JAVA Excel 파일 입출력 (Poi Library 활용)

이학재

Java에서 Excel 파일의 입출력을 위해서는 Poi Library를 필요로 합니다.

Java JDK 7에서는 4 버전 이상의 poi 사용이 불가합니다.

따라서 아래의 링크에서 3버전대의 poi 다운로드가 필요합니다.

https://archive.apache.org/dist/poi/release/bin/'

다운로드 후 Eclipse – Build Path 설정에서 .jar 파일을 추가합니다.

파일을 입/출력하여 데이터를 저장하기 위한 클래스가 필요합니다.

여기서 예시로 들어볼 클래스는 NotifyVO입니다.

Notify.java

```
public class NotifyVO {
	private int seq;
	private String title;
	private String contents;
	private String date;
	private int readView;

	public int getSeq() {
		return seq;
	}
	public String getTitle() {
		return title;
	}
	public String getContents() {
		return contents;
	}
```

```
	public String getDate() {
		return date;
	}
	public int getReadView() {
		return readView;
	}

	public void setSeq(int seq) {
		this.seq = seq;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public void setContents(String contents) {
		this.contents = contents;
	}
	public void setDate(String date) {
		this.date = date;
	}
	public void setReadView(int readView) {
		this.readView = readView;
	}
}
```

파일 입력

파일 입력을 먼저 해보겠습니다.

NotifyVO 클래스의 인스턴스에 데이터를 주입하기 위한 Excel 파일을 준비합니다.

프로젝트 폴더 내 db에 Database.xlsx라는 이름으로 저장하였습니다.

.₩db₩Database.xlsx

seq	title	contents	date	readView
1	배스킨라빈스 401 번째 지점 오픈	401 번째 지점 오픈을 축하합니다 ~~~	2020-04-01	10831
2	31 일의 이벤트	배스킨라빈스는 매 31 일마다 패밀리 → 하프갤런 사이즈업 이벤트를 진행합니다.	2020-05-17	6678
3	빼빼로데이 기념 행사 안내	빼빼로데이를 맞아 패밀리 이상 사이즈 구매 시 빼빼로 1:1 지급 이벤트를 진행합니다. (일부 매장 제외)	2020-11-11	8376
4	새해 복 많이 받으세요.	소의 해를 맞아 인사드립니다. 새해 복 밚이 받으세요.	2021-01-01	678
5	[이달의 맛] 우낀소 (우유속에 끼인 소보로)	우유 아이스크림과 소보로 아이스크림 속에 소보로 크럼블이 쏙쏙!	2021-01-01	372

Excel 파일의 seq와 readView는 숫자, 나머지 요소들은 문자열로 구성되어 있습니다.

이를 입출력하기 위한 Poi 클래스를 생성하고, 메서드를 생성합니다.

Poi.java

```
public class Poi {
	public static List<NotifyVO> readNotifyList() {
		List<NotifyVO> notifyList = new ArrayList<>();
		try {
			FileInputStream file = new
FileInputStream("db\\Database.xlsx");
			XSSFWorkbook workbook = new XSSFWorkbook(file);
			XSSFSheet sheet =
workbook.getSheet("notifyList");
			for (int rowindex = 1; rowindex <
sheet.getPhysicalNumberOfRows(); rowindex++) {

				XSSFRow row = sheet.getRow(rowindex);
				if (row != null) {
					NotifyVO notify = new NotifyVO();

	notify.setSeq((int)row.getCell(0).getNumericCellValue());

	notify.setTitle(row.getCell(1).getStringCellValue());

	notify.setContents(row.getCell(2).getStringCellValue());

	notify.setDate(row.getCell(3).getStringCellValue());

	notify.setReadView((int)row.getCell(4).getNumericCellValue()
);
					notifyList.add(notify);
				}
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
		return notifyList;
	}
}
```

다음은 각 라인에 대한 설명입니다.

```
List<NotifyVO> notifyList = new ArrayList<>();
```

Excel에서 파일을 불러온 후 자료를 불러올 리스트를 생성합니다.

```
FileInputStream file = new FileInputStream("db\\Database.xlsx");
```

파일을 불러옵니다.

```
XSSFWorkbook workbook = new XSSFWorkbook(file);
```
Poi는 Excel 파일을 XSSFWorkbook 인스턴스로 불러옵니다.

```
XSSFSheet sheet = workbook.getSheet("notifyList");
```
Excel 내 "notifyList"라는 이름을 가진 Sheet를 불러옵니다.

만약, sheet의 번호로 불러오고 싶은 경우, .getSheetAt(int)를 사용할 수 있습니다.

```
for (int rowindex = 1; rowindex < sheet.getPhysicalNumberOfRows();
rowindex++)
```
각 라인별로 데이터를 불러오기 위해 rowindex를 가져왔습니다.

불러오는 최대 라인의 수는 sheet.getPhysicalNumberOfRows() 부분을 통해 총 줄 수를 알아낼 수 있습니다.

rowindex가 0이 아닌 1로 시작하는 이유는 제목줄을 제외하기 위함입니다.

```
XSSFRow row = sheet.getRow(rowindex);
```
rowindex와 일치하는 라인 하나를 불러옵니다.

```
if (row != null) {
```
row가 null일 경우, 해당 라인에 대해 아래의 부분을 추가하지 않기 위해 사용합니다.

```
NotifyVO notify = new NotifyVO();
notify.setSeq((int)row.getCell(0).getNumericCellValue());
notify.setTitle(row.getCell(1).getStringCellValue());
notify.setContents(row.getCell(2).getStringCellValue());
notify.setDate(row.getCell(3).getStringCellValue());
notify.setReadView((int)row.getCell(4).getNumericCellValue());
notifyList.add(notify);
```
위에서 생성한 notifyList에 데이터를 저장하기 위해 notify 객체를 생성하고, 데이터를 하나씩 담습니다.

숫자의 경우, .getNumericCellValue() 메서드를 통해 값을 불러오며, 불러온 데이터는 double형이기때문에, 원하는 자료형인 int로 형변환합니다.

문자는 .getStringCellValue() 메서드를 통해 값을 불러옵니다.

이외 .getBooleanCellValue() 메서드를 통해 boolean형의 값을 불러올 수 있습니다.

다른 자료형은 https://poi.apache.org/apidocs/dev/org/apache/poi/ss/usermodel/Cell.html에서 확인하실 수 있습니다.

```
file.close();
```

작업 완료 후 FileInputStream을 닫습니다.

```
return notifyList;
```

생성된 notifyList를 반환합니다.

try – catch의 경우, 파일 입력 과정 및 Excel에서 발생할 수 있는 오류에 대비하여 생성하거나, throw로 다른 클래스로 던질 수 있습니다.

파일 출력

다음은 다시 입력 받았던 파일을 통해 작업 후, 변경된 값을 다시 Excel 파일에 저장하고자 합니다.

Poi.java

```
public class Poi {
	boolean writeNotifyAsExcel(List<NotifyVO> notifyList) {
		try {
			FileOutputStream file = new
FileOutputStream("db\\Database.xlsx");
			XSSFWorkbook workbook = new XSSFWorkbook();
			XSSFSheet sheet =
workbook.createSheet("notifyList");
			XSSFRow curRow = sheet.createRow(0);
			curRow.createCell(0).setCellValue("seq");
			curRow.createCell(1).setCellValue("title");
			curRow.createCell(2).setCellValue("contents");
			curRow.createCell(3).setCellValue("date");
			curRow.createCell(4).setCellValue("readView");
			for (int i = 1; i < notifyList.size() + 1; i++)
{
				curRow = sheet.createRow(i);

	curRow.createCell(0).setCellValue(notifyList.get(i -
1).getSeq());

	curRow.createCell(1).setCellValue(notifyList.get(i -
1).getTitle());

	curRow.createCell(2).setCellValue(notifyList.get(i -
1).getContents());

	curRow.createCell(3).setCellValue(notifyList.get(i -
1).getDate());

	curRow.createCell(4).setCellValue(notifyList.get(i -
1).getReadView());
			}
			workbook.write(file);
			file.close();
			return true;
		} catch (Exception e) {
```

```
                    e.printStackTrace();
                }
                return false;
        }
}
```

다음은 각 줄에 대한 설명입니다.

```
boolean writeNotifyAsExcel(List<NotifyVO> notifyList)
```
이 메서드는 List<NotifyVO> notifyList를 입력 받아 그 결과로 boolean형을 반환할 것입니다.

```
FileOutputStream file = new FileOutputStream("db\\Database.xlsx");
```
파일을 저장할 장소를 지정합니다.

여기서는 기존 파일을 업데이트하려고 하였기에, 입력 받은 파일과 동일 경로로 지정하였습니다.

```
XSSFWorkbook workbook = new XSSFWorkbook();
```
Poi에서는 Excel 파일 저장을 위해 XSSFWorkbook 인스턴스를 필요로 합니다.

```
XSSFSheet sheet = workbook.createSheet("notifyList");
```
파일 내에 notifyList라는 이름을 가진 sheet를 추가합니다.

```
XSSFRow curRow = sheet.createRow(0);
curRow.createCell(0).setCellValue("seq");
curRow.createCell(1).setCellValue("title");
curRow.createCell(2).setCellValue("contents");
curRow.createCell(3).setCellValue("date");
curRow.createCell(4).setCellValue("readView");
```
0번 row를 작성합니다.

여기서는 제목을 작성하였습니다.

```
for (int i = 1; i < notifyList.size() + 1; i++) {
      curRow = sheet.createRow(i);
      curRow.createCell(0).setCellValue(notifyList.get(i -
1).getSeq());
```

```
      curRow.createCell(1).setCellValue(notifyList.get(i -
1).getTitle());
      curRow.createCell(2).setCellValue(notifyList.get(i -
1).getContents());
      curRow.createCell(3).setCellValue(notifyList.get(i -
1).getDate());
      curRow.createCell(4).setCellValue(notifyList.get(i -
1).getReadView());
}
```

매개변수로 전달받은 notifyList로부터 정보를 하나씩 꺼내서 cell에 삽입합니다.

제목 라인을 제외하여 시작하기 위해 row를 1부터 추가하였으며, notifyList의 길이만큼 수행하기 위해, notifyList.size()보다 1 큰 값까지 반복되도록 지정하였습니다.

```
workbook.write(file);
```

workbook에 생성된 값들을 file에 저장합니다.

```
file.close();
```

FileOutputStream을 닫습니다.