Anyframe Excel Plugin

Version 1.1.0

I. Introduction 1
II. Excel 2
1. Excel Sheet Configuration XML 3
1.1. Excel Sheet Configuration XML 위치 3
1.2. Excel Sheet Configuration XML 구성 3
1.3. Excel Sheet Configuration XML 속성 4
2. Excel Download 5
2.1. Download JSP 5
2.2. ExcelController의 excelDownload 6
3. Excel Upload 8
3.1. Upload JSP 8
3.2. ExcelController의 excelUpload 9

I.Introduction

Anyframe excel plugin은 Query Service를 이용해 Microsoft Excel파일을 업, 다운로드 하기 위한 관련 라이브러리와 샘플로 구성되어 있다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 excel plugin을 설치한다.

```
mvn anyframe:install -Dname=excel
```

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

Plugin Name	Version Range
query [http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.6.0/reference/htmlsingle/query.html]	2.0.0 > * > 1.4.0

II.Excel

Apache POI(Poor Obfuscation Implementation) - the JAVA API for Microsoft Documents 는 Microsoft의 OLE2기반의 파일 형식을 JAVA로 개발하기 위한 Apache Software Foundation의 프로젝트 중 하나이다. Anyframe excel plugin은 POI의 서브 프로젝트인 HSSF와 Query Service를 이용해 조회된 결과를 Excel파일로 다운받고, Excel파일의 데이터를 Database에 저장하기 위한 라이브러리와 샘플로 구성되어 있다.

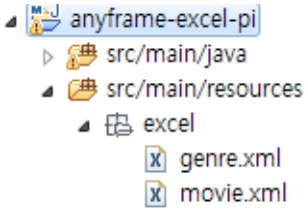
1.Excel Sheet Configuration XML

Excel Sheet Configuration XML은 Excel파일을 업로드, 다운로드 할 때 필요한 설정들을 정의하는 파일이다.

Excel 파일의 Sheet Style을 정의하기 위해서 Excel Sheet Configuration XML 파일을 작성해야 한다. Excel Sheet Configuration XML파일은 DB컬럼과 Excel파일의 해더와의 매핑, 컬럼 너비, 컬럼의 데이터 타입, 마스크 등을 정의한다.

1.1.Excel Sheet Configuration XML 위치

Excel Sheet Configuration XML 아래와 같이 src/main/resources/excel/폴더 아래에 위치한다.



1.2.Excel Sheet Configuration XML 구성

Excel Sheet Configuration XML의 최상위 Tag는 <COLUMNS>이고 하위 Tag로 복수의 <COLUMN>를 갖는다. 다음은 Excel Sheet Configuration XML의 예이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<COLUMNS>
    <COLUMN>
        <COLUMN_NAME>ID</COLUMN_NAME>
        <FIELD_NAME>movieId</FIELD_NAME>
        <COLUMN_WIDTH>20</COLUMN_WIDTH>
        <REQUIRED>true</REQUIRED>
        </COLUMN>
    <COLUMN>
        <COLUMN_NAME>장르</COLUMN_NAME>
        <FIELD_NAME>genreName</FIELD_NAME>
        <COLUMN_WIDTH>20</COLUMN_WIDTH>
    </COLUMN>
    <COLUMN>
        <COLUMN_NAME>제목</COLUMN_NAME>
        <FIELD_NAME>title</FIELD_NAME>
        <COLUMN_WIDTH>50</COLUMN_WIDTH>
    </COLUMN>
    <COLUMN>
        <COLUMN_NAME>감독</COLUMN_NAME>
        <FIELD_NAME>director</FIELD_NAME>
        <COLUMN_WIDTH>10</COLUMN_WIDTH>
    </COLUMN>
    <COLUMN>
        <COLUMN_NAME>배우</COLUMN_NAME>
        <FIELD_NAME>actors</FIELD_NAME>
        <COLUMN_WIDTH>10</COLUMN_WIDTH>
    </COLUMN>
    <COLUMN>
        <COLUMN_NAME>티켓 가격</COLUMN_NAME>
        <FIELD_NAME>ticketPrice</FIELD_NAME>
```

```
        <COLUMN_WIDTH>10</COLUMN_WIDTH>
    </COLUMN>
    <COLUMN>
        <COLUMN_NAME>개봉일자</COLUMN_NAME>
        <FIELD_NAME>releaseDate</FIELD_NAME>
        <COLUMN_WIDTH>10</COLUMN_WIDTH>
        <COLUMN_TYPE>Date</COLUMN_TYPE>
        <MASK>yyyy-MM-dd</MASK>
    </COLUMN>
</COLUMNS>
```

1.3.Excel Sheet Configuration XML 속성

Excel Sheet Configuration XML의 COLUMN의 속성 값은 다음과 같다.

Attribute	Description
COLUMN_NAME	Excel 파일에 출력 될 해더이름
FIELD_NAME	Database의 필드 이름(Query Mapping XML의 mappingStyle값에 따름)
COLUMN_WIDTH	Excel파일에 될 컬럼의 너비
COLUMN_TYPE	컬럼 타입(Default : String)
REQUIRED	not null 필드일 때 true

2.Excel Download

Excel Download는 QueryService의 조회 결과를 Apache POI를 이용해 Excel파일로 만드는 기능이다. Anyframe excel plugin에서는 Excel Download를 하기위해 ExcelController, ExcelInfoHandler, ExcelService 등의 클래스를 제공한다. 각 각의 클래스는 다음과 같은 역할을 한다.

- **ExcelService** : QueryService를 이용해 특정 Query를 실행하고 결과값을 리턴한다.
- **ExcelInfoHandler** : Excel Sheet Configuration XML파일을 읽어 컬럼 정보를 생성한다.
- **ExcelController** : ExcelService를 통해 조회 된 데이터를 HSSF를 이용해 Excel파일을 만든다.

2.1.Download JSP

다음은 Anyframe excel plugin에서 샘플로 제공하고 있는 화면이다.

화면 하단의 Excel Down 버튼을 클릭하면 jsp에서 지정한 query id를 이용해 쿼리가 실행되고 그 결과가 Excel파일로 다운로드 된다. 아래는 Excel Download를 하기 위해 필요한 변수값을 jsp파일에 세팅한 예이다.

```
<form:form method="post" name="excelDownloadForm">
    <input name="queryId" type="hidden" value="" />
    <input name="title" type="hidden" value="" />
    <input name="nowPlaying" type="hidden" value="Y" />
    <input name="fileName" type="hidden" value="movie" />
    <input name="columnInfoFile" type="hidden" value="movie" />
</form:form>
```

queryId, columnInfoFile, fileName은 Excel Download기능을 위한 필수 변수이다.

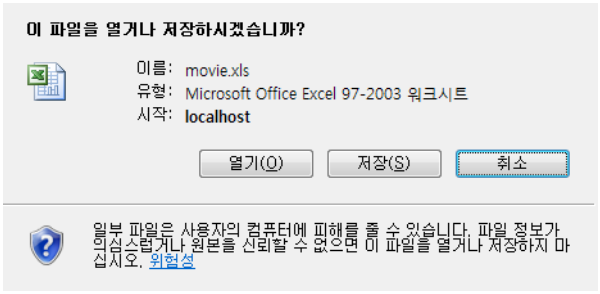
- **queryId** : QueryService를 이용해 실행할 Query Mapping XML 파일의 Query ID이다.
- **fileName** : Download할 Excel파일의 파일명이다.
- **columnInfoFile** : Excel Sheet Configuration XML파일명이다.

title, nowPlaying과 같은 변수는 Query를 실행할 때 필요한 변수들이다(조회조건)

Excel Download 버튼을 클릭했을 때 실행되는 javascript는 다음과 같다.

```
function fncExcelDownload() {
    document.excelDownloadForm.queryId.value="excel.findMovieList";
    document.excelDownloadForm.action="<c:url value='/excelDownload.do?
method=excelDownload'/>";
    document.excelDownloadForm.submit();
}
```

Excel Download의 URL은 /excelDownload.do?method=excelDownload이다. Excel Down버튼을 클릭해 Query Id가 excel.findMovieList인 쿼리를 실행하고 결과가 정상적으로 Excel파일로 만들어졌다면 아래와 같은 화면을 볼 수 있다.



Anyframe Excel Plugin은 약 65000 ROW에 대한 Download 테스트가 완료 되었다.

아래는 Download된 Excel 파일의 일부이다.

	A	B	C	D	E	F	G
1	ID	장르	제목	감독	배우	티켓 가격	개봉일자
2	MV-00001	Adventure	Alice in Wonderland	Tim Burton	Johnny Depp	8000.00	2010-03-04
3	MV-00002	Sci-Fi	Avatar	ımes Camer	ourney Wea	7000.00	2010-02-16
4	MV-00003	Action	Green Zone	aul Greengra	Yigal Naor	7000.00	2010-03-25
5	MV-00004	Drama	Remember Me	Allen Coulte	Caitlyn Runc	8000.00	2010-03-12
6	MV-00005	Comedy	She is Out of My League	m Field Smi	Jay Baruche	7500.00	2010-03-12
7	MV-00006	Crime	Shutter Island	artin Scorse	onardo DiCa	8000.00	2010-03-18

2.2.ExcelController의 excelDownload

ExcelController의 excelDownload는 ExcelService의 download메소드를 호출하고 그 결과값을 HSSF를 이용해 Excel파일을 생성한다. 아래는 ExcelController excelDownload 메소드 소스코드의 일부이다.

```
@RequestMapping(params = "method=excelDownload")
public void excelDownload(HttpServletRequest request, HttpServletResponse response) throws
 Exception {
//중 략
        List<Map<String, Object>> resultList = excelService.download(searchMap);

//중 략

        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet sheet = workbook.createSheet(fileName);

        OutputStream fileOut = null;
        try {
            fileOut = response.getOutputStream();
            HSSFRow row;
            HSSFCellStyle style = workbook.createCellStyle();
            style.setAlignment(HSSFCellStyle.ALIGN_CENTER);

            HSSFCell cell;

            row = sheet.createRow(0);

            List<ColumnInfo> columnInfoList = getColumnInfo(request, columnInfoFile);

            int columnCount = columnInfoList.size();
            String[] header = new String[columnCount];
            int[] cellwidth = new int[columnCount];
            String[] fieldName = new String[columnCount];
            String[] columnType = new String[columnCount];
            String[] mask = new String[columnCount];
```

```
            ColumnInfo columnInfo = null;
            short width = 265;
//중략
    }
}
```

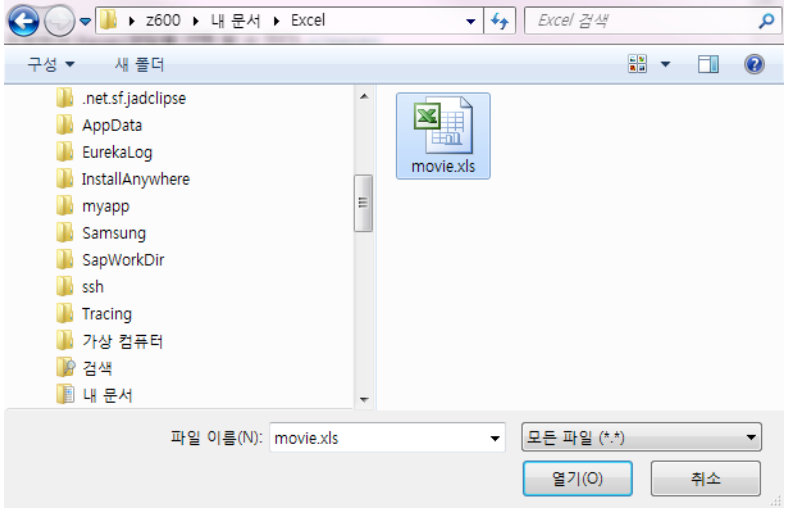
3.Excel Upload

Excel Upload는 Excel Download와는 반대로 Excel파일의 데이터를 QueryService를 이용해 Database에 저장하는 기능이다.

3.1.Upload JSP

다음은 Anyframe excel plugin에서 샘플로 제공하고 있는 화면이다.

화면 하단의 파일 찾기 버튼을 누르면 아래와 같이 로컬에서 Excel파일을 선택 할 수 있다.



Excel 파일을 Upload 하기 위해서는 Download와 마찬가지로 jsp에서 서버로 전송해야할 변수 값들이 있다. 아래는 Upload하기 위해 설정된 jsp파일 내의 변수들이다. Upload를 위해 사용되는 form tag의 enctype은 반드시 multipart/form-data이다.

```
<form:form method="post" name="excelUploadForm" enctype="multipart/form-data">
    <input type="file" name="excelFile" width="0">
    <input name="queryId" type="hidden" value="" />
    <input name="columnInfoFile" type="hidden" value="movie" />
    <input name="resultPage" type="hidden" value="redirect:/excelMovieFinder.do?
method=list" />
</form:form>
```

- **queryId** : QueryService를 이용해 실행할 Query Mapping XML 파일의 Query ID이다.(필수)
- **file** : Upload할 Excel파일이다.(필수)
- **columnInfoFile** : Excel Sheet Configuration XML파일명이다.(필수)
- **resultPage** : Upload 후 이동할 페이지이다.(필수)

- **startRow** : Excel 파일의 데이터중 일부만 저장하고 싶을 때 시작 행번호이다.(선택)
- **endRow** : Excel 파일의 데이터중 일부만 저장하고 싶을 때 마지막 행번호이다.(선택)

Excel Upload 버튼을 클릭했을 때 실행되는 javascript는 다음과 같다.

```
function fncExcelUpload() {
    document.excelUploadForm.queryId.value="excel.excelMovieInsert";

    var filePath = document.excelUploadForm.excelFile.value;

    if(filePath.indexOf('.xls') == -1){
        alert("Excel 파일이 아닙니다.");
    return;
    }
    document.excelUploadForm.action="<c:url value='/excelDownload.do?method=excelUpload'/>";
    document.excelUploadForm.submit();
}
```

Excel Upload의 URL은 /excelDownload.do?method=excelUpload이다. Excel 파일의 데이터를 DB에 저장하기 위해 실행되는 쿼리의 ID는 excel.excelMovieInsert이다.

3.2.ExcelController의 excelUpload

ExcelController의excelUpload메소드 전송 된 Excel파일을 분석해 데이터를 HashMap형태로 만든 다음 ExcelService의 upload메소드를 호출해 DB에 데이터를 저장한다. 아래는 ExcelController excelUpload 메소드 소스코드의 일부이다.

```
@RequestMapping(params = "method=excelUpload")
public String excelUpload(HttpServletRequest request, @RequestParam(value = "excelFile",
 required = false) MultipartFile excelFile) throws Exception {

    Map<String, Object> paramMap = bindRequestToMap(request);

    int startRow = 1;
//저장을 시작할 row값, 설정이 없을 경우 1부터 시작, 0은 Heder정보
    if( paramMap.get("startRow") != null ){
    startRow = Integer.parseInt(paramMap.get("startRow").toString());
    }

    POIFSFileSystem fs = null;
    try{
        fs = new POIFSFileSystem(excelFile.getInputStream());
    }catch(Exception e){
        e.printStackTrace();
    }

    HSSFWorkbook workbook = new HSSFWorkbook(fs);
    int sheetNum = workbook.getNumberOfSheets();

//시트는 한개로 제약
    HSSFSheet sheet = workbook.getSheetAt(0);

//저장한 데이터의 마지막 줄, 설정이 없을 경우는 시트 전체의 Row
    int endRow = 0;
    if( paramMap.get("endRow") != null ){
        endRow = (Integer)paramMap.get("endRow");
    }else{
        endRow = sheet.getLastRowNum();
    }
```

```
    int rowCount = endRow;

    String columnInfoFile = "";

//ColumnInfo파일에서 ColumnName과 FieldName값을 HashMap으로 추출
    if( paramMap.get("columnInfoFile") == null ){
        throw new Exception("fail to get column info xml file name");
    }else{
        columnInfoFile = paramMap.get("columnInfoFile").toString();
    }

    List<ColumnInfo> columnInfoList = getColumnInfo(request, columnInfoFile);

    Map<String, String> fieldMap = new HashMap<String, String>();
    Map<String, Boolean> requiredMap = new HashMap<String, Boolean>();
    for ( int i = 0 ; i < columnInfoList.size() ; i ++ ){
        ColumnInfo columnInfo = columnInfoList.get(i);
        fieldMap.put(columnInfo.getColumnName(), columnInfo.getFieldName());
        requiredMap.put(columnInfo.getColumnName(), columnInfo.isRequired());
    }

    String[] header = null;
    String[] filedName = null;
    Boolean[] requiredField = null;

//컬럼 헤더 정보 추출
    HSSFRow row = sheet.getRow(0);
    if( row != null ){
        int headerCount = row.getPhysicalNumberOfCells();
        header = new String[headerCount];
        filedName= new String[headerCount];
        requiredField = new Boolean[headerCount];

        for( int i = 0 ; i < headerCount ; i ++ ){
            HSSFCell cell = row.getCell(i);
            if (cell != null && cell.getCellType() != HSSFCell.CELL_TYPE_BLANK) {
            header[i] = getCellValue(cell);
            filedName[i] = fieldMap.get(header[i]);
            requiredField[i] = requiredMap.get(header[i]);
        }
    }
}

    List<ListOrderedMap> insertList = new ArrayList<ListOrderedMap>();
    ListOrderedMap insertMap = null;

//컬럼 값 추출
    for( int i = startRow ; i <= endRow ; i ++ ){
        boolean isInsertRow = true;
        row = sheet.getRow(i);
        if( row != null ){
            insertMap = new ListOrderedMap();
            for( int j = 0 ; j < filedName.length ; j ++ ){
                HSSFCell cell = row.getCell(j);
                if( cell != null ){
                    if( !requiredField[j] ){
                        insertMap.put(filedName[j], getCellValue(cell));
                    }else if( cell.getCellType() != HSSFCell.CELL_TYPE_BLANK ){
                        insertMap.put(filedName[j], getCellValue(cell));
                    }else{
                        isInsertRow = false;
                        break;
```

```
                    }
                }
            }
            if(isInsertRow){
            insertList.add(insertMap);
        }
    }
    }
//ExcelService의 upload method호출
    excelService.upload(paramMap, insertList);
    return paramMap.get("resultPage").toString();
}
```