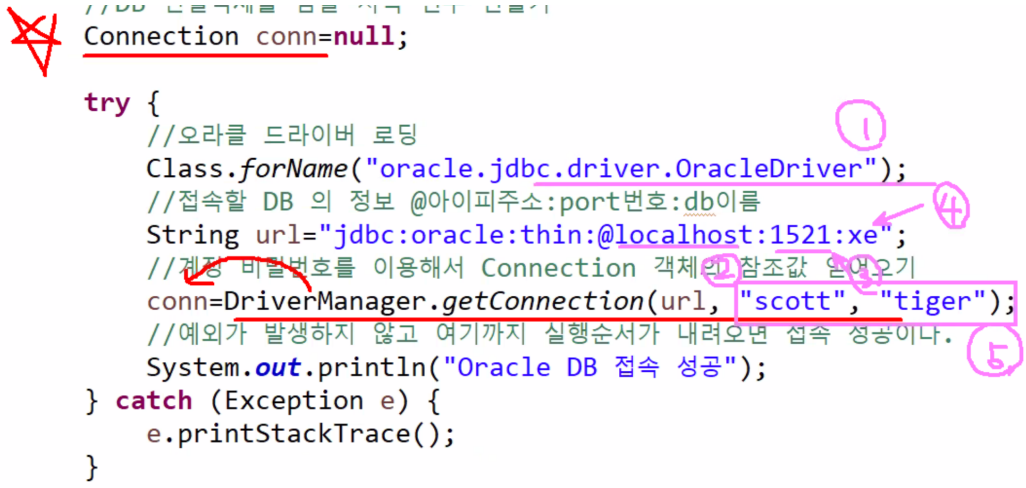
8_08_2022__java_JDBC

Java app 에서 Oracle app을 접속하기 위해서는 Driver Class가 필요하다 (ojdbc6.jar 사용함)

```
Connection conn = null;
```

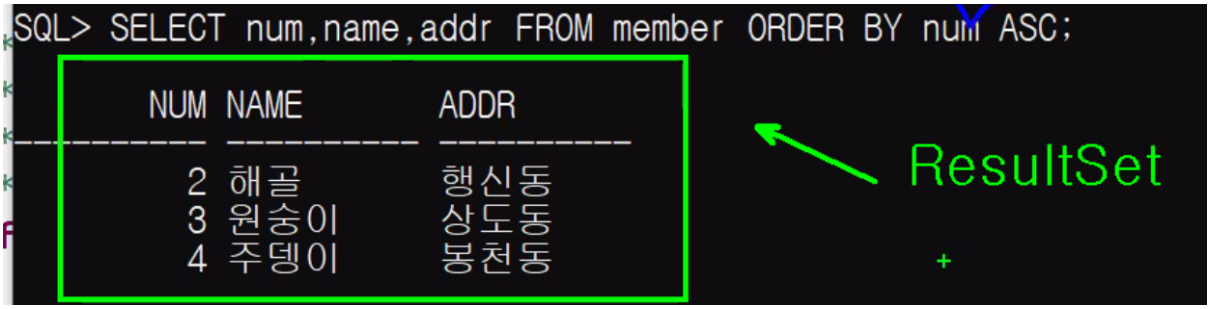
Connection 객체가 잘 접속되어야 한다.



Connection 만 받아주는 class를 만들어서 사용해도 된다.

```
conn = new DBConnect().getConn();
```

select 된 결과 값은 ResultSet에 담겨있다.



rs.next

row 맨 위에 cursor가 존자한다

이 cursor를 한칸씩 내리면서 data를 추출해오는 방식

더 이상 data가 없으면 빠져나온다.

primary key로 select를 하게 되면?

최대 하나밖에 없다 cursor를 여러번 내릴 필요가 없음

이럴 때는 반복문을 사용할 필요가 없음

insert();

동일 클래스 안이라 클래스 명은 입력하지 않아도 된다.

스태틱 메소드라 바로 불러올 수도 있음

▼ 회원 한명의 정보를 삭제하는 메소드

▼ 분석

기본 코드 작성

```
package test.main;

public class MainClass12 {
  public static void main(String[] args) {
    delete(2);

  }

  //회원 한명의 정보를 삭제하는 메소드
  public static void delete(int num) {

  }
}
```

여기서 확실히 삭제가 되었는지 확인하고 싶다면 boolean 타입으로 return 받으면 된다.

```
package test.main;

public class MainClass12 {
  public static void main(String[] args) {
    boolean isSuccess = delete(2);

    if(isSuccess) {

    }else {

    }
  }
  //회원 한명의 정보를 삭제하는 메소드
  public static boolean delete(int num) {
    return false;
  }
}
```

지역변수 만들고 담기

```
package test.main;

import java.sql.Connection;
import java.sql.PreparedStatement;

import test.util.DBConnect;

public class MainClass12 {
  public static void main(String[] args) {
    boolean isSuccess = delete(2);

    if (isSuccess) {

    } else {

    }
  }

  // 회원 한명의 정보를 삭제하는 메소드
  public static boolean delete(int num) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    //변화된(추가, 수정, 삭제) 행의 갯수를 담을 지역변수를 미리 만들어 둔다.
    int updatedRowCount = 0;
    try {
      // DBConnect 객체를 이용해서 Connection 객체의 참조값을 얻어온다.
      conn = new DBConnect().getConn();
      // 실행할 sql 문
      String sql = "DELETE FROM member" + " WHERE num=?";
      // PreparedStatement 객체 얻어내기
      pstmt = conn.prepareStatement(sql);
      // ? 바인딩 할게 있으면 바인딩 한다.
      pstmt.setInt(1, num);
      // 실행 후 메소드가 리턴해주는 변화된 행의 갯수를 지역변수에 담는다.
      updatedRowCount = pstmt.executeUpdate();
      System.out.println("회원정보를 삭제했습니다.");
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
      } catch (Exception e) {
      }
    }
  }
}
```

▼ 코드

```
package test.main;

import java.sql.Connection;
import java.sql.PreparedStatement;

import test.util.DBConnect;

public class MainClass12 {
  public static void main(String[] args) {
    //삭제할 회원의 번호라고 가정
    int num =2;
    //회원의 정보를 삭제하고 성공여부를 리턴 받는다.
```

```
    boolean isSuccess = delete(2);
    //성공이냐 실패냐에 따라 선택적인 작업을 할 수 있따.
    if (isSuccess) {
      System.out.println(num+" 번 회원을 삭제했습니다.");
    } else {
      System.out.println(num+" 번 회원 삭제 실패");
    }
  }

  // 회원 한명의 정보를 삭제하는 메소드
  public static boolean delete(int num) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    // 변화된(추가, 수정, 삭제) 행의 갯수를 담을 지역변수를 미리 만들어 둔다.
    int updatedRowCount = 0;
    try {
      // DBConnect 객체를 이용해서 Connection 객체의 참조값을 얻어온다.
      conn = new DBConnect().getConn();
      // 실행할 sql 문
      String sql = "DELETE FROM member" + " WHERE num=?";
      // PreparedStatement 객체 얻어내기
      pstmt = conn.prepareStatement(sql);
      // ? 바인딩 할게 있으면 바인딩 한다.
      pstmt.setInt(1, num);
      // 실행 후 메소드가 리턴해주는 변화된 행의 갯수를 지역변수에 담는다.
      updatedRowCount = pstmt.executeUpdate();
      System.out.println("회원정보를 삭제합니다.");
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (pstmt != null)pstmt.close();
        if (conn != null)conn.close();
      } catch (Exception e) {
      }
      //만일 변화된 행의 갯수가 0보다 크면
      if(updatedRowCount > 0) {
        //작업 성공의 의미이기 때문에 true를 리턴하고
        return true;
      }else {
        return false;
      }
    }
  }
}
```

DAO(data acess object_

어떤 서비스를 만드는데

- 회원 한명의 정보를 불러오는 작업
- 회원 한명의 정보를 삭제하는 작업
- 회원 한명의 정보를 수정하는 등의 작업을

여러번 사용해야 하면 객체를 만들어 놓고 불러오면 되게 만들 수 있다.

▼ MemberDao 코드

```
package test.dao;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import test.dto.MemberDto;
import test.util.DBConnect;

public class MemberDao {
  // 회원 한명의 정보를 추가하고 성공 여부를 리턴하는 메소드
  public boolean insert(MemberDto dto) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    // 변화된(추가, 수정, 삭제) 행의 갯수를 담을 지역변수를 미리 만들어 둔다.
    int updatedRowCount = 0;
    try {
      // DBConnect 객체를 이용해서 Connection 객체의 참조값을 얻어온다.
      conn = new DBConnect().getConn();
      // 실행할 sql 문
      String sql = "insert into member" + " (num, name, addr)" + " values(member_seq.nextval, ?, ?)";
      // PreparedStatement 객체 얻어내기
      pstmt = conn.prepareStatement(sql);
      // ? 바인딩 할게 있으면 바인딩 한다.
      pstmt.setString(1, dto.getName());
      pstmt.setString(2, dto.getAddr());
      // 실행후 메소드가 리턴해주는 변화된 행의 갯수를 지역변수에 담는다.
      updatedRowCount = pstmt.executeUpdate();
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (pstmt != null)
          pstmt.close();
        if (conn != null)
          conn.close();
      } catch (Exception e) {
      }
    }
    // 만일 변화된 행의 갯수가 0보다 크면
    if (updatedRowCount > 0) {
      // 작업 성공의 의미이기 때문에 true 를 리턴하고
      return true;
    } else {
      // 작업이 실패면 false 를 리턴한다.
      return false;
    }
  }

  // 회원 한명의 정보를 수정하고 성공여부를 리턴하는 메소드
  public boolean update(MemberDto dto) {
    // MemberDto 객체에 담긴 회원정보를 DB에 저장하는 작업을 해보세요(시퀀스 사용하기)
    Connection conn = null;
    PreparedStatement pstmt = null;
    int updatedRowCount = 0;
    try {
      // 오라클 드라이버 로딩
      Class.forName("oracle.jdbc.driver.OracleDriver");
      // 접속할 DB 의 정보 @아이피주소:port번호:db이름
      String url = "jdbc:oracle:thin:@localhost:1521:xe";
      // 계정 비밀번호를 이용해서 Connection 객체의 참조값 얻어오기
      conn = DriverManager.getConnection(url, "scott", "tiger");
      // 예외가 발생하지 않고 여기까지 실행순서가 내려오면 접속 성공이다.
      System.out.println("Oracle DB 접속 성공");
    } catch (Exception e) {
      e.printStackTrace();
    }
    conn = new DBConnect().getConn();
    try {
      String sql = "UPDATE member" + " Set name=?" + " , addr=?" + " Where num=?";
```

```
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, dto.getName());
      pstmt.setString(2, dto.getAddr());
      pstmt.setInt(3, dto.getNum());

      pstmt.executeUpdate();
    } catch (Exception e) {
    }
    if (updatedRowCount > 0) {
      // 작업 성공의 의미이기 때문에 true를 리턴하고
      return true;
    } else {
      return false;
    }

  }

  // 회원 한명의 ㅈ어보를 삭제하고 성공여부를 리턴하는 메소드
  public boolean delete(int num) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    int updatedRowCount = 0;
    // 변화된(추가, 수정, 삭제) 행의 갯수를 담을 지역변수를 미리 만들어 둔다.
    try {
      // DBConnect 객체를 이용해서 Connection 객체의 참조값을 얻어온다.
      conn = new DBConnect().getConn();
      // 실행할 sql 문
      String sql = "DELETE FROM member" + " WHERE num=?";
      // PreparedStatement 객체 얻어내기
      pstmt = conn.prepareStatement(sql);
      // ? 바인딩 할게 있으면 바인딩 한다.
      pstmt.setInt(1, num);
      // 실행 후 메소드가 리턴해주는 변화된 행의 갯수를 지역변수에 담는다.
      updatedRowCount = pstmt.executeUpdate();
      System.out.println("회원정보를 삭제합니다.");
    } catch (Exception e) {
      e.printStackTrace();
    }
    if (updatedRowCount > 0) {
      // 작업 성공의 의미이기 때문에 true를 리턴하고
      return true;
    } else {
      return false;
    }
  }
}
```

▼ 사용예제

```
package test.main;

import test.dao.MemberDao;
import test.dto.MemberDto;

public class MainClass13 {
  public static void main(String[] args) {
    String name="Jajangmyeon";
    String addr="IsacBurger";


    /*
     * 위의 회원 정보를 MemberDao 객체를 이용해서 DB에 저장하고
     * 성공이면 "회원 정보를 추가했습니다."
     * 실패면 "추가 실패"
     * 를 콘솔창에 출력하는 code를 작성
     */
```

```
    //추가할 회원의 정보를 MemberDto 객체를 생성해서 담는다.
    MemberDto dto = new MemberDto();
    dto.setName(name);
    dto.setAddr(addr);
    //MemberDao 객체를 생성해서
    MemberDao dao = new MemberDao();
    //insert()메소드를 이용, 회원의 정볼르 추가하고 성공 여부를 리턴 받는다.
    boolean isSuccess = dao.insert(dto);

    if (isSuccess) {
      System.out.println("회원 정보를 추가했습니다.");
    } else {
      System.out.println("추가 실패");
    }
  }
}
```

```
public boolean insert(MemberDto dto) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    // 변화된(추가, 수정, 삭제) 행의 갯수를 담을 지역변수를 미리 만들어 둔다.
    int updatedRowCount = 0;
    try {
      // DBConnect 객체를 이용해서 Connection 객체의 참조값을 얻어온다.
      conn = new DBConnect().getConn();
      // 실행할 sql 문
      String sql = "insert into member" + " (num, name, addr)" + " values(member_seq.nextval, ?, ?)";
      // PreparedStatement 객체 얻어내기
      pstmt = conn.prepareStatement(sql);
      // ? 바인딩 할게 있으면 바인딩 한다.
      pstmt.setString(1, dto.getName());
      pstmt.setString(2, dto.getAddr());
      // 실행후 메소드가 리턴해주는 변화된 행의 갯수를 지역변수에 담는다.
      updatedRowCount = pstmt.executeUpdate();
    } catch (Exception e) {
      e.printStackTrace();
    }
    // 만일 변화된 행의 갯수가 0보다 크면
    if (updatedRowCount > 0) {
      // 작업 성공의 의미이기 때문에 true 를 리턴하고
      return true;
    } else {
      // 작업이 실패면 false 를 리턴한다.
      return false;
    }
  }
```

위처럼 긴 메소드를 다른 클래스에 먼저 만들어 놓고

```
MemberDao dao = new MemberDao();
```

위처럼 메소드를 new 해서 짧게 사용한다.

▼ 입력

```
package test.main;
```

```
import test.dao.MemberDao;
import test.dto.MemberDto;

public class MainClass13 {
  public static void main(String[] args) {
    String name="Jajangmyeon";
    String addr="IsacBurger";


    /*
     * 위의 회원 정보를 MemberDao 객체를 이용해서 DB에 저장하고
     * 성공이면 "회원 정보를 추가했습니다."
     * 실패면 "추가 실패"
     * 를 콘솔창에 출력하는 code를 작성
     */

    //추가할 회원의 정보를 MemberDto 객체를 생성해서 담는다.
    MemberDto dto = new MemberDto();
    dto.setName(name);
    dto.setAddr(addr);
    //MemberDao 객체를 생성해서
    MemberDao dao = new MemberDao();
    //insert()메소드를 이용, 회원의 정볼르 추가하고 성공 여부를 리턴 받는다.
    boolean isSuccess = dao.insert(dto);

    if (isSuccess) {
      System.out.println("회원 정보를 추가했습니다.");
    } else {
      System.out.println("추가 실패");
    }
  }
}
```

▼ 수정

```
package test.main;

import test.dao.MemberDao;
import test.dto.MemberDto;

public class MainClass14 {
  public static void main(String[] args) {
    int num=3;
    String name="ramyeon";
    String addr="jip";

    /*
     * 위의 정보는 수정할 회원의 정보이다.
     *
     * MemberDao 객체를 이용해서 회운의 정보를 수정해 보세요
     */

    //수정할 회원의 정보를 MemberDto 객체에 담고
    MemberDto dto = new MemberDto(num, name, addr);
    //MemberDao 객체를 이용해서 수정한다.
    MemberDao dao = new MemberDao();
    boolean isSuccess = dao.update(dto);

    if(isSuccess) {
      System.out.println("수정 성공");
    }else {
      System.out.println("수정 실패");
    }
  }
}
```

▼ 삭제

```
package test.main;

import test.dao.MemberDao;

public class MainClass15 {
  public static void main(String[] args) {
    int num = 3;

    /*
     *  MemberDao 객체를 이용해서 위의 번호에 해당하는 회원정보를 삭제해 보세요
     */
    boolean isSuccess = new MemberDao().delete(num);
    if(isSuccess) {
      System.out.println("삭제 성공");
    }else {
      System.out.println("삭제 실패");
    }
  }
}
```