JAVA와 DB연동

- 로드
  - JDBC드라이버 연결
- 연결
  - DataBase 연결
- 실행
  - Sql문장 실행
- 닫기

JAVA와 DB연동

❑ 로드

Class.forName("연결하고자하는DB드라이버클래스이름");

❑ DB드라이버클래스이름

Oracle : oracle.jdbc.driver.OracleDriver

Ms_Sql : sun.jdbc.odbc.JdbcOdbcDriver

My_Sql : org.git.mm.mysql.Driver

JAVA와 DB연동

❑ DB연결

```
Connection con = DriverManager.getConnection
                 ("JDBCURL", "사용자명", "패스워드")
```

❑ JDBCURL

- jdbc:oracle:thin:@localhost:1521:ORCL
- jdbc:odbc:odbc설정을통해만든db원본명
- jdbc:mysql://localhost:3306/db명

JAVA와 DB연동

❑ Sql문 실행

```
Statement stm = con.createStatement();
```

```
stm.executeUpdate("Insert into ....................");
```

»Update문, delete문등에서 사용

```
stm.executeQuery("Select ..........................");
ResultSet rs= stm.executeQuery("Select.....");
rs.next();      -> data읽기 위해 커서이동
rs.getString("컬럼명");  -> 내용 읽어오기( get이름("컬럼명") )
rs.next(); ..............
```

PreparedStatement

- Sql문장 사용 할 때 statement사용과 비슷
- 하나의 객체를 가지고 여러 번 sql문장 실행
  - (객체 여러 번 생성하지 않는다)

```
PreparedStatement ps= con.prepareStatement
                    (“Insert into member(id,name) values(? , ? )” );
    ps.setString(1, ”jang”);
    ps.setString(2, ”장희정”);
    ps.executeUpdate(); -> insert or update문장
    ps.executeQuery(); -> select문장
    ps.close();
```

ResultSetMetaData

ResultSetMetaData 객체는 ResultSet 으로 얻어온 레코드들의 정보에 해당하는 컬럼의 정보들을 제공함

메소드	설명
getColumnCount()	ResultSet에 저장되어 있는 테이블의 컬럼수를 반환함
getColumnLabel(int column)	해당 번호 컬럼의 레이블을 반환함
getColumnName(int column)	해당 번호 컬럼의 이름을 반환함
getColumnType(int column)	해당 범호 컬럼의 데이터 타입을 int형으로 반환함
getColumnTypeName(int column)	해당 번호 컬럼의 데이터 타입을 String형으로 반환함

ResultSet의 커서 자유롭게 움직이기

ResultSet 객체는 여러 레코드들이 저장되어 있는 객체임 여기서 자신이 원하는 레코드에 접근할 수 있음

메소드	설명
absolute(int row)	지정한 위치로 커서를 이동함
beforeFirst()	커서를 처음 위치로 이동함
afterLast()	커서를 마지막 위치로 이동함
first()	처음 레코드가 존재하는 행으로 이동함
last()	마지막 레코드가 존재하는 행으로 이동함
next()	다음 레코드 행으로 이동함
previous()	이전 레코드 행으로 이동함

◆커서 옵션

```
ps = con.prepareStatement(sql ,
                          ResultSet.TYPE_SCROLL_SENSITIVE,
                          ResultSet.CONCUR_UPDATABLE);
```

resultSet 옵션값을 사용하지 않으면 first(), last() 등 커서 이동관련 메소드를 사용할 수 없음

메소드	설명
TYPE_FORWARD_ONLY	커서 이동을 다음 레코드로만 이동되도록 함
TYPE_SCROLL_SENSITIVE	커서 이동을 자유롭게 하고 업데이트 내용을 반영함
TYPE_SCROLL_INSENSITIVE	커서 이동을 자유롭게 하고 업데이트 내용을 반영하지 않음
CONCUR_UPDATABLE	데이터 변경이 가능하도록 함
CONCUR_READ_ONLY	데이터 변경이 불가능하도록 함

◆프로시져 실행

CallableStatement cs =
 con.prepareCall("{ call 프로시져이름(?,?,…) }");

//?의 순서대로 값 설정
In인 경우
   cs.setString(int parameterIndex, String value);
   cs.setInt(int parameterIndex, int value);

Out인 경우
 cs.registerOutParameter
      (int parameterIndex, int OracleTypes.*종류);*

프로시져 실행

-프로시져 실행문장 insert, update, delete 인 경우

```
int re =  cs.executeUpdate();
```

-프로시져 실행문장 select인 경우

```
cs.executeQuery();
ResultSet rs= (ResultSet)cs.getObject(1); //1은 out의 순서
while(rs.next()){
   System.out.println(rs.getString(1));
   System.out.println(rs.getString(2));
 }
```