

* JDBC(Java Database Connectivity) 프로그래밍

- JDBC란? 자바 프로그램에서 SQL문을 실행하여 데이터를 관리하기 위한 JAVA API입니다.
- 특징은 다양한 데이터베이스에 대해서 별도의 프로그램을 만들 필요 없이, 해당 데이터베이스의 JDBC를 이용하면 하나의 프로그램으로 데이터베이스를 관리할 수 있습니다.
- 우리는 MySQL을 사용하므로 MySQL용 JDBC를 사용합니다.

* 데이터베이스와 JAVA의 연결 순서

1. JDBC 드라이버 로드.
2. 데이터베이스 Connection 객체 생성.
3. 쿼리문 실행을 위한 Statement 객체 생성.
4. 쿼리문을 실행.
5. ResultSet 객체를 통해 쿼리문 실행 결과값을 소비.
6. Statement 객체 종료.
7. 데이터베이스 Connection 객체 종료.

* DBMS와의 통신을 위한 JDBC 드라이버

- JDBC 드라이버는 DBMS와의 통신을 담당하는 자바 클래스로서 각 DBMS마다 별도의 드라이버가 필요합니다.
- 주요 DBMS의 JDBC 드라이버 클래스
 1. MySQL: "com.mysql.jdbc.Driver"
 2. ORACLE: "oracle.jdbc.driver.OracleDriver"

* 데이터베이스 식별을 위한 JDBC URL

- 웹이 주소를 구분할 때 URL을 사용하는 것처럼 데이터베이스도 URL을 통해 데이터베이스들을 구분합니다.
- 주요 DBMS의 JDBC URL 패턴
 1. MySQL: "jdbc:mysql://호스트이름:포트번호/DB이름"
 2. ORACLE: "jdbc:oracle:thin:호스트이름:포트번호:DB이름"

* 데이터베이스 연결을 위한 Connection 객체

- JDBC를 이용해서 데이터베이스를 사용하려면 데이터베이스와 연결된 커넥션을 구해야 합니다.
- java.sql 패키지에 있는 Connection 클래스가 데이터베이스 커넥션을 지원하며 DriverManager 클래스가 제공하는 getConnection() 메서드를 사용하여 커넥션을 구할 수 있습니다.
- getConnection() 메서드에 파라미터 값으로 JDBC URL, DB 사용자 계정명, DB 사용자 암호를 전달하면 메서드는 DB와 연결된 커넥션 객체를 리턴합니다.
- 만일 제대로 객체를 생성하지 못하면 SQLException이 발생하므로 getConnection() 메서드를 사용할 때는 반드시 try ~ catch 구문으로 예외처리를 해줘야 합니다.
- Connection 객체를 다 사용한 뒤에는 반드시 close() 메서드를 호출하여 Connection 객체가 사용한 시스템 자원을 반환해야 합니다. 그렇지 않으면 시스템 자원이 불필요하게 소모되어 커넥션을 구할 수 없는 상황이 발생할 수도 있습니다.

* 쿼리문을 실행하기 위한 Statement 객체

- Connection 객체를 생성한 후에는 Connection 객체로부터 Statement를 생성하고 쿼리문을 실행할 수 있습니다.
 - Statement 객체는 Connection객체의 createStatement() 메서드를 이용하여 생성합니다.
 - Statement 객체를 사용하면 쿼리문을 실행시킬 수 있습니다.
1. executeQuery(String query):ResultSet - Select 쿼리문을 실행합니다.
 2. executeUpdate(String query):int - Insert, Update, Delete 쿼리문을 실행합니다.

* 쿼리 실행 결과 값을 읽어오는 ResultSet 객체

- Statement 객체의 executeQuery() 메서드는 Select 쿼리문의 결과를 ResultSet객체에 담아서 리턴합니다.
- 따라서 데이터 조회의 결과값을 ResultSet이 제공하는 메서드를 통해 읽어올 수 있습니다.
- ResultSet 객체가 제공하는 next() 메서드는 Select 쿼리문의 결과값의 존재 여부를 확인하는 메서드입니다.

- ResultSet 주요 메서드

1. getString(String name):String - 지정한 컬럼 값을 String으로 읽어옴. 파라미터 변수 name에는 DB 테이블의 컬럼이름을 적습니다.
2. getInt(String name):int - 지정한 컬럼 값을 int 타입으로 읽어옴.
3. getDouble(String name):double - 지정한 컬럼 값을 double 타입으로 읽어옴

* Statement 객체를 대신하는 PreparedStatement 객체

- Statement 객체와 PreparedStatement객체는 쿼리문을 실행하는 동일한 기능을 제공합니다.
- 그런데 PreparedStatement 객체를 사용하는 이유는 이 객체가 값 변환을 자동으로 해주는 기능을 제공하고, 간결한 코드를 만들 수 있기 때문입니다.
- Statement객체는 지정할 값이 많아질 경우 따옴표가 복잡하게 얽히기 때문에 코드 작성에서 오류가 발생할 수도 있고, 코드 수정시에도 어려움이 발생합니다.
- 그러나 PreparedStatement 객체는 값을 지정할 때 값 부분을 물음표(?)로 처리하기 때문에 간단히 값을 지정할 수 있습니다. 이 때 첫번째 물음표의 인덱스는 1이며, 이후 물음표의 인덱스는 나오는 순서대로 인덱스 값이 1씩 증가합니다.