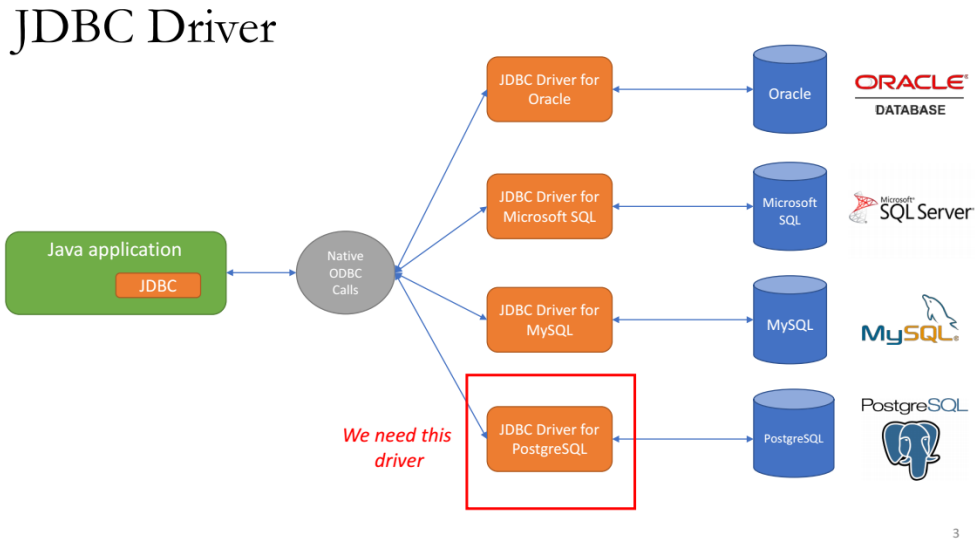
[summary] JDBC

노트북:	0050_java		
만든 날짜:	2022-08-06 오후 10:49	수정한 날짜:	2022-08-28 오전 7:46
작성자:	nomad.lab.kr@gmail.com		
URL:	about:blank#blocked		

JDBC Java DataBase Connectivity 는 자바 / JSP 프로그램 내에서 데이터베이스와
관련된 작업을 처리할 수 있도록 도와주는 자바 표준 인터페이스로, 관계형 데이터베이스 시스템에
접근하여 SQL 문을 실행하기 위한 자바 API 또는 자바 라이브러리.
JDBC API를 사용하면 DBMS의 종류에 상관없이 데이터베이스 작업을 처리할 수 있음.



JDBC API는 java.sql.* 패키지에 의해 구현되며, 이 패키지는 여러 종류의 데이터베이스에 접근 할 수 있음.
java.sql.* 패키지는 단일 API를 제공하는 클래스와 인터페이스의 집합.

1) java.sql.* 패키지 임포트.
2) JDBC 드라이버 로딩.
3) 데이터베이스 접속을 위한 Connection 객체 생성.
4) 쿼리문을 실행하기 위한 Statement / PrepareStatement / CallableStatement 객체 생성.
5) 쿼리 실행
6) 쿼리 실행의 결과 값(int, ResultSet) 사용.

7) 사용된 객체 (ResultSet, Statement / PrepareStatement / CallableStatement. Connection) 종료.

1. Statement 객체

`boolean execute(String sql)`	주어진 SQL문 sql을 실행. select 구문을 실행하는 경우에는 true를 리턴, 그렇지 않은 경우에는 false 리턴. true를 리턴하는 경우에는 getResultSet() 메서드를 이용하여 ResultSet 객체를 생성. update, insert, delete 구문을 사용하는 경우에는 false를 리턴하고, getUpdateCount() 메서드를 이용하여 영향받은 행의 갯수를 알아낼 수 있음.
`ResultSet executeQuery(String sql)`	select 구문을 실행할 때 사용.
`int executeUpdate(String sql)`	select를 제외한 나머지 insert, create, update, delete 구문을 실행할 때 사용. 이 때 영향을 받은 행의 개수를 리턴.
`ResultSet getResultSet()`	현재 SQL 구문을 실행한 결과를 리턴. select 구문을 실행했을 경우에만 유효.
`int getUpdateCount()`	현재 SQL 구문의 실행으로 영향을 받은 행의 개수를 리턴. select를 제외한 나머지 구문에서만 유효.

2. ResultSet 객체

Statement 객체의 getResultSet(), executeQuery() 메서드가 리턴하는 객체로서 select 구문 실행 결과를 다룰 때 사용.
select 구문을 실행하여 ResultSet 객체가 생성되면 커서 cursor 가 만들어지고 select 구문 실행 결과를 가르킴.

`boolean next()`	커서를 다음 행으로 이동. ResultSet 객체가 처음 생성된 직후에 next() 메서드를 한 번 호출해야 첫 번째 행을 커서가 가르키게 됨. 성공적으로 커서가 이동하면 true를 리턴하고 더 이상 결과가 없어서 커서를 이동시킬 수 없으면 false를 리턴.
`boolean previous()`	커서를 이전 행으로 이동. 성공적으로 커서가 이동하면 true를 리턴하고 더 이상 결과가 없어서 커서를 이동시킬 수 없으면 false

	를 리턴.
`Statement getStatement()`	현재 ResultSet을 생성시킨 Statement 객체를 리턴.
`<자료형> get<자료형>(String colName),` `<자료형> get<자료형>(int colIndex)`	colName에 지정된 속성명에 해당하는 실제 데이터를 리턴. 예를 들어 속성의 데이터 자료형이 String 형과 호환되는 속성이라면 getString() 메서드를 사용. 속성 이름 대신 속성의 위치 정보를 colIndex로 줄 수 있음. 맨 앞의 속성은 '1', 두 번째 속성은 '2'와 같이 숫자로 속성의 위치를 지정.

3. PreparedStatement 객체

Statement 객체의 execute계열 메서드는 모두 SQL 문을 컴파일하고 바로 수행시켜서 결과를 리턴.
PreparedStatement 객체는 SQL 문을 미리 컴파일하여 실행하기 직전의 상태로 만든 후 실제 실행은
나중에 필요에 따라 여러 번 할 수 있음.
따라서 같은 SQL 문을 여러 번 실행시켜야 하는 경우에는 Statement 객체보다 PreparedStatement 객체를
사용하는 것이 훨씬 더 효과적.

또한 PreparedStatement 객체로 생성되는 SQL 문은 마치 함수에서처럼 매개변수를 설정하여 필요에 따라 매개변수의 값을 바꿔
실행되도록 할 수 있음. 비슷한 SQL 구문을 반복적으로 실행시켜야 하는 경우에도 유용하게 사용.
SQL 문에 매개변수를 정의할 때에는 '?' 문장을 사용하고 매개변수에 값을 설정할때에는 set계열 메서드를 사용.

PreparedStatement가 제공하는 메서드는 Statement가 제공하는 메서드와 거의 같고,
PreparedStatement가 매개변수를 지원하므로 매개변수에 값을 설정하기 위한 set계열 메서드를 제공.

* DBMS와 Java의 자료형 변환.
DBMS에서의 컬럼의 자료형과 Java 자료형, 그리고 관련된 JDBC 메소드 간의 변환표.

DBMS 자료형	Java 자료형	ResultSet 메서드	PreparedStatement 메서드

CHAR	String	getString	setString
VARCHAR	String	getString	setString
DECIMAL	java.math.BigDecimal	getBigDecimal	setBigDecimal
NUMBER	java.math.BigDecimal	getBigDecimal	setBigDecimal
TINYINT	byte	getByte	setByte
SMALLINT	short	getShort	setShort
INTEGER	int	getInt	setInt
BIGINT	long	getLong	setLong
REAL	float	getFloat	setFloat
FLOAT	double	getDouble	setDouble
DOUBLE	double	getDouble	setDouble
DATE	java.sql.Date	getDate	setDate
TIME	java.sql.Time	getTime	setTime
CLOB	Clob	getClob	setClob
BLOB	Blob	getBlob	setBlob
TIMESTAMP	java.sql.Timestamp	getTimestamp	setTimestamp