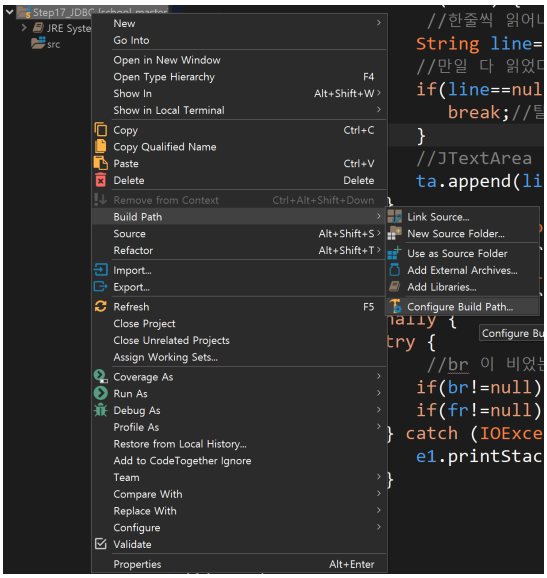
8_05_2022__java_JDBC

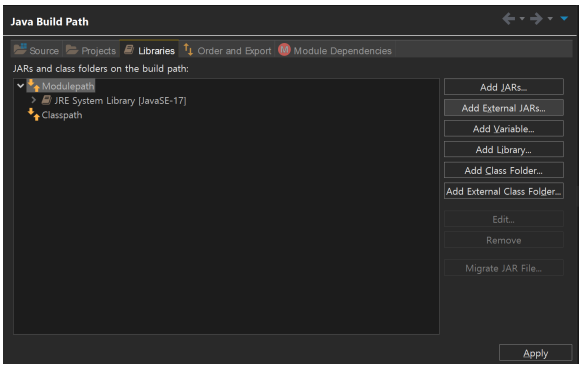
Step17_JDBC

DB연동

먼저 기본적으로 JAVA가 제공하는 기본 Library만 쓸 수 있지만 더 많은 기능을 추가해려 함



configure Build Path



add External jars

▼ 콘솔창에 sql정보 출력하는 방법

▼ 분석

- db 연결을 위해서는 Connection 이 필요
- 오라클 드라이버 로딩

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

- 접속할 db의 정보 확인@아이피주소:port번호:db이름

```
String url="jdbc:oracle:thin:@localhost:1521:xe";
```

- 비밀번호를 이용해 객체의 참조값 불러오기

```
conn=DriverManager.getConnection(url, "scott", "tiger");
```

- 예외가 발생하지 않으면 성공!
- preparedStatement

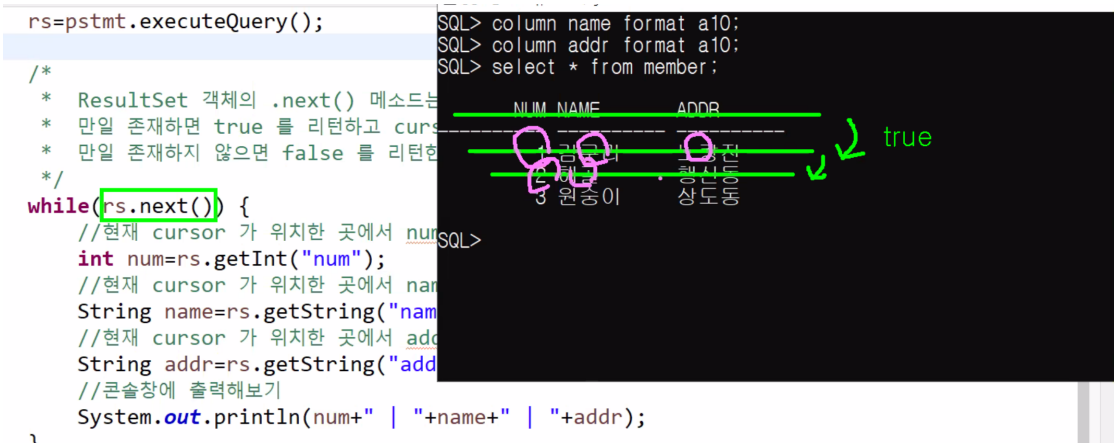
  ```
  pstmt=conn.prepareStatement(sql); 을 사용하기 위한 문
  ```

  - sql문을 대신 실행해 주는 역할
    - insert, update, delete 작업을 대신 해줌
  - select 문을 실행할때는 excuteQuery()를 실행하라
    - wxcuteQuery() 는 resultSet(결과셋)을 리턴한다.
  - ResultSet 객체의 .next()메소드는 cursor 밑에 row가 존재하는지 확인해서
    - 만일 존재하면 true를 리턴하고 cursor 가 한칸 밑으로 이동한다.
    - 만일 존재하지 않으면 false를 리턴한다.
  - PreparedStatement 객체를 이용해서 query문 수행하고 결과를 ResultSet 객체로 받아오기

  ```
  rs=pstmt.executeQuery();
  ```

- 커서기준으로 더이상 정보가 없을때 까지 받아오기

```
while(rs.next()) {
  //현재 cursor 가 위치한 곳에서 num 이라는 칼럼의 정수 얻어내기
  int num=rs.getInt("num");
  //현재 cursor 가 위치한 곳에서 name 이라는 칼럼의 문자열 얻어내기
  String name=rs.getString("name");
  //현재 cursor 가 위치한 곳에서 addr 이라는 칼럼의 문자열 얻어내기
  String addr=rs.getString("addr");
  }
```



- 콘솔창에 출력하기

```
System.out.println(num+" | "+name+" | "+addr);
```

```
package test.main;

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

public class MainClass {
  public static void main(String[] args) {
        //DB 연결객체를 담을 지역 변수 만들기
        Connection conn=null;

        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();
        }

        //select 작업을 위해서 필요한 객체의 참조값을 담을 지역변수 미리 만들기
        PreparedStatement pstmt=null;
        ResultSet rs = null;
        try {
      //실행할 sql문
          String sql ="select num,name,addr from member"
              + " order by num asc";
          //PreparedStatement 객체의 참조값 얻어오기
          pstmt=conn.prepareStatement(sql);
          //PreparedStatement 객체를 이용해서 query문 수행하고 결과를
          //ResultSet 객체로 받아오기
          rs=pstmt.executeQuery();
          /*
           * ResultSet 객체의 .next()메소드는 cursor 밑에 row가 존재하는지 확인해서
           * 만일 존재하면 true를 리턴하고 cursor 가 한칸 밑으로 이동한다.
           * 만일 존재하지 않으면 false를 리턴한다.
           */
```

```
          while(rs.next()) {
            //현재 cursor 가 위치한 곳에서 num 이라는 칼럼의 정수 얻어내기
            int num=rs.getInt("num");
            //현재 cursor 가 위치한 곳에서 name 이라는 칼럼의 문자열 얻어내기
            String name=rs.getString("name");
            //현재 cursor 가 위치한 곳에서 addr 이라는 칼럼의 문자열 얻어내기
            String addr=rs.getString("addr");
            //콘솔창에 출력해보기
            System.out.println(num+" | "+name+" | "+addr);
          }
    } catch (Exception e) {
    }
        System.out.println("maim 메소드가 종료됩니다.");
  }
}
```

▼ 정보 추가하는 방법

▼ 분석

- 미완성의 sql 문

```
String sql="INSERT INTO member"
         + " (num, name, addr)"
         + " VALUES(?, ?, ?)";
```

- ?에 원하는 값을 바인딩 하는 방법

```
pstmt.setInt(1, num);
pstmt.setString(2, name);
pstmt.setString(3, addr);
```

- executeUpdate()

```
pstmt.executeUpdate();
```

```
package test.main;

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

public class MainClass02 {
   public static void main(String[] args) {
      //DB 연결객체를 담을 지역 변수 만들기
      Connection conn=null;

      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();
      }

      //member 테이블에 추가할 회원의 정보라고 가정
      int num=4;
      String name="Jazz";
      String addr="Shinrim";

      //SELECT 작업을 위해서 필요한 객체의 참조값을 담을 지역변수 미리 만들기
      PreparedStatement pstmt=null;
      try {
         //실행할 미완성의 sql 문
         String sql="INSERT INTO member"
               + " (num, name, addr)"
               + " VALUES(?, ?, ?)";
         //PreparedStatement 객체의 참조값 얻어오기
         pstmt=conn.prepareStatement(sql);
         // ? 에 값을 바인딩해서 미완성의 sql 문을 완성 시킨다.
         pstmt.setInt(1, num);
         pstmt.setString(2, name);
         pstmt.setString(3, addr);
           //sql 문 실행하기
           pstmt.executeUpdate();
           System.out.println("회원 정보를 저장했습니다.");
      }catch(Exception e) {
         e.printStackTrace();
      }

   }
}
```

▼ 정보를 수정 하는 법

▼ 분석

- 모양이 조금 다르다.

```
String sql="UPDATE member"
     + " SET addr=?"
     + " WHERE num=?";
```

- 참조값을 불러와서 적용시켜준다

```
 //PreparedStatement 객체의 참조값 얻어오기
 pstmt=conn.prepareStatement(sql);
 //sql 문 완성하기
 pstmt.setString(1, addr);
 pstmt.setInt(2, num);
```

```
package test.main;

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

/*
 *  JDBC ( Java DataBase Connectivity )
 *
 *  DataBase 에 연결해서 SELECT, INSERT, UPDATE, DELETE 작업하기
```

```
 *
 *  Oracle 에 연결하기 위해서는 드라이버 클래스가 들어있는 ojdbc6.jar 파일을
 *  사용할수 있도록 설정해야 한다.
 */
public class MainClass03 {
   public static void main(String[] args) {
      // 3 번 회원의 주소를 동물원으로 수정하고자 한다.
      int num=3;
      String addr="동물원";

      //DB 연결객체를 담을 지역 변수 만들기
      Connection conn=null;

      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();
      }

      //SELECT 작업을 위해서 필요한 객체의 참조값을 담을 지역변수 미리 만들기
      PreparedStatement pstmt=null;
      try {
         //실행할 미완성의 sql 문
         String sql="UPDATE member"
               + " SET addr=?"
               + " WHERE num=?";
         //PreparedStatement 객체의 참조값 얻어오기
         pstmt=conn.prepareStatement(sql);
         //sql 문 완성하기
         pstmt.setString(1, addr);
         pstmt.setInt(2, num);

         //sql 문 실행하기
         pstmt.executeUpdate();
         System.out.println("회원 정보를 수정 했습니다.");
      }catch(Exception e) {
         e.printStackTrace();
      }

      System.out.println("main 메소드가 종료 됩니다.");

   }
}


insert, update, delete 모두 excuteUpdate()를 사용한다.
```

▼ 정보 삭제 하는 방법

▼ 분석

- 삭제도 마찬가지로 미완성을 만들고 정보를 불러온다.

```
String sql="DELETE FROM member"
       + " WHERE num=?";
 //PreparedStatement 객체의 참조값 얻어오기
 pstmt=conn.prepareStatement(sql);
 //? 에 바인딩(연결)할 내용이 있으면 연결하고
 pstmt.setInt(1, num);
```

```
package test.main;

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

public class MainClass05 {
   public static void main(String[] args) {
      /*
       *  run 했을때 member 테이블에서 1번 회원의 정보를 삭제 해 보세요.
       */
      int num=1;
      //DB 연결객체를 담을 지역 변수 만들기
      Connection conn=null;

      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();
      }

      //DELETE 작업을 위해서 필요한 객체의 참조값을 담을 지역변수 미리 만들기
      PreparedStatement pstmt=null;
      try {
         //실행할 sql 문
         String sql="DELETE FROM member"
               + " WHERE num=?";
         //PreparedStatement 객체의 참조값 얻어오기
         pstmt=conn.prepareStatement(sql);
         //? 에 바인딩(연결)할 내용이 있으면 연결하고
         pstmt.setInt(1, num);
         //sql 문 실행하기
         pstmt.executeUpdate();

      }catch(Exception e) {
         e.printStackTrace();
      }

      System.out.println("main 메소드가 종료 됩니다.2");
   }
}
```

▼ 자동commit

- JAVA에서 executeUpdate 를 하게 되면 자동으로 commit이 된다.
- setAutoCommit()을 false로 해 놓으면 수동으로 commit을 할 수 있다.
  - 작업을 마치고 XXX.commit(); 해주면 됨
  - 여러 작업을 동시에 할때 사용하면 좋다.
  - exception이 발생할 때 rollback();을 하면 관리하기 쉽다.

▼ Connection을 잡아주는 클래스 만들기

- Connection conn = new Connection(); 해 쉽게 사용하기 위함
  - 필드지정

```
package test.util;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnect{
  // 필드
  Connection conn;
  // 생성자
  public DBConnect() {
    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();
    }
  }
  // Connection 객체를 리턴해주는 메소드
  public Connection getConn() {
    return conn;
  }
}
```

▼ 정보를 추가하는 메소드를 만들기

▼ 분석

- 추가할 회원의 정보를 입력하고

```
String name="Jazz";
String addr="Shinrim";
```

- 회원 한명의 정보를 한번에 추가하는 메소드
  - Dto를 만든다.(MemberDto)
  - 회원 정보를 MemberDto 객체에 담고 호출

```
// 회원 한명의 정보를 MemberDto 객체에 담고
MemberDto dto = new MemberDto();
```

```
dto.setName(name);
dto.setAddr(addr);

// 메소드를 호출하면서 MemberDto 객체를 전달한다.
insert(dto);
```

- dto 참조값을 가져온다.

```
pstmt.setString(1, dto.getName());
pstmt.setString(2, dto.getAddr());
```

```
package test.main;

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

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

public class MainClass07 {
  public static void main(String[] args) {
    // member 테이블에 추가할 회원의 정보
    String name = "Jazz";
    String addr = "Shinrim";

    // 회원 한명의 정보를 MemberDto 객체에 담고
    MemberDto dto = new MemberDto();
    dto.setName(name);
    dto.setAddr(addr);

    // 메소드를 호출하면서 MemberDto 객체를 전달한다.
    insert(dto);

  }

  // 회원 한명의 정보를 추가하는 메소드
  public static void insert(MemberDto dto) {
    // MemberDto 객체에 담긴 회원정보를 DB에 저장하는 작업을 해보세요(시퀀스 사용하기)
    Connection conn = null;
    PreparedStatement pstmt = null;
    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());
      // 실행
      pstmt.executeUpdate();
      System.out.println("회원정보를 저장했습니다.");
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (pstmt != null)
          pstmt.close();
        if (conn != null)
          conn.close();
      } catch (Exception e) {
      }
    }
```

▼ Map을 사용해 추가

```
package test.main;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.HashMap;
import java.util.Map;

import test.util.DBConnect;

public class MainClass08 {
  public static void main(String[] args) {
    String name = "Timon";
    String addr = "High Desert";
    // 추가 할 회원 정보를 Map객체에 담고
    Map<String, String> map = new HashMap<>();
    map.put("name", name);
    map.put("addr", addr);
    // insert 메소드에 전달해서 DB에 저장되는 기능을 완성해 보세요
    insert(map);
  }

  // 인자로 전달 받은 Map 객체에 있는 회원 정보를 DB에 저장하는 메소드
  public static void insert(Map<String, String> map) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    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, map.get("name"));
      pstmt.setString(2, map.get("addr"));
      // 실행
      pstmt.executeUpdate();
      System.out.println("회원정보를 저장했습니다.");
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (pstmt != null)
          pstmt.close();
        if (conn != null)
          conn.close();
      } catch (Exception e) {
      }
    }

  }
}
```

▼ 정보 찾아 전달받기

```
package test.main;

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

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

public class MainClass11 {
  public static void main(String[] args) {
    int num = 2;
    // 지역변수 num에 담긴 내용을 메소드의 인자로 전달해서 회원 한명의 ㅈ어보를 얻어낸다.
    MemberDto dto = getData(num);
    if (dto == null) {
      System.out.println(num + "번 회원은 존재하지 않습니다.");
    } else {

      System.out.println(num + "번 회원의 이름은" + dto.getName() + " 이고 주소는 " + dto.getAddr() + " 입니다.");
    }
  }

  // 1하면 1정보 리턴 2하면 2정보 리턴 3 하면 3정보 리턴하려는 기능을 만드려 함
  // 인자로 전달되는 번호에 해당되는 회원 한명의 정보를 리턴하는 메소드
  public static MemberDto getData(int num) {
    // 필요한 객체를 담을 지역 변수를 미리 만든다
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    MemberDto dto = null;
    try {
      conn = new DBConnect().getConn();
      // 실행할 sql 문 구성하기
      String sql = "select name, addr" + " from member" + " where num=?";
      pstmt = conn.prepareStatement(sql);
      // ?에 바인딩 할 내용이 있으면 한다.
      pstmt.setInt(1, num);
      // select 문 수행하고 결과를 ResultSet으로 얻어내기
      rs = pstmt.executeQuery();
      /*
       * primary key 로 select 를 하게 되면 select 된 row는 최대 1개이므로 cursor를 반복문 사용 하면서 여러번 내릴
       * 필요가 없다. 즉 rs.next() 한번만 수행되면 된다.
       */

      if (rs.next()) {
        // 커서가 위치한 곳에 select 된 칼럼 정보를 얻어낸다.
        String name = rs.getString("name");
        String addr = rs.getString("addr");
        // select 된정보를 MemberDto 객체를 생성해서 담는다.
        dto = new MemberDto();
        dto.setNum(num);
        dto.setName(name);
        dto.setAddr(addr);
      }
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (rs != null)
          rs.close();
        if (pstmt != null)
          pstmt.close();
        if (conn != null)
          pstmt.close();
      } catch (Exception e) {
      }
    }
    // select 된 row 가 없다면 dto 는 null이다.
    return dto;
  }
}
```