()를 호출해 추가 로직을 실행한다.

## 3.5.Testcase

다음은 MiPService의 기능을 테스트 하는 Main.java 중 Dataset을 이용한 조회 기능을 테스트 하는 코드의 일부이다. querySet1, querySet2라는 Id를 가진 두 개의 Dataset의 "SEARCH\_CONDITION", "SEARCH\_KEYWORD" 컬럼에 검색 조건과 검색 문자를 입력한 후 , MiPService의 getList()를 호출 해 Query Id가 findBoardList인 query를 실행해 정상적으로 동작하는지 확인하는 테스트케이스이다.

```

/**
 * Dataset에 검색 조건이 세팅 되어 있을 때
 * Dataset을 이용해 목록 조회를 한다.
 * 검색조건과 검색키워드를 두 개의 Dataset에 세팅한 후 쿼리 문이
 * 정상적으로 동작해 기대했던 값과 조회 결과 값을 비교한다.
 */
public void testGetListUsingDataset() throws Exception {
    DatasetList inDl = new DatasetList();
    VariableList inVl = new VariableList();
    DatasetList outDl = new DatasetList();
    VariableList outVl = new VariableList();

    inVl.add("querySetCount", 2);
    inVl.add("querySet1", "findBoardList");
    inVl.add("querySet2", "findBoardList");

    Dataset dsSearch1 = new Dataset("querySet1");
    dsSearch1.addStringColumn("SEARCH_CONDITION");
    dsSearch1.addStringColumn("SEARCH_KEYWORD");
}

```

```
dsSearch1.appendRow();
dsSearch1.setColumn(0, "SEARCH_CONDITION", "BOARD_NO");
dsSearch1.setColumn(0, "SEARCH_KEYWORD", "BOARD-00002");
inDl.addDataset("querySet1", dsSearch1);

Dataset dsSearch2 = new Dataset("querySet2");
dsSearch2.addStringColumn("SEARCH_CONDITION");
dsSearch2.addStringColumn("SEARCH_KEYWORD");

dsSearch2.appendRow();
dsSearch2.setColumn(0, "SEARCH_CONDITION", "BOARD_TITLE");
dsSearch2.setColumn(0, "SEARCH_KEYWORD", "0001");
inDl.addDataset("querySet2", dsSearch2);

mipService.getList(inVl, inDl, outVl, outDl);

assertEquals(2, outDl.size());

Dataset ds1 = outDl.get("querySet1");
Dataset ds2 = outDl.get("querySet2");

assertEquals("BOARD-00002", ds1.getColumnAsString(0, "BOARD_NO"));
assertEquals("TITLE-0001", ds2.getColumnAsString(0, "BOARD_TITLE"));
}
```

---

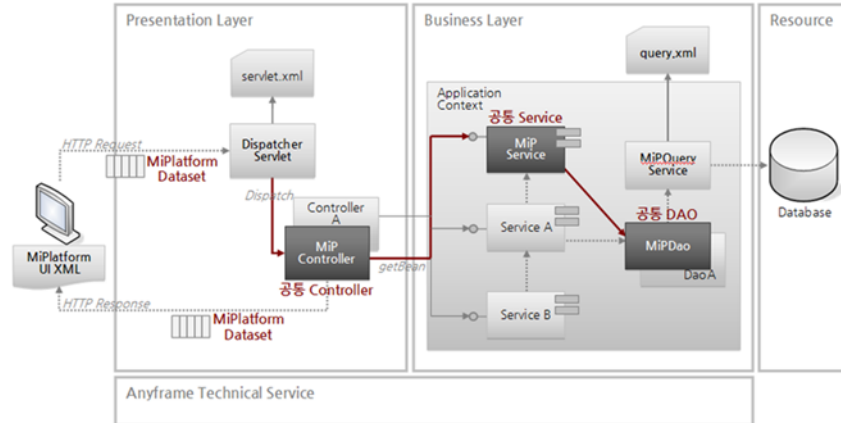
## IV.MiPlatform UI Sample

Java EE 어플리케이션 개발 프레임워크인 Anyframe Java와 RIA UI 개발 플랫폼인 MiPlatform에 대한 연계 방안을 제시함으로써 Anyframe Java와 함께 MiPlatform을 기반으로 어플리케이션을 개발하는 프로젝트에 효과적인 개발 방법을 제안하고자 한다. 프로젝트 초기에 개발 템플릿을 정의하는데 소요되는 시간을 줄이고자, 참조할 수 있는 MiPlatform 화면 샘플 및 Eclipse 프로젝트 샘플을 제공하고, 설치에서부터 구현 기능 및 적용 방법에 대한 설명을 하고자 한다

Anyframe MiPlatform UI Sample은 최근 Rich User Experience 실현을 위해서 많이 사용되고 있는 UI 개발 플랫폼 중 하나인 MiPlatform 을 사용하여 UI를 개발할 때, Anyframe 기반으로 Java EE 어플리케이션을 개발하는 방법을 보여주고, Anyframe과 MiPlatform을 사용하여 진행되었던 몇몇 프로젝트에서 추출한 기본적인 화면들을 구현하여 샘플로 제공하고 있다. 기존에 Anyframe Community Portal을 통해 제공된 MiPlatform Sample과의 차이점은 단순 CRUD 기능에 대해서 불필요한 중복 코딩을 제거하기 위해 공통 Controller, 공통 Service 등으로 구성된 공통 모듈, Anyframe Ria MiPlatform 을 사용한다는 것이다.

## 4. Architecture

본 문서의 이해를 돕기 위하여 Anyframe 기반의 MiPlatform UI Sample 전체 소프트웨어 아키텍처를 나타내면 다음과 같다.



위의 아키텍처는 공통 Controller와 공통 Service, 공통 Dao에 의해서 최적화되며, 일반적인 CRUD 형태의 작업은 이들을 이용하는 공통 Flow(Common Flow)를 통해서 수행이 된다. 만약 단순 CRUD가 아닌 복잡한 비즈니스 로직을 구현하는 경우에 대해서는 공통 Service를 대체하는 별도의 Service (예: Service A, Service B)를 구현하면 되며, 파일업로드/다운로드, 로그인 등과 같이 공통 Controller의 범위를 벗어나는 Controller의 경우에는 Service와 마찬가지로 별도의 Controller(예: Controller A)를 구현하여 수행한다.

위의 그림에서 볼 수 있듯이 Model 2 MVC 구조에 기반한 프레젠테이션, 비즈니스 레이어는 각 레이어 별로 개발 생산성을 향상시키기 위한 공통 클래스를 사용하기 때문에, 경우에 따라서 다음과 같이 3가지 유형으로 개발할 수 있다.

- (1) 단일 테이블에 대한 단순 CRUD의 경우에는 UI XML + Query XML 만을 작성
- (2) 복잡한 로직을 가진 기능의 경우에는 UI XML + Service + Query XML 을 작성
- (3) 파일 업로드, 다운로드 등과 같이 표준적인 인터페이스를 가지지 않은 웹 컨트롤러를 개발하는 경우에는 공통 Controller 대신 별도의 Controller + UI XML + Service + Query XML을 작성

본 샘플에서는 각각의 화면 샘플은 (1)의 방식을, 사용자 로그인 기능은 (3)의 방식을 사용하여 개발되어 있다.

### 4.1. Presentation Layer

본 샘플에 적용된 Web 프레임워크는 MVC 모델의 View와 Controller 영역을 담당하며, 이는 프레젠테이션 레이어에 해당된다. 본 절에서는 프레임워크 기반 개발 시 기본적으로 필요한 설정 파일 구성에 대해서 기술하도록 한다.

web.xml, \*-servlet.xml 파일의 설정은 다음과 같다.

- web.xml

web.xml은 Web Application 배포 지시자로서, JavaEE 환경에서 해당 Web Application이 서버상에 어떻게 배포되어야 하는지에 대하여 기술하는 XML 파일이다. 프레임워크 기반 개발과 관련하여 web.xml 작성 방법은 Anyframe 매뉴얼을 참조하도록 한다. (<http://www.anyframejava.org>) [<http://www.anyframejava.org>]

- common-servlet.xml

본 샘플에서는 각각의 샘플 화면들은 공통 Controller인 MiPController를 사용하고 있고, 로그인 기능에 대해서만 LoginController를 생성하였으므로 common-servlet.xml 파일에 모두 정의하였다.

```
<bean name="/mipController.do"
      class="org.anyframe.mip.query.web.MiPController">
</bean>
<bean name="/mipQueryLogin.do"
      class="org.anyframe.plugin.mip.query.security.web.LoginController" />
```

위의 Controller 정의에서 볼 수 있듯이, 단순 CRUD 기능의 경우에는 개발자가 별도의 Controller를 구현할 필요 없이 MiPController를 사용하면 된다. 다만 파일 업로드/다운로드와 같이 UI와의 인터페이스가 별도로 구성되거나, 또는 MiPController에서 지원하지 못하는 기능을 추가하고자 하는 경우, 개발자의 필요에 따라 AnyframeMiPController나 AnyframeMiPDispatchController 상속을 통해 Controller를 신규로 작성할 수 있다. 본 샘플의 LoginController가 그 예이다.

## 4.2.Business Layer

본 샘플은 Anyframe에서 제공하는 MiPlatform 연계 공통 모듈인 Anyframe Ria MiPlatform을 기반으로 구현되어 비즈니스 레이어에는 로그인 처리를 위한 클래스만 존재한다. 따라서 본 절에서는 Anyframe Ria MiPlatform을 사용하기 위해 필요한 기본적인 설정 파일 구성에 대해서만 기술한다.

주요 설정 파일은 다음과 같다.

- mapping-xxx-xxx.xml

QueryService를 사용할 경우 실행하고자 하는 쿼리를 정의한 파일로, 작성방법은 Anyframe 매뉴얼을 참조하도록 한다. (<http://www.anyframejava.org>) [<http://www.anyframejava.org>]

- context-miplatform.xml

context-xxx.xml은 Spring에서 관리하는 Bean 속성 정의 파일로, context-miplatform.xml에는 Anyframe Ria MiPlatform의 공통 Service, 공통 Dao에 대한 Bean 정의가 명시되어 있다.

```
<bean name="mipService" class="org.anyframe.mip.query.service.impl.MiPServiceImpl">
  <constructor-arg index="0" ref="mipDao" />
</bean>

<bean name="mipDao" class="org.anyframe.mip.query.dao.impl.MiPDaoQuery">
  <constructor-arg index="0" ref="mipQueryService" />
</bean>
```

# 5.Sample UI

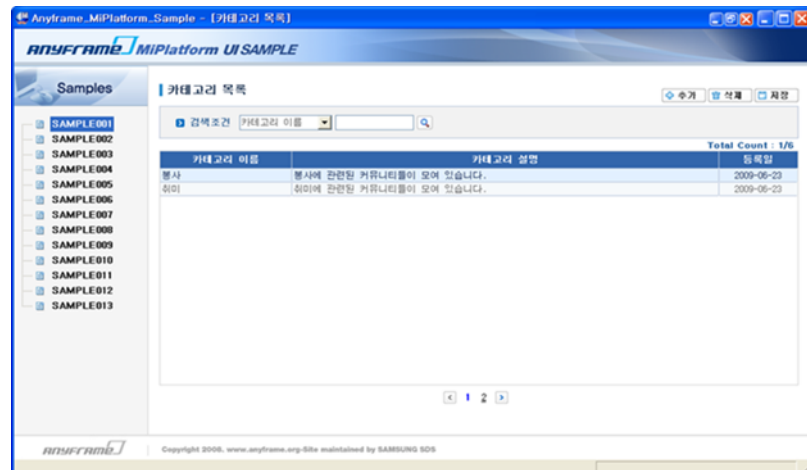
## 5.1.Introduction

샘플로 제공하는 화면은 총 13개이며 Anyframe MiPlatform UI Sample을 실행하면 왼쪽의 메뉴로 제공된다. 각각의 화면은 커뮤니티의 카테고리, 커뮤니티, 각 커뮤니티의 게시글, 사용자 등을 관리하는 기능으로 구성되어 있다. 각 샘플 화면을 통해 UI 컴포넌트의 기본적인 사용법, 이벤트 핸들링, 스크립트 사용법, Anyframe에서 제공하는 공통 모듈 사용하기 위한 Dataset 구성 방법, Validation 처리 방법 등을 보여준다. 각 UI 컴포넌트의 자세한 사용법은 MiPlatform PID의 Help 매뉴얼을 참조하기 바란다.

## 5.2.Set of Sample UI

### 5.2.1.01GRD – 샘플1 (검색 + Grid + Paging Control)

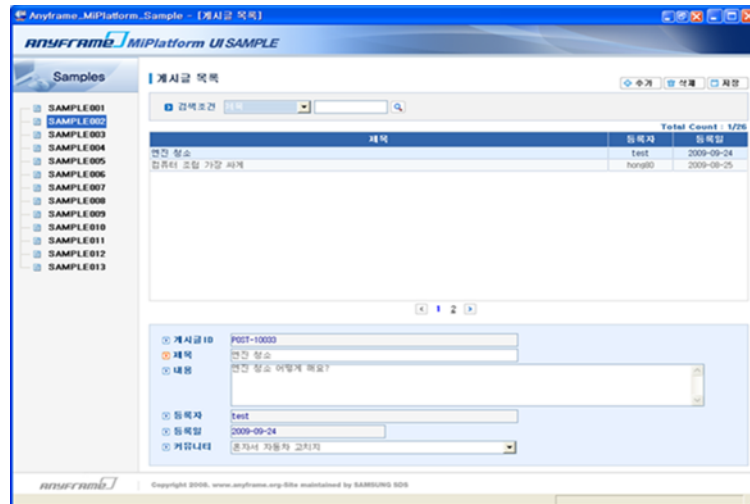
커뮤니티 카테고리 목록을 조회하고 카테고리를 관리하는 기능의 화면이다. Grid 하단에는 Paging Control이 존재한다. 본 샘플에서 Paging은 페이지 번호 클릭 시에 해당 페이지의 데이터만 테이블에 가져오는 방식으로 구현되어 있다.



- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 신규 생성할 카테고리 정보를 입력할 수 있다.
- 기존 카테고리 정보를 Grid에서 직접 수정할 수 있다.
- 카테고리 삭제는 Grid에서 삭제할 카테고리를 선택하고 삭제 버튼을 클릭하면 된다. Grid에서 Shift키나 Ctrl키를 이용한 Multi Row 선택이 가능하므로 여러 항목을 한꺼번에 삭제할 수 있다.
- 사용자가 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

### 5.2.2.02GRDFRM – 샘플2 (검색 + Grid +입력Form + Paging Control)

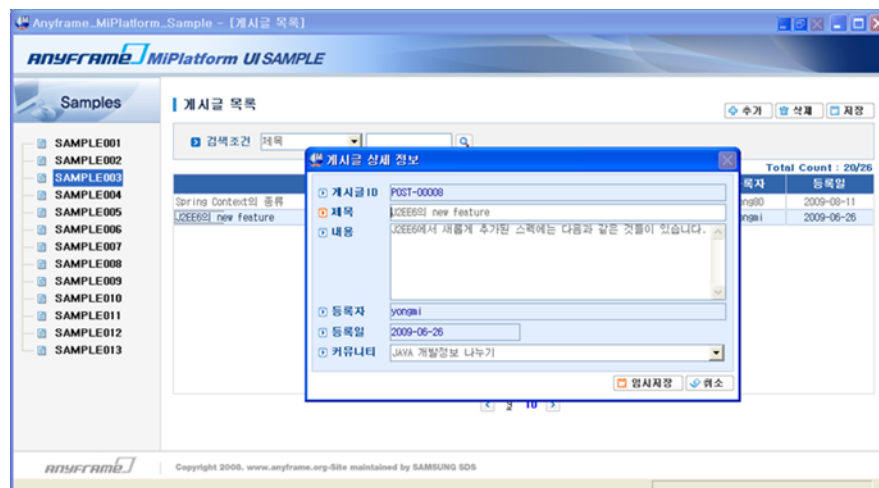
커뮤니티에 등록된 게시글 목록을 조회하고, 게시글을 관리하는 화면이다. Grid 하단에는 Paging Control이 존재한다.



- 목록에서 게시글을 선택하면 하단 입력 Form에서 해당 게시글에 대한 상세 내용을 조회하고 수정할 수 있다.
- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 하단에 빈 입력 Form에서 신규 게시글을 작성할 수 있다.
- 게시글 삭제는 Grid에서 삭제할 게시글을 선택하고 삭제 버튼을 클릭하면 된다. Grid에서 Shift키나 Ctrl키를 이용한 Multi Row 선택이 가능하므로 여러 항목을 한꺼번에 삭제할 수 있다.
- 사용자가 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

### 5.2.3.03GRDPOP – 샘플3 (검색 + Grid + 입력Form 팝업 + Paging Control)

커뮤니티에 등록된 게시글 목록을 조회하고, 게시글을 관리하는 화면이다. Grid 하단에는 Paging Control이 존재한다.



- 목록에서 게시글을 선택하여 더블 클릭하면 상세 내용을 조회하고 수정할 수 있는 팝업 창이 나타난다.
- 추가 버튼을 클릭하면 빈 입력 Form이 팝업 창으로 나타나고 신규 게시글을 작성할 수 있다.
- 게시글 삭제는 Grid에서 삭제할 게시글을 선택하고 삭제 버튼을 클릭하면 된다. Grid에서 Shift키나 Ctrl키를 이용한 Multi Row 선택이 가능하므로 여러 항목을 한꺼번에 삭제할 수 있다.
- 사용자가 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.



## 5.2.4.04GRDTAB – 샘플4 (검색 + 상단 Grid + 하단 Tab 입력 Form)

사용자 목록을 조회하고, 관리하는 화면이다. 하단에는 사용자의 상세정보를 조회할 수 있는 Tab이 존재한다.

The screenshot displays the 'AnyFrame MiPlatform UI Sample' application. On the left, a 'Samples' sidebar lists various sample applications. The main area is titled '사용자 목록' (User List) and features a search bar and a table of users. The table has columns for '이름' (Name), 'ID', '전자우편' (Email), '전화' (Phone), and '이동전화' (Mobile Phone). Below the table, there are two tabs: '기본정보' (Basic Info) and '추가정보' (Additional Info). The '기본정보' tab is active, showing fields for '사용자' (User), 'ID', '이름' (Name), 'Password', and '영문이름' (English Name). The '추가정보' tab shows fields for '주소' (Address), '주요번호' (Main Number), '선택주소' (Selected Address), '이메일주소' (Email Address), '이동전화' (Mobile Phone), and '주요주소' (Main Address).

- 목록에서 사용자를 선택하면 하단 Tab 입력 Form에서 해당 사용자에 대한 상세 내용을 조회하고 수정할 수 있다.
- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 하단에 빈 Tab 입력 Form이 나타나 신규 사용자를 생성할 수 있다.
- Grid에서 삭제할 사용자의 Checkbox에 체크를 하고 삭제 버튼을 클릭하면 사용자를 삭제할 수 있다. 또한 Checkbox를 이용해서 여러 사용자를 한꺼번에 삭제할 수 있다.
- 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

## 5.2.5.05GRDTAB – 샘플5 (검색 + 좌측 Grid + 우측 Tab 입력Form)

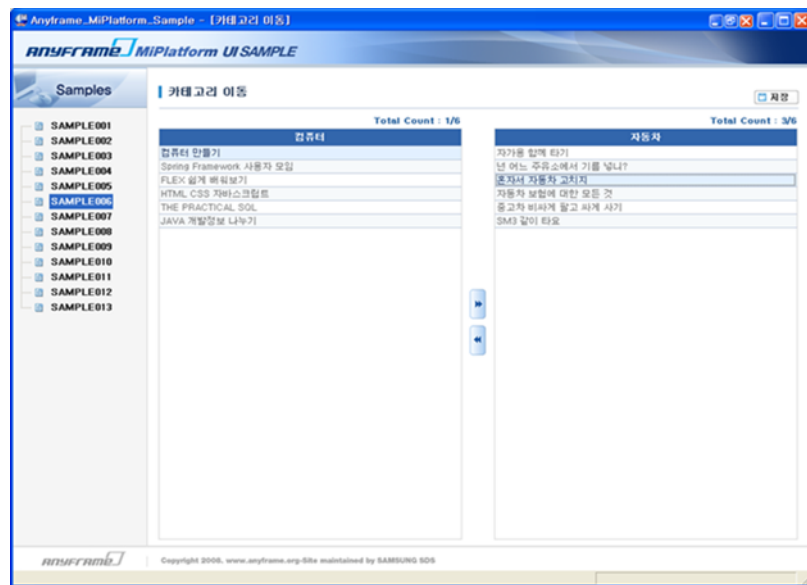
사용자 목록을 조회하고, 관리하는 화면이다. 우측에 사용자의 상세정보를 조회할 수 있는 Tab이 존재한다.

The screenshot displays the 'AnyFrame MiPlatform UI Sample' application. On the left, a 'Samples' sidebar lists various sample applications. The main area is titled '사용자 목록' (User List) and features a search bar and a table of users. The table has columns for '이름' (Name), 'ID', '전자우편' (Email), '전화' (Phone), and '이동전화' (Mobile Phone). To the right of the table, there are two tabs: '기본정보' (Basic Info) and '추가정보' (Additional Info). The '기본정보' tab is active, showing fields for '사용자' (User), 'ID', '이름' (Name), 'Password', and '영문이름' (English Name). The '추가정보' tab shows fields for '주소' (Address), '주요번호' (Main Number), '선택주소' (Selected Address), '이메일주소' (Email Address), '이동전화' (Mobile Phone), and '주요주소' (Main Address).

- 목록에서 사용자를 선택하면 우측 Tab 입력 Form에서 해당 사용자에 대한 상세 내용을 조회하고 수정할 수 있다.
- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 우측에 빈 Tab 입력 Form이 나타나 신규 사용자를 생성할 수 있다.
- Grid에서 삭제할 사용자의 Checkbox에 체크를 하고 삭제 버튼을 클릭하면 사용자를 삭제할 수 있다. 또한 Checkbox를 이용해서 여러 사용자를 한꺼번에 삭제할 수 있다.
- 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

## 5.2.6.06GRDGRD – 샘플6 (좌/우Grid 간 항목 이동)

커뮤니티의 카테고리를 이동하는 화면이다. 좌측에 컴퓨터 카테고리에 속한 커뮤니티 목록이 나열되고, 우측에 자동차 카테고리에 속한 커뮤니티가 나열된다. 화면 가운데 버튼을 이용하여 커뮤니티를 이동시킬 수 있다. 좌/우 Grid 간 항목 이동 방법을 보여주는 샘플이다.



- 목록에서 커뮤니티를 선택하고 좌측으로 이동하는 버튼 또는 우측으로 이동하는 버튼을 클릭하면 Grid에서 커뮤니티가 이동한다.
- 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

## 5.2.7.07GRDGRD – 샘플7 (검색 + 상단 Master Grid + 하단 Sub Grid + Validation)

커뮤니티 목록과 커뮤니티에 등록된 게시물 목록을 조회하고, 게시글을 관리할 수 있는 화면이다. 하단의 커뮤니티 목록은 상단의 카테고리 목록을 클릭했을 때 해당 카테고리 ID를 가지고 MiPlatform 자체 Service API인 filter()를 사용해서 출력한다. 하단 커뮤니티 목록에서 입력데이터에 대한 Validation 처리가 적용되어 있다.



- 상단 Master Grid에서 커뮤니티를 선택하면 하단 Sub Grid에 해당 커뮤니티에 속한 게시글 목록이 나타난다.
- 추가 버튼을 클릭하면 Sub Grid에 Row가 추가되고 신규 게시글을 작성할 수 있다.
- 기존 게시글 정보를 Grid에서 직접 수정할 수 있다.
- Grid에서 삭제할 게시글의 Checkbox에 체크를 하고 삭제 버튼을 클릭하면 된다. 또한 Checkbox를 이용해서 여러 게시글을 한꺼번에 삭제할 수 있다.
- 사용자가 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.
- Grid 입력 데이터에 대해 공통Script함수인 gfnValidate()를 이용한 Validation 처리가 적용되어 있다.

## 5.2.8.08TRVGRD – 샘플8 (검색 + Tree + Grid + Tab 입력 Form 팝업 + Validation)

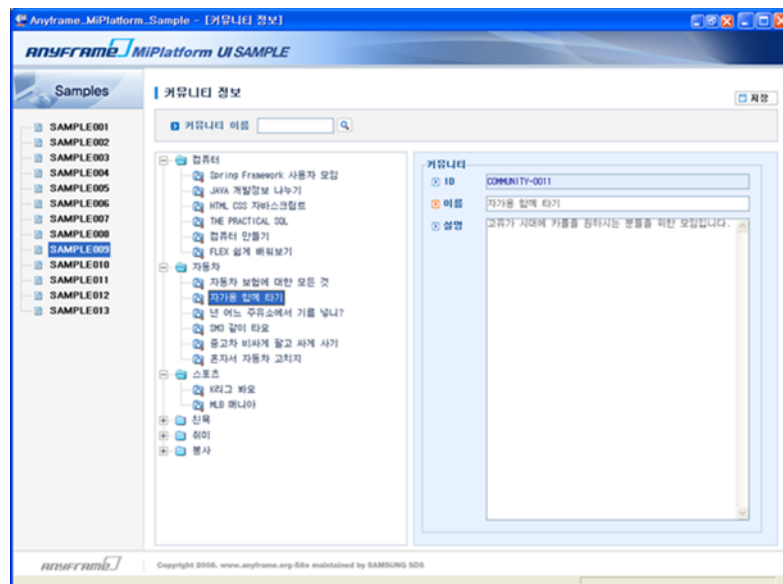
부서 목록을 Tree로 조회하고, 부서에 속한 사용자를 관리하는 화면이다. Tree와 Grid 사용 방법을 제시하고 있다. 입력데이터에 대한 Validation 처리가 적용되어 있다.



- 좌측 Tree에서 부서를 선택하면 우측 Grid에 해당 부서에 속한 사용자 목록이 나열된다.
- 우측 목록에서 사용자를 선택하여 더블 클릭하면 사용자에 대한 상세 내용을 조회하고 수정할 수 있는 Tab 입력 Form 팝업 창이 나타난다.
- 추가 버튼을 클릭하면 빈 Tab 입력 Form 팝업 창이 나타나 신규 사용자를 생성할 수 있다.
- 사용자 삭제는 Grid에서 삭제할 사용자를 선택하고 삭제 버튼을 클릭하면 된다. Grid에서 Shift키나 Ctrl키를 이용한 Multi Row 선택이 가능하므로 여러 항목을 한꺼번에 삭제할 수 있다.
- 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.
- 공통Script함수인 gfnValidate()를 이용한 Validation 처리가 적용되어 있다.

## 5.2.9.09TRVFRM – 샘플9 (검색 + Tree + 입력 Form)

카테고리와 커뮤니티 목록을 Tree로 조회하고, 커뮤니티 정보를 관리하는 화면이다. Tree와 입력 Form 사용 방법을 제시하고 있다.



- 좌측 Tree에서 카테고리를 선택하면 우측에 카테고리 정보가 나타난다.
- 좌측 Tree에서 커뮤니티를 선택하면 우측에 커뮤니티 정보가 나타난다.
- 수정한 커뮤니티 정보는 저장버튼을 클릭했을 때 일괄 저장된다.

## 5.2.10.10CTGGRD – 샘플10 (카테고리구분 + 검색 + Grid)

카테고리 별 커뮤니티 목록을 조회하는 화면이다. Grid를 이용하여 카테고리 구분표를 구성하는 방법을 제시하고 있다. 하단의 커뮤니티 목록은 상단의 카테고리를 클릭했을 때 해당 카테고리 ID를 가지고 MiPlatform 자체 Service API인 filter()를 사용해서 출력한다.



- 상단에서 카테고리를 선택하면 하단에 커뮤니티 목록이 나열된다.
- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 신규 생성할 커뮤니티 정보를 입력할 수 있다.
- 기존 커뮤니티 정보를 Grid에서 직접 수정할 수 있다.
- 삭제할 커뮤니티의 Checkbox를 체크하고 삭제 버튼을 클릭하면 커뮤니티를 삭제할 수 있다. Checkbox를 이용하여 여러 개의 항목을 한꺼번에 삭제할 수 있다.
- 사용자가 입력/수정/삭제 등 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

## 5.2.11.11CALMTLY – 샘플11 (Grid를 이용한 월간 Calendar)

월간 일정 관리 화면이다. Grid를 이용하여 Calendar를 구성하는 방법을 보여준다.



- 버튼을 이용하여 월별로 이동하며 일정을 조회할 수 있다.
- 기존에 정의한 일정이 존재하는 경우 해당 날짜를 더블 클릭하면 일정 상세조회 팝업 창이 나타난다.
- 일정이 정의되지 않은 날짜를 더블 클릭하면 신규 일정을 추가할 수 있는 팝업 창이 나타난다.

## 5.2.12.12CALWKLY – 샘플12 (Grid를 이용한 주간 Calendar)

주간 일정 관리 화면이다. Grid를 이용하여 Calendar를 구성하는 방법을 보여준다.



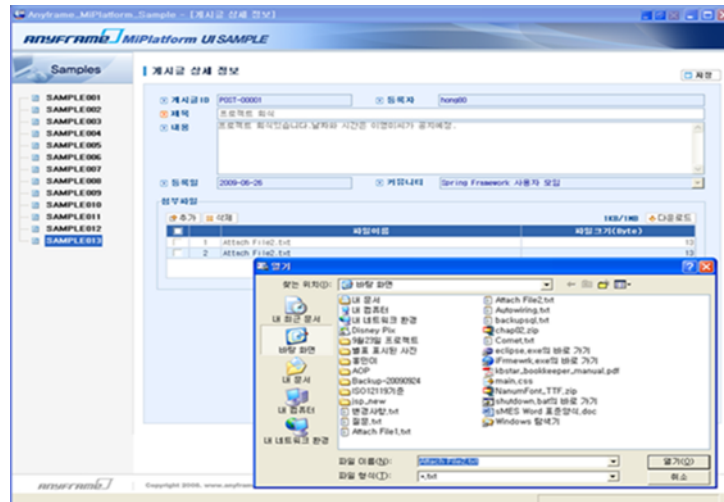
- 버튼을 이용하여 주 별로 이동하며 일정을 조회할 수 있다.
- 기존에 정의된 일정이 존재하는 경우 해당 날짜를 더블 클릭하면 일정 상세조회 팝업 창이 나타난다.
- 일정이 정의되지 않은 날짜를 더블 클릭하면 신규 일정을 추가할 수 있는 팝업 창이 나타난다.

## 5.2.13.13FILEATT - 샘플13 (파일 첨부)

샘플13에서 File 첨부 기능을 위한 HttpFile Component를 사용하기 위해서는 다음과 같은 절차가 선행 되어야 한다.

- 현재 사용하는 MiPlatform 버전에 맞는 CyHttpFile.dll, CyHttpFileU.dll 파일을 TOBE SOFT 홈페이지에서 다운로드 받는다. (본 샘플은 330U 버전으로 작성 되었다.)
- 다운로드 받은 두 개의 dll 파일을 C:\User\현재사용자\AppData\Local\TOBESOFT\MiPlatform330U\Component 폴더로 옮긴다. (Windows XP의 경우 C:\Documents and Settings\현재사용자\AppData\Local\TOBESOFT\MiPlatform330U\Component)

'POST-00001'이라는 ID의 게시글의 상세 정보 조회화면으로, File 컴포넌트와 File Dialog 컴포넌트를 사용하여 파일 첨부 기능을 구현한 샘플화면이다. 첨부된 파일을 context.properties 파일에 정의된 repository.path 상에 지정된 위치로 업로드를 한다.



- 추가 버튼을 클릭하면 파일을 하나씩 첨부할 수 있는 File Dialog 창이 나타난다.
- 첨부 파일 목록에서 Checkbox에 체크를 하고 다운로드 버튼을 클릭하면 파일을 다운로드 할 폴더를 선택할 수 있는 Dialog가 나타난다.
- 첨부 파일 목록에서 첨부파일을 더블 클릭하면 첨부파일을 저장할 위치와 파일 이름을 지정할 수 있는 Dialog가 나타난다.
- 삭제할 첨부파일의 Checkbox를 체크하고 삭제버튼을 클릭하면 첨부파일을 삭제할 수 있다. Checkbox를 이용하여 여러 개의 항목을 한꺼번에 삭제할 수 있다.
- 사용자가 편집한 내용은 저장 버튼을 클릭했을 때 한꺼번에 테이블에 저장된다.

## 6. Standards

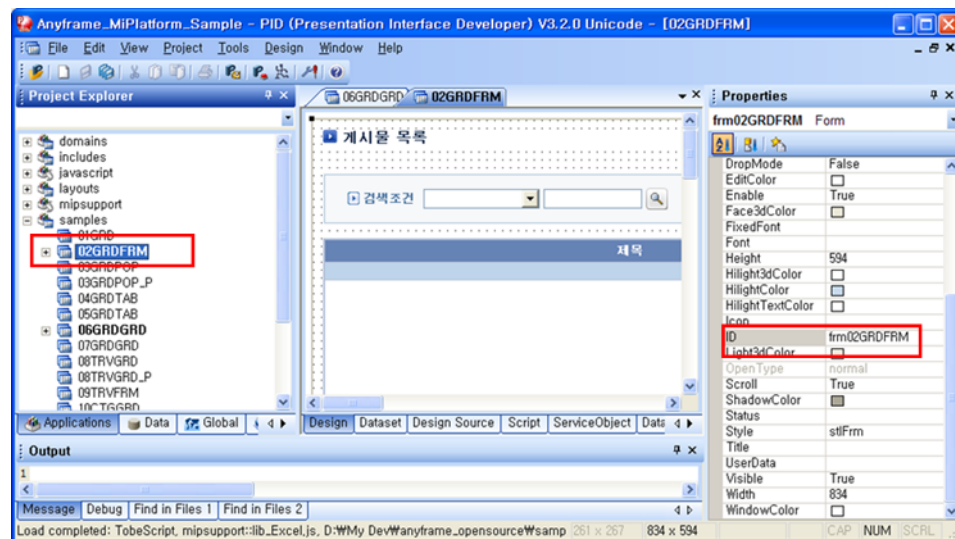
### 6.1. Naming Rules of Form

화면 파일 하나당 Form은 하나로 구성되어 있으며 Form의 ID는 "frm"+ 파일명 형태로 이루어진다.

표 6.1. 예)

File Name	Example of Form ID
O1GRD.xml	frmO1GRD
CategoryMgmt.xml	frmCategoryMgmt

PID에서 Form ID는 Form의 속성으로 지정한다.



### 6.2. Naming Rules of UI Component

UI 컴포넌트를 나타내는 Prefix와 의미 있는 이름을 조합하여 정의한다.

- UI 컴포넌트 별 영문 약어를 Prefix로 사용
- 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분

예) 사업장 'Combo': "cboPlantCd", 목록을 출력하는 'Grid': "grdTitleList", 업체명을 수정하는 'Edit': "edtClientNm", 조회 'Button': "btnSearch", 저장 'Button': "btnSave", 초기화 'Button': "btnReset",

- Dataset의 이름은 Prefix ds + 'Dataset이 Binding되는 컴포넌트의 이름'으로 정의

예) dsCboRepairItem

- Dataset의 컬럼 ID는 DB Table 컬럼 명을 그대로 사용

UI 컴포넌트 별 Prefix는 다음과 같다.



**표 6.2.**

UI Component	Prefix	Example of Component ID
Button	btn	btnSave, btnSearch
Calendar	cal	calFromDt, calToDt
Checkbox	chk	chkAll
ComboBox	cbo	cboAcclItem
Dataset	ds	dsUser, dsMenu
Division	div	divUserInfo
Edit	edt	edtUserNm
File	file	fileUserImg
FileDialog	fdlg	fdlgUserImg
Flash	fls	flsMenu
Grid	grd	grdClientList
Image	img	imgTitle
List	lst	lstMenu
MaskEdit	mdt	mdtAmount
MenuBar	mb	mbTopMenu
Pie	pie	pieChart
PopupDiv	pdiv	pdivMemo
Progressbar	pb	pbLoading
Radio	rdo	rdoYn
Shape	shp	shpBox
Spin	sp	spAddVal
Static	st	stName
Tab	tab	tabClient
TeeChart	tc	tcIncome
TextArea	txa	txaMemo
TreeView	trv	trvMenu
WebBrowser	web	webMail

## 6.3.Naming Rules of Variable

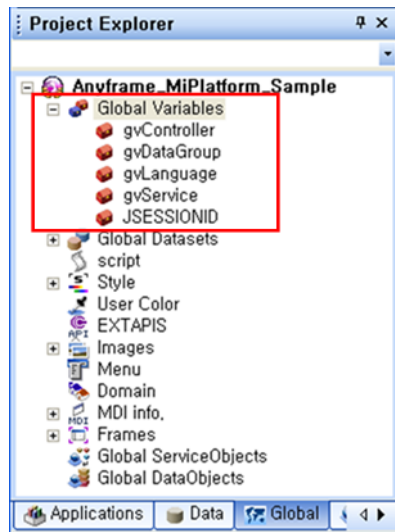
### 6.3.1.Global Variable

MiPlatform PID의 Global 탭에서 작성하고 Start XML내 <Variables> 및 <Script> 에 선언되는 전역변수로 어플리케이션 전역에서 사용 가능하다.

- "gv" 로 시작하고 영문 약어 사용.
- 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분

예) gvLanguage, gvController, .....

- 예외: 'JSESSIONID' - Web Server의 JSESSIONID 값을 화면에서 사용하기 위해 Web Server의 Cookie 변수와 동일한 변수 명으로 정의



## 6.3.2.Common Script Variable

공통 스크립트(예: javascript/common.js) 내에 선언되며, 스크립트를 include하는 모든 Form 에서 사용 가능하다.

- Form 스크립트에 include 형태로 포함됨
- "g" 로 시작하고 영문 약어 사용
- 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분

예) gSysdate, gUserName, .....

- 함수 안에서 선언되는 변수는 '일반 변수' 규칙을 따름

## 6.3.3.Local Variable

각 화면 Form 스크립트에서 사용하는 변수는 사용하고자 하는 데이터 타입에 따라 Prefix를 붙이고 UI 상의 의미를 정의한 영문 약어를 사용한다. 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분한다.

**표 6.3.**

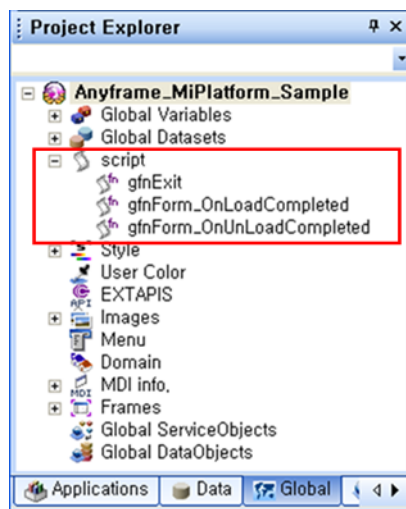
Data Type	Prefix	Example of Variable
문자형	str	strUserNm, strUserId, ...
숫자형	n	nTotAmt, nTaxAmt, ...
boolean형	b	bIsNull, bHasAuth, ...
Array형	arr	arrCategoryIds, arrHolidays, ...
DateTime형	dt	dtToday
Object	obj	objDataset, objGrid

## 6.4.Naming Rules of Function

### 6.4.1.Global Function

MiPlatform PID의 Global 탭에서 작성하고 Start XML내 <Script> 에 정의되는 함수로, include 하지 않아도 어플리케이션 전역에서 사용 가능하다. 함수의 기능을 표현하는 영문 약어로 정의한다.

- "gfn" 으로 시작
- 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분  
예) gfnSetUserInfo(strPara1) { ... }, gfnSetAuth(strPara1, strPara2) { ... }
- 이벤트 발생 시 작동하는 공통 함수명은 "gfn" + UI컴포넌트명 + "\_" + 이벤트명  
예) gfnForm\_OnUnLoadCompleted ({ ... }, gfnForm\_OnLoadCompleted{ ... }



### 6.4.2.Common Script Function

공통 스크립트(예: javascript/common.js) 내에 선언되며, 스크립트를 include하는 모든 Form 에서 사용 가능하다. 함수의 기능을 표현하는 영문 약어로 정의한다.

- "gfn" 으로 시작
- 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분  
예) gfnIsNull(strArg) { ... }, gfnIsNotNull(strArg) { ... }
- 이벤트 발생 시 작동하는 공통 함수명은 "gfn" + UI컴포넌트명 + "\_" + 이벤트명  
예) gfnGrid\_OnHeadClick(){...}, gfnForm\_OnLoadCompleted{...}

### 6.4.3.Local Function

일반적으로 개발자들이 화면 개발 시에 Form 스크립트 안에 선언하는 함수로 UI 컴포넌트에서 이벤트 발생 시 작동하는 함수 명은 PID의 Design 탭에서 UI 컴포넌트를 더블 클릭했을 때 자동으로 생성되는 이름을 사용한다.

- UI 컴포넌트 ID + "\_" + 이벤트명  
예) btnValNm\_Onclick() { ... }, cbValNm\_OnChange() { ... }

그 외 개발자의 필요에 의해 직접 정의하는 경우 함수의 기능을 표현하는 영문약어를 사용한다.

- "fn"으로 시작, 단어와 단어 사이는 단어 첫 알파벳을 대문자로 시작하여 구분

예) fnCallback() { ... }, fnGetUserInfo() { ... }

## 7. Working with Common Flow

앞에서 설명했듯이 Anyframe에서는, MiPlatform을 UI 개발 플랫폼으로 사용할 때 단순 CRUD작업에 대해서 어떠한 화면에서도 **공통Flow**를 거쳐 기능 처리가 가능하도록 Anyframe Ria MiPlatform 을 제공하고 있다. 본 절에서는 공통 Flow를 사용하기 위해서 MiPlatform 화면 개발 시에 반드시 설정해 주어야 하는 부분에 대해서 설명한다.

### 7.1. Common Script

모든 화면의 Form 스크립트에는 반드시 최상위에 “**#include javascript::common.js**” 을 기술해야 한다. common.js에는 Anyframe MiPlatform UI Sample에서 공통 적으로 사용되는 스크립트 함수들이 정의되어 있고, 또한 그 외 공통 스크립트(\*.js) 파일들을 포함하고 있다. (예:message.js, util.js) common.js 에 정의된 함수 중 Service Call과 관련된 함수는 다음과 같다.

#### 7.1.1. Service Call

gfnService()는 서버 측에 Service를 요청하기 위한 함수로 내부적으로 MiPlatform 자체 Service API인 transaction()을 호출한다.

##### 7.1.1.1. Syntax

**gfnService(strServiceId, strArgument)**

예) gfnService(“getListCommunity”), gfnService(“saveAllBoard”)

##### 7.1.1.2. Parameters

Parameter	Description
strServiceId	사용자가 임의로 정하는 Service에 대한 고유한 ID 접두어는 아래 정의된 내용으로 한정함. <ul style="list-style-type: none"><li>• <b>getList</b>: 조회</li><li>• <b>getPagingList</b>: 페이지조회</li><li>• <b>get</b>: 단건 조회</li><li>• <b>create</b>: 등록</li><li>• <b>update</b>: 수정</li><li>• <b>remove</b>: 삭제</li><li>• <b>saveAll</b>: 등록/수정/삭제 모두 수행 (하나의 Transaction)</li><li>• <b>execute</b>: DBMS의 Stored Procedure 실행</li></ul> 예) getListCommunity
strArgument	Service 호출 시 HTTP GET 방식으로 전달할 파라미터 Syntax: “name1=value1 name2=value2”

## 7.1.2.Callback

MiPlatform에서는 기본적으로 Service 호출이 Async 방식으로 이루어진다. gfnService()를 이용해 서버 측에 요청을 보냈을 경우, 요청에 대한 응답이 도착했을 때 디폴트로 호출되는 함수는 **gfnCallback()** 이다. gfnCallback() 내부에서는 디폴트로 화면 스크립트 내의 **fnCallback()**을 호출한다. 따라서 Service 호출이 완료된 후에 메시지 처리나 컴포넌트 reload 등의 기능을 수행하도록 적절하게 fnCallback() 을 정의하면 된다.

### 7.1.2.1.Syntax

**fnCallback(strServiceId, strErrorCode, strErrorMsg)**

예)

```
function fnCallback(strServiceId, strErrorCode, strErrorMsg) {
    if ( strErrorCode == -1 ) {
        gfnMsg(strErrorMsg, "ERR");
    } else {
        if(strServiceId == "getPagingListBoard") {
            divPage.objListDataset = dsGrdBoard;
            divPage.objPageDataset = dsSearch;
            divPage.fnMakePage();
        } else if(strServiceId == "saveAllBoard") {
            gfnMsg("MSG_SAVE_SUCCESS");
            divSearch_btnSearch_OnClick();
        }
    }
}
```

### 7.1.2.2.Parameters

Parameter	Description
strServiceId	gfnService()로 Service 호출 시 입력했던 strServiceId와 동일한 값
strErrorCode	에러 코드, '-1' 인 경우 Error
strErrorMsg	호출한 Service에서 넘겨준 에러 메시지

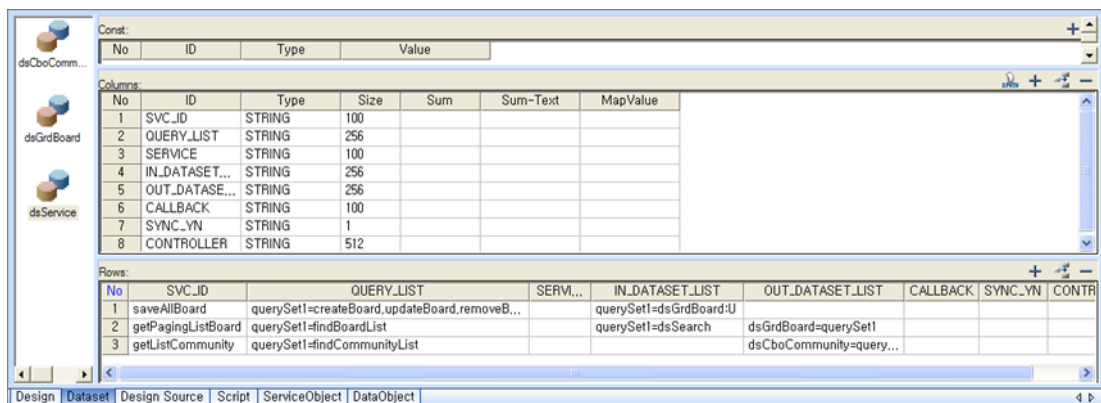
## 7.2.Common Dataset

### 7.2.1.Dataset for Service

데이터를 가져오기 위해서 gfnService()를 통해 Service를 호출할 경우 호출할 Service에 대한 여러 가지 파라미터 정보를 Dataset에 정의한다. 따라서 모든 화면에 다음과 같은 값들이 정의된 **'dsService'**라는 이름의 Dataset이 반드시 포함되어야 한다.

Column	Description
SVC_ID	gfnService()의 strServiceId와 동일한 값을 지님 예) getListCommunity
QUERY_LIST	Service에서 실행해야 할 쿼리ID (space로 구분하여 다중입력 가능) Syntax: "querySet" + 순번 + "=" + 쿼리ID 예) querySet1=getListMethodCode querySet1=getListMethodCode querySet2=getListCategory

Column	Description
	SVC_ID가 'saveAll'로 시작하는 경우(Grid 일괄 저장 시)는 comma(.)를 사용하여 <b>create, update, remove</b> 쿼리를 한번에 지정  예) querySet1=createCategory, updateCategory, removeCategory
SERVICE	공통Service를 사용하지 않을 경우 호출할 Service 정보 미지정시, 디폴트로 'gvService'에 설정된 공통Service인 'mipService' 가 호출됨. Syntax: Service명 + "." + Method명  예) categoryMgmtService.getListCategory, categoryMgmtService.createCategory
IN_DATASET_LIST	쿼리 실행 시, 파라미터로 사용될 Dataset ID (space로 구분하여 다중 입력 가능) Syntax: "querySet"+ 순번 + "=" + DatasetID  예) querySet1=dsSearch querySet1=dsSearch querySet2=dsParam
OUT_DATASET_LIST	쿼리 수행 결과로 받을 Dataset 이름 목록 (다중입력 가능 - space로 구분) Syntax: DatasetID + "=" + "querySet" + 순번  예) dsSearch=querySet1 dsSearch=querySet1 dsParam=querySet2
CALLBACK	Service로부터 응답을 받았을 때 실행될 Callback 함수 이름 gfnCallback() 내에서 호출. 미지정시, 디폴트로 'fnCallback' 호출.
SYNC_YN	Y: Sync 호출 N: Async 호출 (default) – 권장
CONTROLLER	공통Controller를 사용하지 않을 경우 호출할 Controller 정보 미지정시, 디폴트로 'gvController'에 설정된 공통Controller인 'mipController.do' 가 호출됨.  공통Controller를 사용하지 않는 경우, 호출 할 Service에 대한 정보는 신규 작성한 Controller 에서 포함하고 있으므로 SERVICE 속성을 따로 정의하지 않아도 됨.



No	ID	Type	Size	Sum	Sum-Text	MapValue
1	SVC_ID	STRING	100			
2	QUERY_LIST	STRING	256			
3	SERVICE	STRING	100			
4	IN_DATASET...	STRING	256			
5	OUT_DATASE...	STRING	256			
6	CALLBACK	STRING	100			
7	SYNC_YN	STRING	1			
8	CONTROLLER	STRING	512			

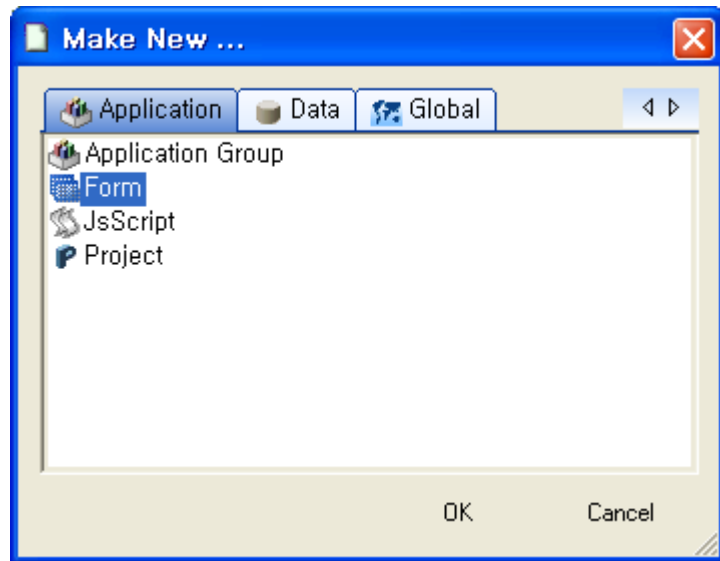
  

No	SVC_ID	QUERY_LIST	SERV...	IN_DATASET_LIST	OUT_DATASET_LIST	CALLBACK	SYNC_YN	CONTR
1	saveAllBoard	querySet1=createBoard,updateBoard,removeB...		querySet1=dsGrdBoard.tJ				
2	getPagingListBoard	querySet1=findBoardList		querySet1=dsSearch	dsGrdBoard=querySet1			
3	getListCommunity	querySet1=findCommunityList			dsCboCommunity=query...			

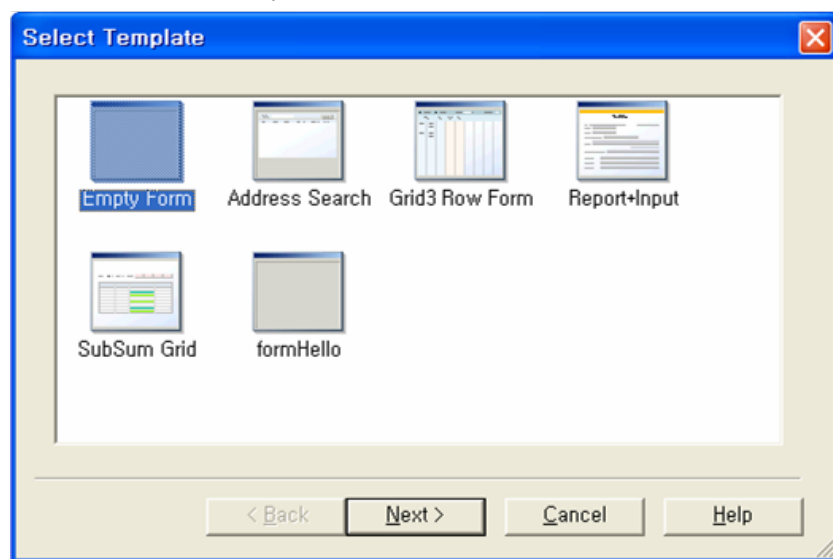
## 7.3.Example

위에서 설명한 내용들을 바탕으로 공통Flow를 통해 커뮤니티 목록을 조회하여 Grid로 출력하는 단순 조회화면을 생성해본다.

- MiPlatform PID의 Project Explorer에서 'samples' AppGroup을 선택한 상태로 메뉴 바의 File > New 를 클릭한다.
- 'Make New...' 팝업 창에서 'Form'을 선택하고 OK버튼을 클릭한다



- Select Template 팝업 창에서 'Empty Form'을 선택하고 Next 버튼을 클릭한다.



- Form Properties 팝업 창에서 Form의 이름과 파일명, 제목 등을 적절하게 입력하고 Finish 버튼을 클릭하면 새로운 Form이 생성된다.



**Form Properties**

Name: FORM\_EXAMPLE Title: Form for Example

Form ID: frmFORM\_EXAMPLE Character Set: utf-8

Path: D:\My Dev\Wanyframe\_opensource\samples\trunk\dev\Wanyframe,samples,

Save Format: ☒ Include Event Script ☐ Component & Dataset, Image only

Application Group: samples Script:

< Back Finish Cancel Help

- 새로 만든 Form 편집 창에서 하단의 Dataset 탭을 열어 +버튼을 이용하여 Dataset('dsService', 'dsGrdCommunity')을 추가한다. 또는 다른 샘플화면에서 'dsService'와 'dsGrdCommunity' Dataset을 복사하여 붙여 넣기 한다.



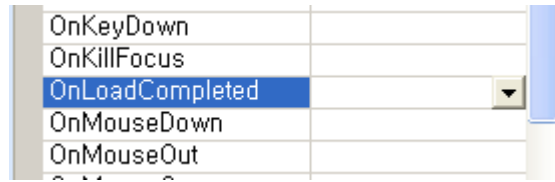
'dsService'은 아래와 같은 컬럼 데이터를 포함한다.

Column	Description
SVC_ID	getListCommunity
QUERY_LIST	querySet1=findCommunityList
SERVICE	
IN_DATASET_LIST	
OUT_DATASET_LIST	dsGrdCommunity=querySet1
CALLBACK	
SYNC_YN	
CONTROLLER	

'dsGrdCommunity'는 아래와 같이 구성되어 있다.

Column	Type	Size
COMMUNITY_ID	STRING	16
COMMUNITY_NAME	STRING	256
COMMUNITY_DESC	STRING	256
CATEGORY_ID	STRING	16
REG_ID	STRING	256
REG_DATE	STRING	10

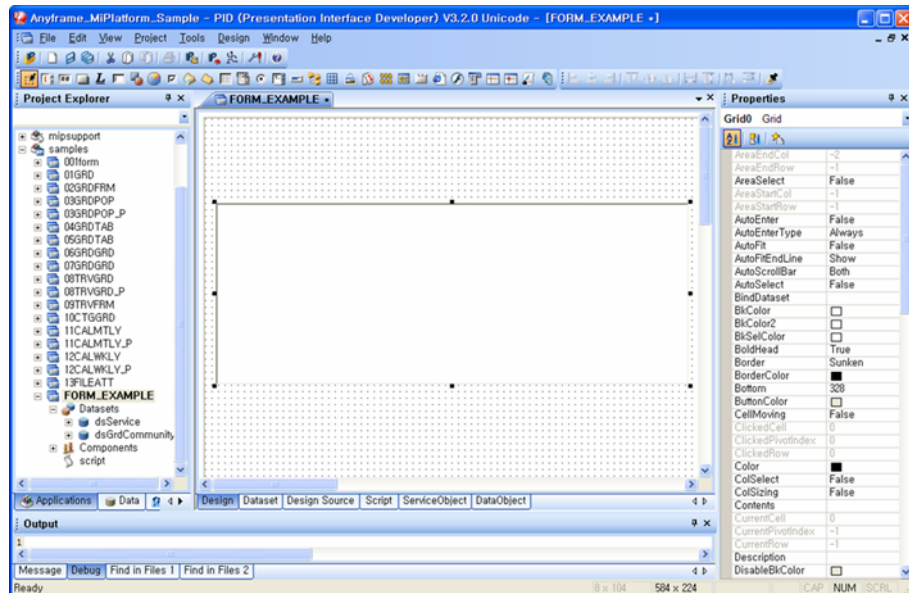
- Design 탭을 클릭하여 Form 편집 창을 연 다음, 오른쪽의 Properties에서 이벤트 보기 아이콘 을 클릭 하고 Form의 OnLoadCompleted 이벤트의 ComboBox 부분을 더블 클릭 한다.



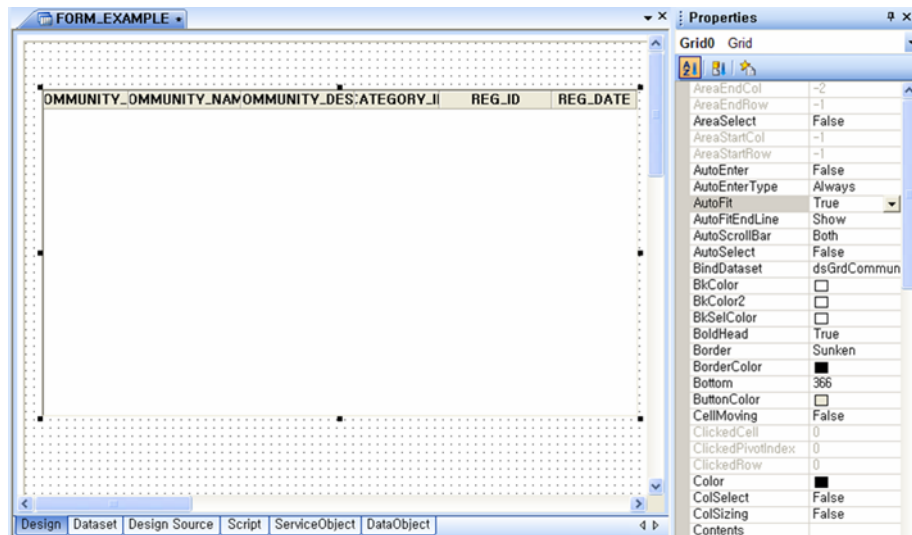
- OnLoadCompleted 이벤트 발생 시에 호출될 Script 함수가 자동으로 생성되며 Script 편집 창으로 이동한다. 아래와 같이 Script 편집 창 가장 상단에 common.js를 include 하도록 선언하고, 자동 생성된 Script함수를 수정한다.

```
#include "javascript::common.js";
function frmFORM_EXAMPLE_OnLoadCompleted(obj) {
    gfnService("getListCommunity");
}
```

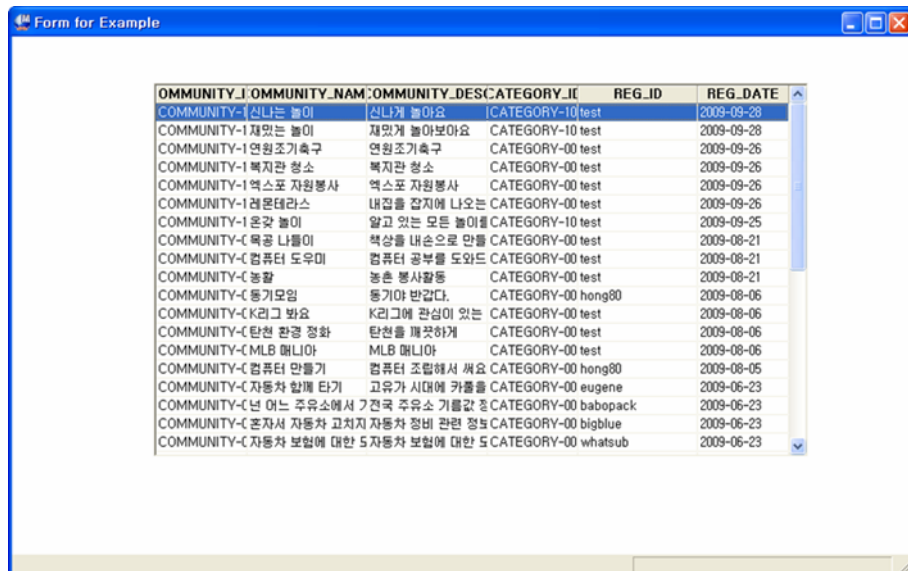
- 다시 Design탭을 클릭하여 Form 편집 창을 열고, Components Tool Bar에서 Grid 아이콘 을 클릭한다. Form 위에서 Drag하여 적당한 크기의 Grid를 생성한다.



- 왼쪽의 Project Explorer에서 FORM\_EXAMPLE 하위의 Datasets 중 'dsGrdCommunity'를 선택하여 위에서 생성한 Grid위로 Drag하고, 오른쪽 Properties에서 속성보기 아이콘 을 클릭한 후, AutoFit 속성을 True로 변경한다.



- 상단 Tool Bar에서 'Quick View' 아이콘을 클릭하면 커뮤니티 목록이 출력되는 것을 확인할 수 있다.



---

## 8.Validation

MiPlatform을 이용하여 화면을 개발할 때 사용자 입력 정보에 대한 Validation 처리는 어떻게 할 수 있는지 알아본다.

### 8.1.Using UI Component

MiPlatform PID로 화면 작성 시 UI 컴포넌트의 속성을 이용하면 기본적인 Validation 처리를 할 수 있다. 자세한 내용은 MiPlatform PID의 Help 매뉴얼을 참고한다.

#### 8.1.1.Size Validation

입력될 정보의 Size를 지정하는 방법이다.

##### 8.1.1.1.Grid

Body Cell Property을 이용한다.

Property	Description
Limit	최대 입력 길이 제한.
CheckLength	Char: 글자 단위로 길이를 제한. 모든 1개의 글자가 1단위로 계산됨 Byte: Byte단위로 길이를 제한. Byte단위의 경우 아시아 언어와 일부 기호가 2단위로 계산

##### 8.1.1.2.EditBox

Property	Description
MaxLength	최대 입력 길이 제한.
CheckLength	Char: 글자 단위로 길이를 제한. 모든 1개의 글자가 1단위로 계산됨 Byte: Byte단위로 길이를 제한. Byte단위의 경우 아시아 언어와 일부 기호가 2단위로 계산

#### 8.1.2.Type Validation

입력될 정보의 Type이나 Format을 지정하는 방법이다.

##### 8.1.2.1.Grid

Property	Description
Edit	<ul style="list-style-type: none"><li>• 해당 Cell의 입력모드 Type을 지정</li><li>• upper: 대문자만 입력됨</li><li>• lower: 소문자만 입력됨</li><li>• integer: 정수만 입력됨</li></ul>

Property	Description
	<ul style="list-style-type: none"> <li>mask: Mask Property에 정의된 Mask 형태에 맞는 값만 입력됨</li> </ul>
Mask	적용할 Mask 형식

### 8.1.2.2.MaskEdit

MaskEdit 컴포넌트를 이용하여 우편번호나 주민번호 등의 입력Form을 작성할 수 있다.

## 8.2.Using Script

본 샘플에서는 화면에서 사용자가 입력한 정보에 대한 Validation 처리를 일괄적으로 할 수 있도록 Script 함수를 제공한다.

### 8.2.1.Check Validity

gfnValidate()는 입력 파라미터로 지정된 컴포넌트 하위에 속한 모든 컴포넌트들의 Validation을 처리한다. 체크기준은 각 UI 컴포넌트의 **UserData** 속성에 입력된 Check List내용으로 판단한다. Grid의 경우 UserData 속성이 아닌, Grid의 BindDataset 에서 각 컬럼의 **MapValue** 속성에 입력된 Check List내용을 이용한다.

#### 8.2.1.1.Syntax

**gfnValidate(objTarget)**

예) if(gfnValidate(this)) {…}

#### 8.2.1.2.Parameters

Property	Description
objTarget	Validation을 수행할 Base컴포넌트 (예: Form, Div)

### 8.2.2.Check List for Validation

Validation 처리를 위한 Check List 작성 시 각 항목은 comma(,)로 구분한다. 또한 오류메세지 출력 시 사용될 title/titleObj/titleId 중 하나는 반드시 가장 앞에 기술되어야 한다.

#### 8.2.2.1.General Components

Grid를 제외한 일반적인 UI 컴포넌트의 경우 아래와 같은 항목으로 구성된 Check List를 UserData 속성에 지정한다.

Check Item	Description
title=항목명	오류메세지 출력 시, 사용될 항목명
titleObj=특정ObjID	오류메세지 출력 시, 사용될 항목명을 나타내는 Object ID
titleId=DomainID	오류메세지 출력 시, 사용될 항목명을 나타내는 Domain ID
required	필수항목인 경우, 값이 없으면 오류메세지 출력
minLength=값	항목 값이 최소길이로 설정된 값보다 작은 길이이면 오류메세지 출력 Byte 단위가 아니므로 입력될 문자길이로 지정할 것.

Check Item	Description
	최대길이 체크는 MaxLength 속성값 설정으로 대신함. MaxLength 값을 설정하면 설정한 길이 이상으로 입력되지 않음
fromNum=값	항목값을 체크하여 최소값보다 작으면 오류메세지 출력
toNum=값	항목값을 체크하여 최대값보다 크면 오류메세지 출력
format=mail	항목값이 email 주소 형식에 적합하지 않으면 오류메세지 출력
format=url	항목값이 URL 형식에 적합하지 않으면 오류메세지 출력
format=phone	항목값이 전화번호 형식에 적합하지 않으면 오류메세지 출력
format=resno	항목값이 주민번호 형식에 적합하지 않으면 오류메세지 출력
format=date	항목값이 날짜 형식(YYYYMMDD)에 적합하지 않으면 오류메세지 출력
format=time	항목값이 시간 형식에 적합하지 않으면 오류메세지 출력
fromDate=특정ObjID	시작일로 지정된 항목의 날짜가 종료일보다 이후 날짜인 경우 오류메세지 출력

MiPlatform PID에서 UserData 속성에 Check List를 명시한 예이다.

UseIME	True
UserData	titleId=biz.user.phone,format=phone
VAlign	Middle

## 8.2.2.2.Grid Component

일반적인 UI 컴포넌트와 달리, Grid 컴포넌트에 입력되는 정보에 대한 Validation 처리를 하기 위해서는 BindDataset 각 컬럼의 MapValue 속성에 Check List를 기술해야 한다.

### Notice

- Validation 대상 Grid의 BindDataset에 해당하는 Dataset의 **UseClientLayout** 속성을 **true**로 설정할 것.
- Validation 대상 Grid의 BindDataset에 대해서는 절대 clear()를 사용하지 말 것. 필요한 경우 **clearData()**로 대체해서 사용할 것.

Check Item	Description
required	컬럼에 대한 필수항목 체크
minLength=값	컬럼에 대한 최소길이 체크 Byte 단위가 아니므로 입력될 문자길 이로 지정할 것  최대길이 체크는 Grid의 Body Cell Property인 Limit 속성값 설정으로 대신함. Limit 값을 설정하면 설정한 길이 이상으로 입력되지 않음.
fromNum=값	컬럼에 대한 최소값 체크
toNum=값	컬럼에 대한 최대값 체크
format=mail	컬럼에 대한 email 형식 체크
format=url	컬럼에 대한 URL 형식 체크
format=phone	컬럼에 대한 전화번호 형식 체크
format=resno	컬럼에 대한 주민등록번호 형식 체크
format=date	컬럼에 대한 날짜(YYYYMMDD) 형식 체크
format=time	컬럼에 대한 시간 형식 체크
fromDate=특정COLID	컬럼에 대한 기간 체크

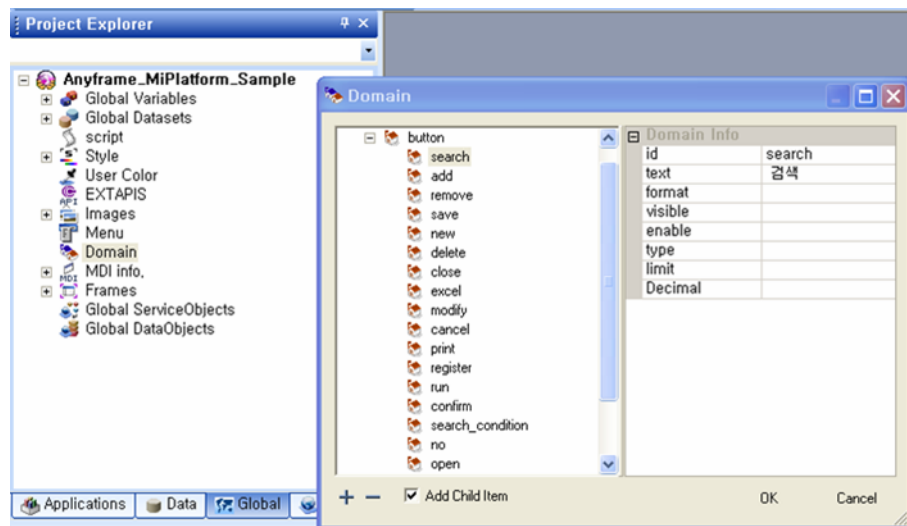
Grid의 BindDataset에서 컬럼의 MapValue 속성에 Check List를 명시한 예이다.

Columns:						
No	ID	Type	Size	Sum	Sum-Text	MapValue
1	_chk	CHAR	1			
2	TITLE	STRING	256			required
3	CONTENTS	STRING	256			minLength=1
4	REG_ID	STRING	256			
5	REG_DATE	STRING	256			
6	COMMUNITY...	STRING	256			
7	POST_ID	STRING	256			

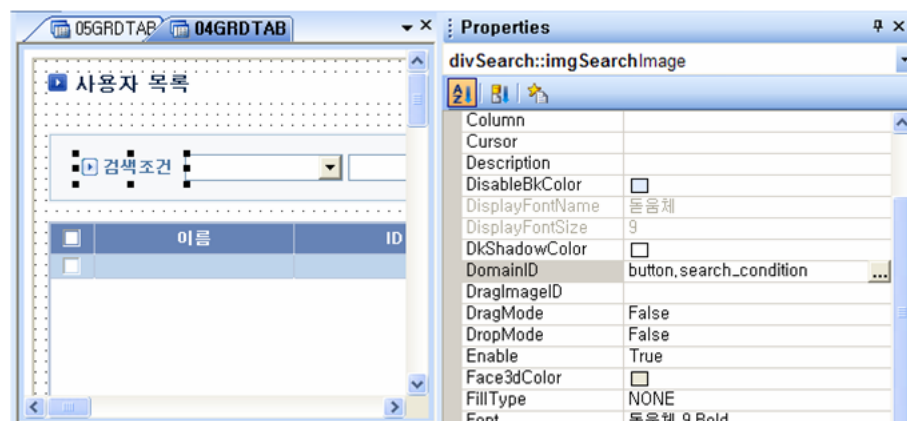
## 9. Internationalization (i18n)

### 9.1. Domain

MiPlatform에서 Domain이란 어플리케이션 전체에 적용할 수 있는 공통적인 지침 또는 속성을 정의할 수 있는 기능이다. Domain은 MiPlatform PID의 Global 탭에서 편집할 수 있고, 별도의 Domain File (/miplatform/domains/domain\_XX.xml)로 저장된다.



본 샘플에서 Static이나 Image 컴포넌트를 통해 화면 상에 표시되는 Label항목은 Domain 기능을 이용하여 다국어 처리를 수행한다.



Domain File은 domain\_KO.xml, domain\_EN.xml 두 개의 파일로 이뤄져 있으며, 다음과 같은 구조로 내부가 구성되어 있다.

- domain\_KO.xml

```
<?xml version="1.0" encoding="utf-8"?>
<domain>
  <item id="button" text="버튼">
    <item id="search" text="검색"/>
    <item id="add" text="추가"/>
    <item id="remove" text="삭제"/>
    <item id="save" text="저장"/>
    <item id="new" text="신규"/>
  </item>
</domain>
```



- domain\_EN.xml

```
<?xml version="1.0" encoding="utf-8"?>
<domain>
  <item id="button" text="Button">
    <item id="search" text="Search"/>
    <item id="add" text="Add"/>
    <item id="remove" text="Remove"/>
    <item id="save" text="Save"/>
    <item id="new" text="New"/>
  </item>
</domain>
```

위 XML 내용 중, 'id'에 해당하는 값이, 컴포넌트의 DomainID에 등록된 내용과 매칭되어서 'text'에 정의된 내용이 출력된다.

로그인 시, layouts/LOGIN.xml의 Script에서, 서버에서 전달한 Language 값을 'gvLanguage'에 저장 하고 해당 Language에 맞는 Domain 파일을 로딩하도록 구현하였다.