

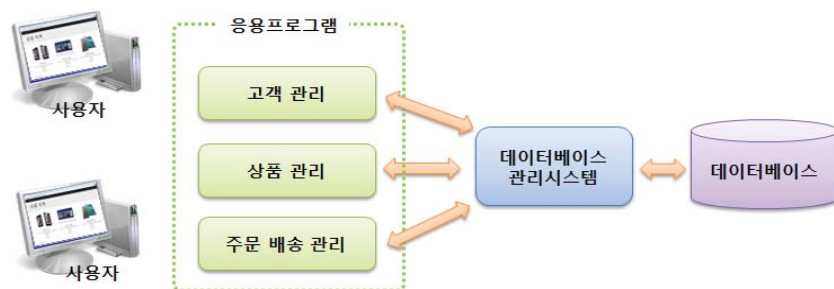
JDBC 프로그래밍

524730-1
2019년 봄학기
4/8/2019
박경신

데이터베이스의 개념

- 데이터베이스(Database)
 - 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 집합
 - 데이터의 저장, 검색, 갱신을 효율적으로 수행할 수 있도록 데이터를 고도로 조직화하여 저장
- DBMS
 - 데이터베이스 관리 시스템(DataBase Management System)
 - 오라클(Oracle), 마이크로소프트의 SQL Server, MySQL, IBM의 DB2 등
- 데이터베이스 종류
 - 관계형 데이터베이스
 - 키(key)와 값(value)들의 관계를 테이블로 표현한 데이터베이스 모델
 - 키는 테이블의 열(column)이 되며 테이블의 행(row)은 하나의 레코드(record)를 표현
 - 현재 사용되는 대부분의 데이터베이스는 관계형 데이터베이스
 - 객체 지향 데이터베이스
 - 객체 지향 프로그래밍에 쓰이는 것으로, 정보를 객체의 형태로 표현하는 데이터베이스 모델
 - 오브젝트 데이터베이스(object database)라고도 부름

데이터베이스 사례



관계형 데이터베이스 구조

- 관계형 데이터 베이스
 - 데이터들이 다수의 테이블로 구성
 - 각 행은 하나의 레코드
 - 각 테이블은 키(key)와 값(value)들의 관계로 표현
 - 여러 테이블 간에 공통된 이름의 열 포함
 - 이런 경우 서로 다른 테이블 간에 관계(relation)가 성립
 - 대부분의 데이터베이스는 관계형 데이터베이스
 - JDBC API

키 값

| employee_id | name |
|-------------|------|
| 00001 | 김철수 |
| 00002 | 최고봉 |
| 00003 | 이기자 |

테이블 A

두 테이블이 동일한 키를 가지고 있음 : 관계

| payroll_id | amount | employee_id |
|------------|---------|-------------|
| 00000001 | 1000000 | 00002 |
| 00000002 | 2000000 | 00010 |
| 00000003 | 3000000 | 00021 |

테이블 B

객체지향 데이터베이스

- 객체지향 데이터 베이스
 - 객체 지향 프로그래밍에 사용
 - 정보를 객체의 형태로 표현
 - 오브젝트 데이터베이스(object database)라고도 부름
 - 객체 모델이 그대로 데이터베이스에도 적용되므로 응용프로그램의 객체 모델과 데이터베이스의 모델이 부합됨

SQL과 JDBC

- SQL(Structured Query Language)
 - 관계형 데이터베이스 관리 시스템에서 사용
 - 데이터베이스 스키마 생성, 자료의 검색, 관리, 수정, 그리고 데이터베이스 객체 접근 관리를 위해 고안된 언어
 - 데이터베이스로부터 정보를 추출하거나 갱신하기 위한 표준 대화식 프로그래밍 언어
 - 다수의 데이터베이스 관련 프로그램들이 SQL을 표준으로 채택
- JDBC(Java DataBase Connectivity)
 - 관계형 데이터베이스에 저장된 데이터를 접근 및 조작할 수 있게 하는 API
 - 다양한 DBMS에 대해 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있게 함

JDBC 구조

- JDBC 드라이버 매니저
 - 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버
 - DBMS마다 고유한 JDBC 드라이버 제공, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS
 - 데이터베이스 관리 시스템, 데이터베이스 생성·삭제, 데이터 생성·검색·삭제 등 전담 소프트웨어 시스템

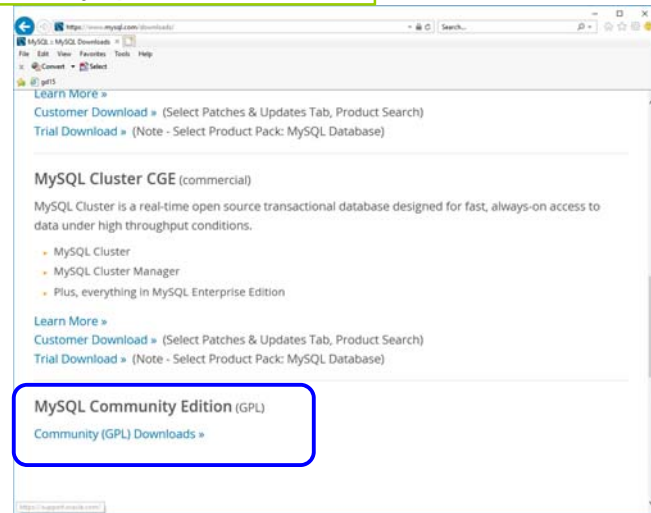


MySQL

- MySQL
 - 전 세계적으로 가장 널리 사용되고 있는 오픈 소스 관계형 데이터베이스 관리 시스템
 - 관계형 데이터베이스 관리 시스템의 표준화된 사용자 및 프로그래밍 인터페이스인 질의언어 SQL(Structured Query Language)을 사용
 - 매우 빠르고 유연하고 사용하기 쉽기 때문에 많은 기업에서 다양한 웹 기반 애플리케이션을 개발하는 데 사용

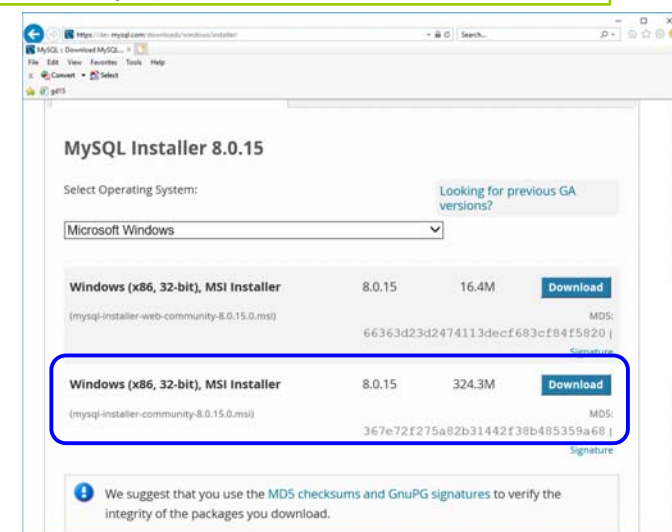
MySQL 서버 설치

<https://www.mysql.com/downloads/>

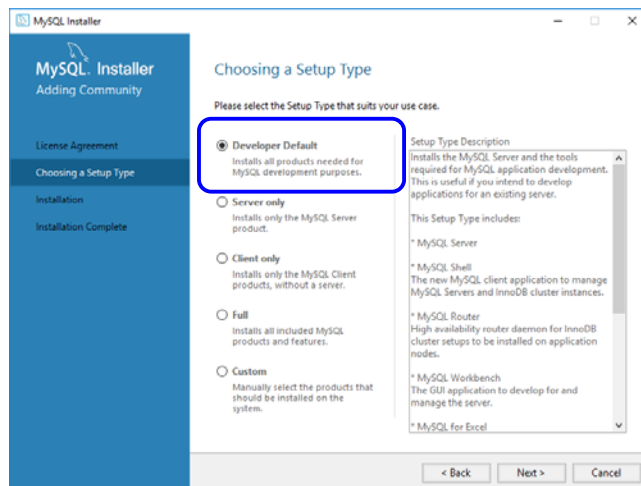


MySQL 서버 설치파일 다운로드

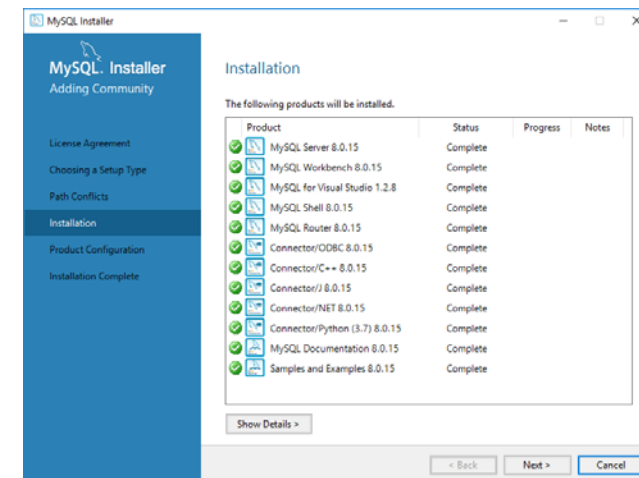
<https://dev.mysql.com/downloads/windows/installer>



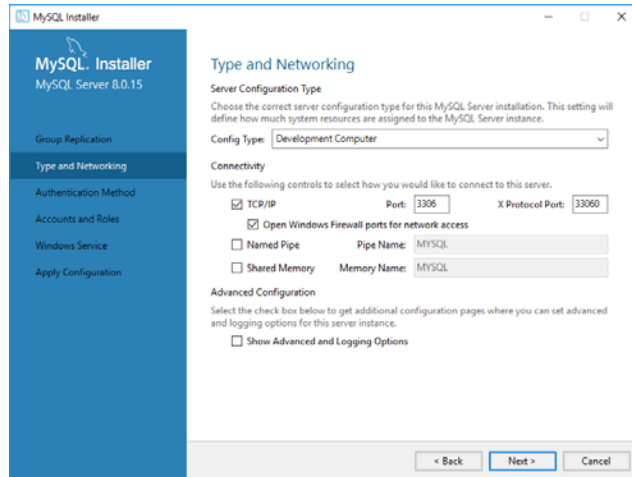
MySQL 서버 Default 타입으로 설치



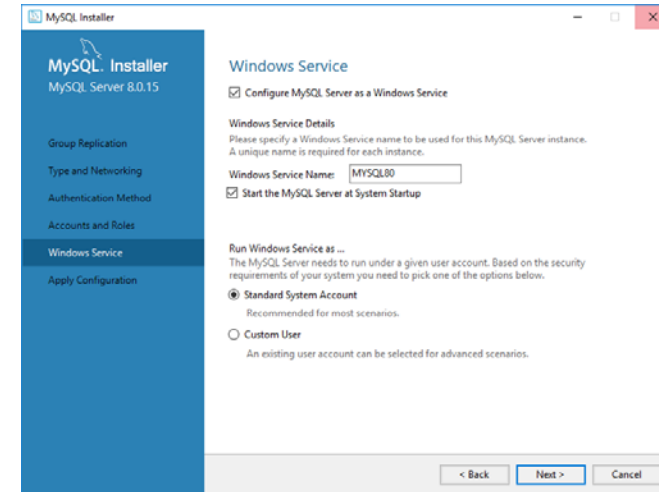
MySQL 서버 설치 및 설정



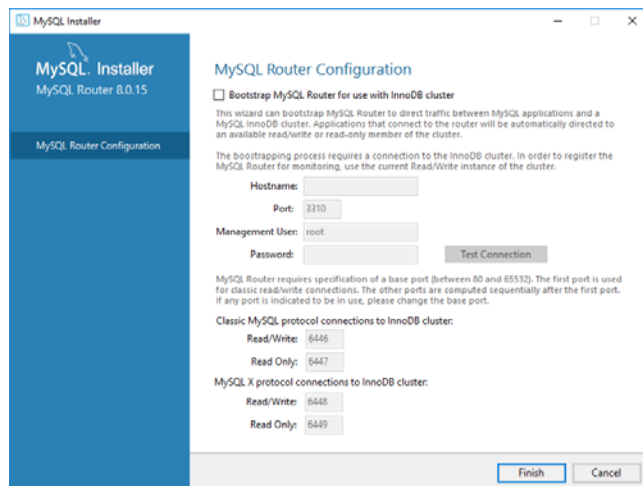
MySQL 서버 설치 및 설정



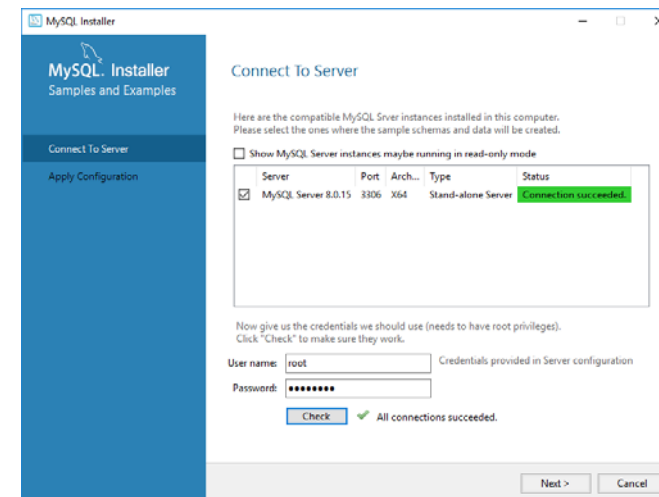
MySQL 서버 설치 및 설정



MySQL 서버 설치 및 설정

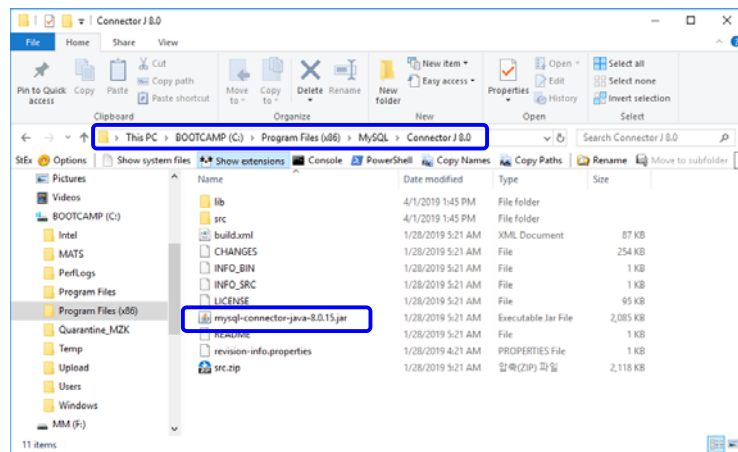


MySQL 설치 완료 및 서버 설정



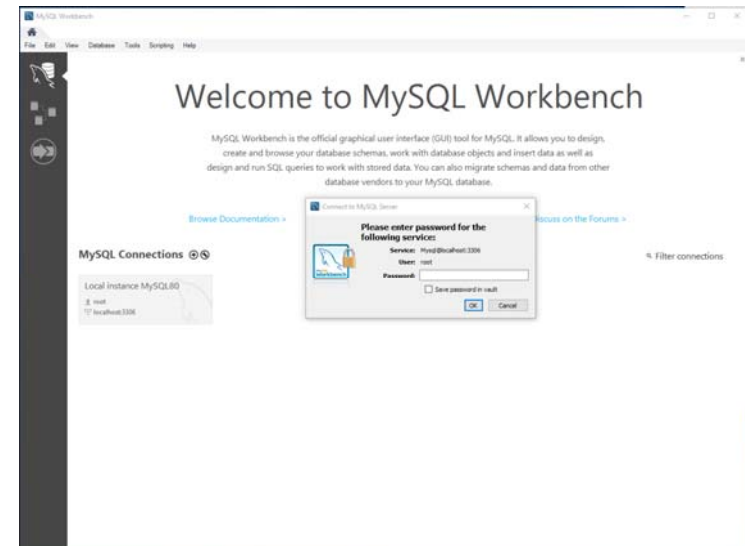
JDBC 드라이버 설치

□ JDBC 드라이버 확인



<https://dev.mysql.com/downloads/connector/j/>

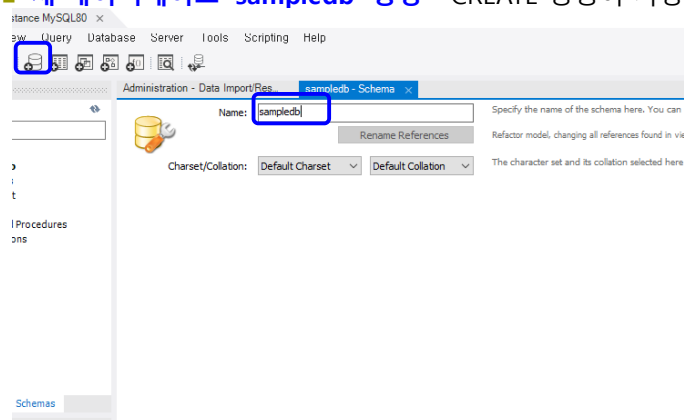
MySQL Workbench 실행



MySQL Workbench 실행

□ MySQL Workbench를 사용하여 새 DB 생성

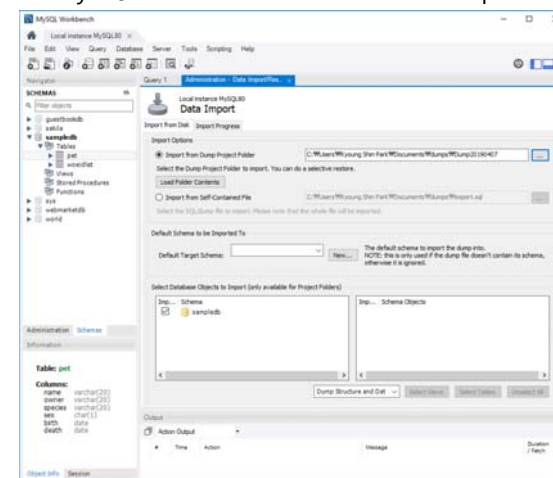
- MySQL Command Line Client 또는 MySQL Workbench를 실행한 후 root 계정의 비밀번호를 입력하여 MySQL에 접속
- 새 데이터베이스 'sampledb' 생성 - CREATE 명령어 사용



MySQL Workbench 실행

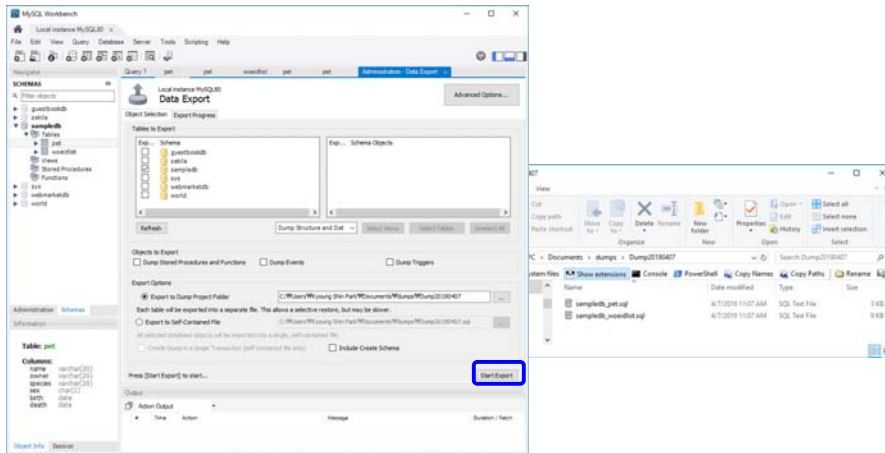
□ MySQL 데이터베이스 import

- MySQL Workbench Server->Data Import 'sampledb_pet.sql'



MySQL Workbench 실행

- MySQL 데이터베이스 export
 - MySQL Workbench Server->Data Export

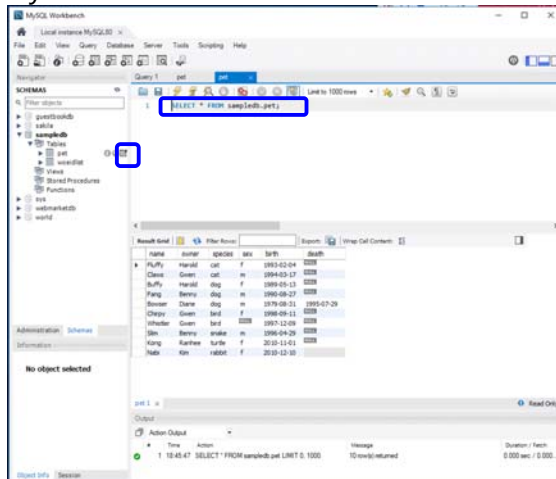


Eclipse와 DB 연동

- DB 연동
 - 이클립스에서 DB 를 연동하려면 DB Connection을 설정해야 함
 - DB Connection을 설정하기 전에 사용할 DB가 반드시 생성되어 있어야 함

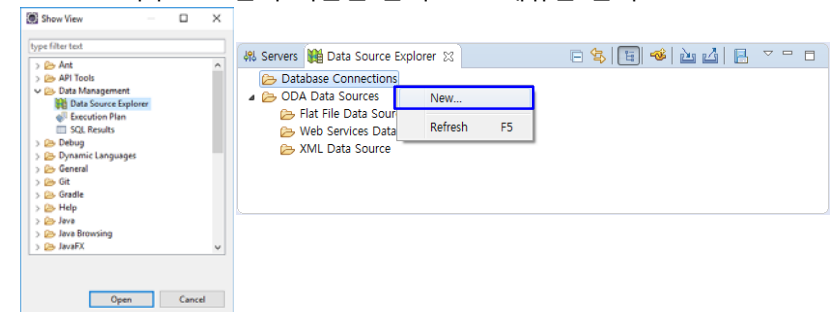
Eclipse와 DB 연동

- MySQL Workbench를 사용하여 DB 확인



Eclipse와 DB 연동

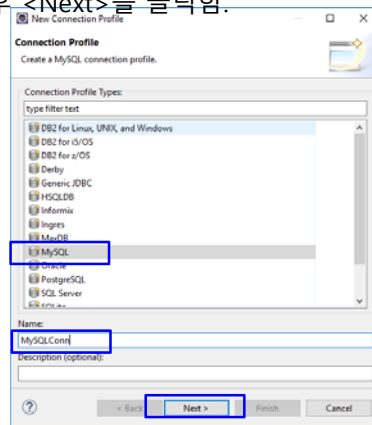
- 데이터베이스 커넥션 설정
 - Data Source Explorer 뷰를 이용하여 이클립스에 데이터베이스 커넥션을 설정
 - Data Source Explorer 뷰 열기: 이클립스 Window->Show View->Data Source Explorer를 선택하면 이클립스의 뷰에 나타남. Data Source Explorer 뷰에서 [Database Connections]를 선택한 후 마우스 오른쪽 버튼을 눌러 New 메뉴를 선택



Eclipse와 DB 연동

■ 커넥션 유형 설정하기:

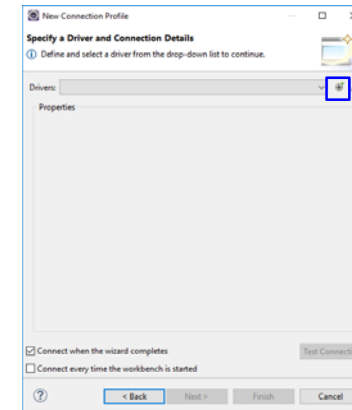
- [Connection Profile] 화면이 나타나면 Connection Profile Types에서 'MySQL'을 선택하고 Name에 'MySQLConn'을 입력한 후 <Next>를 클릭함.



Eclipse와 DB 연동

■ 드라이버와 커넥션 상세 설정하기:

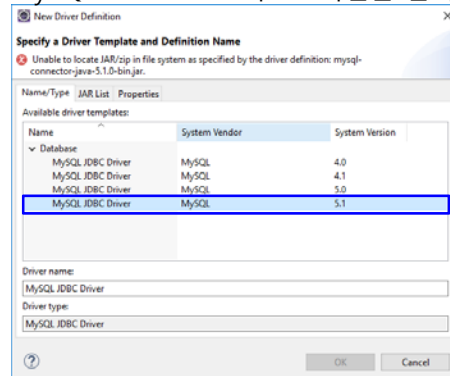
- [Specify a Driver and Connection Details] 화면이 나타나면 Drivers 항목의 (New Driver Definition) 버튼을 클릭함.



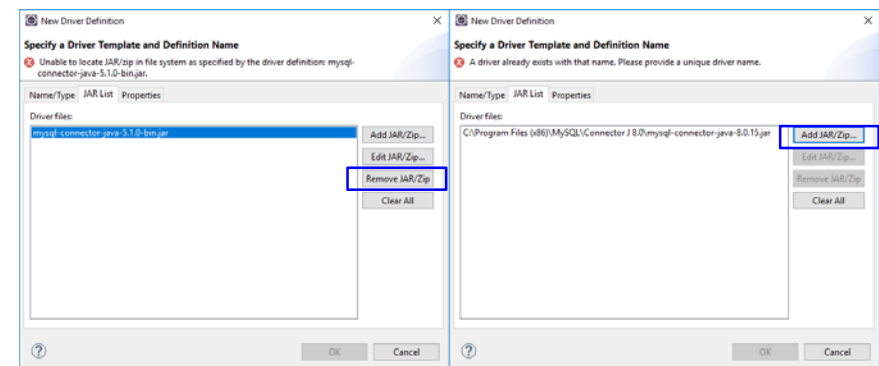
Eclipse와 DB 연동

■ 새로운 드라이버 설정하기:

- [New Driver Definition] 화면이 나타나면 Name/Type, JAR List, Properties 탭에 차례대로 필요한 설정을 진행.
- Name/Type 탭에서는 사용할 JDBC 드라이버와 버전을 선택함.
- Name/Type 탭에서 MySQL JDBC Driver의 5.1 버전을 선택함.



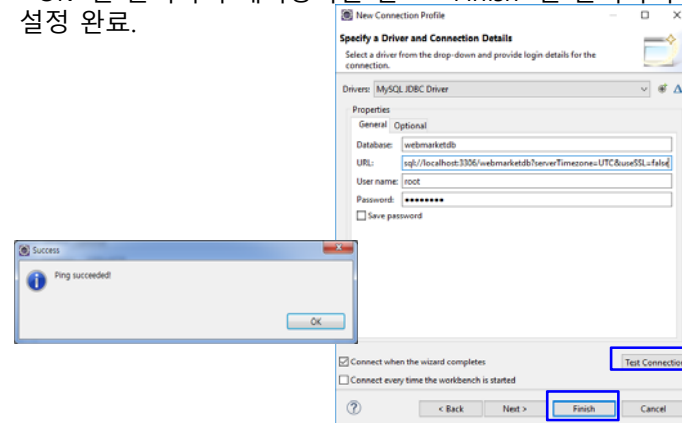
통합 개발 환경과 데이터베이스 연동



통합 개발 환경과 데이터베이스 연동

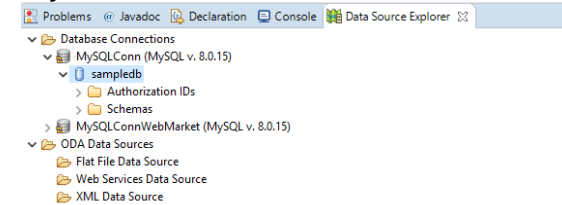
□ 커넥션 연결 설정 끝내기:

- 설정된 JDBC 드라이버와 데이터베이스의 이름이 표시되면 <Test Connection>을 클릭. [Success] 대화상자가 나타나면 <OK>를 클릭하여 대화상자를 닫고 <Finish>를 클릭하여 모든 설정 완료.

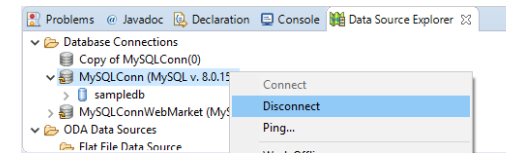


통합 개발 환경과 데이터베이스 연동

□ MySQL을 직접 제어하는 커넥션이 연결된 상태



□ 앞으로 커넥션 연결은 [Connect] 메뉴를, 커넥션 해제는 [Disconnect] 메뉴를 사용



데이터 베이스 접속 및 사용

□ DB 사용

- MySQL 8.0 command Line Client 실행하여, 생성된 DB에 접속



- -u 옵션은 root 계정으로 명령 수행을 의미
- use는 DB 사용 명령
- sampledb는 사용할 DB 이름

테이블 생성

- 데이터 저장을 위해 테이블을 생성
- 다음과 같은 구조의 pet 테이블 생성

| name | owner | species | sex | birth | death |
|-------------|-------------|-------------|---------|-------|-------|
| varchar(20) | varchar(20) | varchar(20) | char(1) | date | date |

- name, owner, species은 varchar 타입으로 20자
- sex는 char 타입으로 1자
- birth, death은 date 타입

테이블 생성

▣ 저장할 데이터

```
mysql> SELECT * from pet;
```

```
+-----+-----+-----+-----+-----+-----+
| name  | owner | species | sex | birth      | death      |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Harold | cat     | f   | 1993-02-04 | NULL       |
| Claws  | Gwen  | cat     | m   | 1994-03-17 | NULL       |
| Buffy  | Harold | dog     | f   | 1989-05-13 | NULL       |
| Fang   | Benny  | dog     | m   | 1990-08-27 | NULL       |
| Bowser  | Diane | dog     | m   | 1979-08-31 | 1995-07-29 |
| Chirpy  | Gwen  | bird    | f   | 1998-09-11 | NULL       |
| Whistler | Gwen | bird    | NULL | 1997-12-09 | NULL       |
| Slim   | Benny  | snake   | m   | 1996-04-29 | NULL       |
| Puffball | Diane | hamster | f   | 1999-03-30 | NULL       |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

테이블 생성

▣ create 문으로 테이블 생성

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
-> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
Query OK, 0 rows affected (0.15 sec)
```

- create table 다음에 테이블 이름 지정
- 열 이름 데이터 타입(데이터 크기)을 콤마로 분리하여 나열
- not null은 행의 데이터에 null값이 올 수 없음을 의미
- primary key는 키로 사용될 행 지정

▣ drop 명령은 기존의 테이블 삭제

```
mysql> DROP TABLE pet;
Query OK, 0 rows affected (0.15 sec)
```

테이블 생성

▣ desc 명령은 생성된 테이블의 구조 표시

```
mysql> DESCRIBE pet;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20) | YES  |     | NULL    |       |
| owner | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex   | char(1)    | YES  |     | NULL    |       |
| birth | date       | YES  |     | NULL    |       |
| death | date       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
Query OK, 0 rows affected (0.15 sec)
```

데이터 추가

▣ insert 문으로 테이블의 데이터 추가

```
mysql> INSERT INTO pet
-> VALUES ('Puffball', 'Diane', 'hamster', 'f', '1999-03-30', NULL);
Query OK, 1 row affected (0.10 sec)
mysql> INSERT INTO pet
-> VALUES ('Fluffy', 'Harold', 'cat', 'f', '1993-02-04', NULL);
Query OK, 1 row affected (0.10 sec)
mysql> INSERT INTO pet
-> VALUES ('Claws', 'Gwen', 'cat', 'm', '1994-03-17', NULL);
Query OK, 1 row affected (0.10 sec)
```

- insert into 다음에 테이블 이름 지정
- 테이블 이름 다음 괄호 안에 열 이름을 콤마로 구분하여 나열
- values 다음 괄호 안에 열의 값들을 콤마로 구분하여 나열
- 문자 타입의 데이터는 단일 인용 부호로 묶어서 표시함에 유의

데이터 추가

□ load문으로 파일로부터 테이블의 데이터 추가

```
mysql> LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/pet.csv' INTO TABLE pet FIELDS TERMINATED BY ',' LINES
TERMINATED BY '\r\n';
Query OK, 8 rows affected (0.10 sec)
mysql> SELECT * from pet;
+-----+-----+-----+-----+-----+-----+
| name   | owner | species | sex | birth      | death      |
+-----+-----+-----+-----+-----+-----+
| Fluffy  | Harold | cat      | f   | 1993-02-04 | NULL       |
| Claws   | Gwen  | cat      | m   | 1994-03-17 | NULL       |
| Buffy   | Harold | dog      | f   | 1989-05-13 | NULL       |
| Fang    | Benny  | dog      | m   | 1990-08-27 | NULL       |
| Bowser  | Diane  | dog      | m   | 1979-08-31 | 1995-07-29 |
| Chirpy  | Gwen  | bird     | f   | 1998-09-11 | NULL       |
| Whistler | Gwen  | bird     | NULL | 1997-12-09 | NULL       |
| Slim    | Benny  | snake    | m   | 1996-04-29 | NULL       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

데이터 검색

□ select문으로 테이블 내의 데이터 검색

```
mysql> SELECT * from pet where owner='Gwen';
+-----+-----+-----+-----+-----+-----+
| name   | owner | species | sex | birth      | death      |
+-----+-----+-----+-----+-----+-----+
| Claws   | Gwen  | cat      | m   | 1994-03-17 | NULL       |
| Chirpy   | Gwen  | bird     | f   | 1998-09-11 | NULL       |
| Whistler | Gwen  | bird     | NULL | 1997-12-09 | NULL       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- select 다음에는 데이터를 추출할 열 이름을 콤마로 분리하여 나열
- 모든 열에 대해 데이터를 추출할 때는 *를 열 이름 대신 사용
- from 다음에 테이블 이름을 지정
- where 다음에 검색 조건 지정. where는 생략 가능

데이터 검색

□ select문으로 테이블 내의 데이터 검색

```
mysql> SELECT owner, species, birth FROM pet WHERE name = 'Buffy';
+-----+-----+-----+
| owner | species | birth      |
+-----+-----+-----+
| Harold | dog      | 1989-05-13 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

데이터 수정

□ update문을 이용하여 데이터 수정

```
mysql> UPDATE pet set owner='Benny' where name='Puffball';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> SELECT * from pet where owner='Benny';
+-----+-----+-----+-----+-----+-----+
| name   | owner | species | sex | birth      | death      |
+-----+-----+-----+-----+-----+-----+
| Fang    | Benny  | dog      | m   | 1990-08-27 | NULL       |
| Slim    | Benny  | snake    | m   | 1996-04-29 | NULL       |
| Puffball | Benny  | hamster  | f   | 1999-03-30 | NULL       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- update 다음에는 테이블 이름 지정
- set 다음에 수정할 열의 이름과 값을 콤마로 분리하여 나열
- where 다음에는 검색 조건을 지정. where는 생략 가능

레코드 삭제

delete문을 이용하여 데이터 삭제

```
mysql> DELETE from pet where name='Puffball';
Query OK, 1 row affected (0.06 sec)
mysql> SELECT * from pet where owner='Benny';
+-----+-----+-----+-----+-----+-----+
| name | owner | species | sex | birth   | death |
+-----+-----+-----+-----+-----+-----+
| Fang | Benny | dog     | m   | 1990-08-27 | NULL  |
| Slim | Benny | snake   | m   | 1996-04-29 | NULL  |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

- delete from 다음에는 테이블 이름 지정
- where 다음에는 검색 조건을 지정. 위의 예에서는 name 값이 'Puffball'인 레코드 삭제
- where는 생략 가능

JDBC 프로그래밍

DB 연결 설정

■ JDBC 드라이버 로드

```
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

- Class.forName()은 동적으로 자바 클래스 로딩
- MySQL의 JDBC 드라이버 클래스인 *com.mysql.cj.jdbc.Driver* 로드
- 드라이버의 클래스 이름은 DB의 드라이버마다 다를 수 있으므로 JDBC 드라이버 문서 참조할 것
- 자동으로 드라이버 인스턴스를 생성하여 DriverManager에 등록
- 해당 드라이버가 없으면 ClassNotFoundException 발생

자바 응용프로그램과 JDBC의 연결

■ 연결

```
static final String DB_URL = "jdbc:mysql://localhost/sampled";
static final String USER = "root";
static final String PASS = "*****";
try {
    Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
} catch (SQLException e) {
    e.printStackTrace();
}
```

- DriverManager는 자바 어플리케이션을 JDBC 드라이버에 연결시켜주는 클래스로서 getConnection() 메소드로 DB에 연결하여 Connection 객체 반환
- getConnection에서 jdbc: 이후에 지정되는 URL의 형식은 DB에 따라 다르므로 JDBC 문서를 참조
 - MySQL 서버가 같은 컴퓨터에서 동작하므로 서버 주소를 localhost로 지정
 - MySQL의 경우 디폴트로 3306 포트를 사용
 - sampled는 앞서 생성한 DB의 이름
- "root"는 DB에 로그인할 계정 이름이며, ""는 계정 패스워드

예제 : sampledb의 데이터베이스 연결하는 JDBC 프로그램 작성

JDBC를 이용하여 sampledb 데이터베이스에 연결하는 자바 응용프로그램을 작성하라.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class JDBCtest {
    public static void main (String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("DB 연결 완료");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC 드라이버 로드 에러");
        } catch (SQLException e) {
            System.out.println("DB 연결 오류");
        }
    }
}
```

DB 연결 오류

또는

DB 연결 완료

데이터베이스 사용

□ Statement 클래스

- SQL문을 실행하기 위해서는 Statement 클래스를 이용
- 주요 메소드

| 메소드 | 설명 |
|------------------------------------|---|
| ResultSet executeQuery(String sql) | 주어진 SQL문을 실행하고 결과는 ResultSet 객체에 반환 |
| int executeUpdate(String sql) | INSERT, UPDATE, 또는 DELETE과 같은 SQL문을 실행하고, SQL문 실행으로 영향을 받은 행의 개수나 0을 반환 |
| void close() | Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |

- 데이터 검색을 위해 executeQuery() 메소드 사용
- 추가, 수정, 삭제와 같은 데이터 변경은 executeUpdate() 메소드 사용

데이터베이스 사용

□ ResultSet 클래스

- SQL문 실행 결과를 얻어오기 위해서는 ResultSet 클래스를 이용
- 현재 데이터의 행(레코드 위치)을 가리키는 커서(cursor)를 관리
- 커서의 초기 값은 첫 번째 행 이전을 가리킴

| 메소드 | 설명 |
|---------------------------|---------------------------------------|
| boolean first() | 커서를 첫 번째 행으로 이동 |
| boolean last() | 커서를 마지막 행으로 이동 |
| boolean next() | 커서를 다음 행으로 이동 |
| boolean previous() | 커서를 이전 행으로 이동 |
| boolean absolute(int row) | 커서를 지정된 행으로 이동 |
| boolean isFirst() | 첫 번째 행이면 true 반환 |
| boolean isLast() | 마지막 행이면 true 반환 |
| void close() | ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |

데이터베이스 사용

□ 테이블의 모든 데이터 검색

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from student");
```

- Statement의 executeQuery()는 SQL문의 실행하여 실행 결과를 넘겨줌
- 위의 SQL문의 student 테이블에서 모든 행의 모든 열을 읽어 결과를 rs에 저장

□ 특정 열만 검색

```
ResultSet rs = stmt.executeQuery("select name, id from student");
```

- 특정 열만 읽을 경우는 select문을 이용하여 특정 열의 이름 지정

□ 조건 검색

```
rs = stmt.executeQuery("select name, id, dept from student where id='0494013'");
```

- select문에서 where절을 이용하여 조건에 맞는 데이터 검색

데이터베이스 사용

□ 검색된 데이터의 사용

```
while (rs.next()) {
    System.out.println(rs.getString("name"));
    System.out.println(rs.getString("owner"));
    System.out.println(rs.getString("species"));
    System.out.println(rs.getString("sex"));
    System.out.println(rs.getDate("birth"));
    System.out.println(rs.getDate("death"));
}
rs.close();
```

- Statement객체의 executeQuery() 메소드
 - ResultSet 객체 반환
- ResultSet 인터페이스
 - DB에서 읽어온 데이터를 추출 및 조작할 수 있는 방법 제공
- next() 메소드
 - 다음 행으로 이동

데이터베이스 사용

- ResultSet의 getXXX() 메소드
 - 해당 데이터 타입으로 열 값을 읽어옴
 - 인자로 열의 이름이나 인덱스를 줄 수 있음
 - DB 데이터 타입에 해당하는 자바 데이터 타입으로 데이터를 읽어와 함.
 - 모든 데이터 타입에 대해 getString() 메소드로 읽을 수 있으나 사용할 때는 해당 데이터 타입으로 변환해서 사용
- ResultSet에서 모든 데이터를 읽은후 close()를 호출하여 자원 해제

| 메소드 | 설명 |
|-------------------------------------|--|
| Xxx getXxx(String columnName) | ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 이 름에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다. |
| Xxx getXxx(int columnIndex) | Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 인 덱스에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽 는 메소드는 getInt()이다. |

한글 처리 문제

□ 문자열 코드 문제

- MySQL의 문자 집합은 ISO-8859-1
 - Unicode를 사용하는 자바에서 한글이 깨져 출력
 - ISO-8859-1를 Unicode로 변환하여 출력해야 함

```
String ISO_8859_1String = rs.getString("name");  
byte [] UnicodeBytes = ISO_8859_1String.getBytes("ISO-8859-1");  
String UnicodeString = new String(UnicodeBytes);  
System.out.print(UnicodeString);
```

유니코드로 변환

- 반대로 MySQL에서 사용될 한글이 포함된 문자열은 Unicode에서
ISO-8859-1로 변환해야 MySQL에서 정상적으로 처리됨

```
byte [] UnicodeBytes = "홍길동".getBytes();  
String IOS_8859_1String = new String(UnicodeBytes, "ISO-8859-1");  
stmt.executeQuery("select name, id, dept from student where name='"+  
IOS_8859_1String +"'");
```

유니코드를 ISO-8859-1로 변환

예제: 데이터 검색과 출력

앞서 생성한 sampledb의 pet 테이블의 모든 데이터를 출력하는 프로그램과 이름이
"Claws"인 pet의 데이터를 출력하는 프로그램을 작성하시오.

```
import java.io.UnsupportedEncodingException;  
import java.sql.*;  
public class SampleDBPetTest {  
    public static void main (String[] args) {  
        Connection conn;  
        Statement stmt = null;  
        try {  
            Class.forName("com.mysql.cj.jdbc.Driver");  
            conn = DriverManager.getConnection(DB_URL, USER, PASS);  
            System.out.println("DB 연결 완료");  
            stmt = conn.createStatement();  
            printTable(stmt);  
            ResultSet rs = stmt.executeQuery("select name, owner, species, sex, birth, death from pet  
                where owner='Gwen'");  
            printData(rs);  
        } catch (ClassNotFoundException e) {  
            System.out.println("JDBC 드라이버 로드 에러");  
        } catch (SQLException e) {  
            System.out.println("SQL 실행 에러");  
        } catch (UnsupportedEncodingException e) {  
            System.out.println("지원되지 않는 인코딩 타입");  
        }  
    }  
}
```

예제: 데이터 검색과 출력(소스 계속)

```
private static void printTable(Statement stmt) throws SQLException {  
    String sql = "SELECT name, owner, species, sex, birth, death FROM pet";  
    ResultSet rs = stmt.executeQuery(sql);  
    while(rs.next()) {  
        //Retrieve by column name  
        String name = rs.getString("name");  
        String owner = rs.getString("owner");  
        String species = rs.getString("species");  
        String sex = rs.getString("sex");  
        Date birth = rs.getDate("birth");  
        Date death = rs.getDate("death");  
        //Display values  
        System.out.print("name: " + name);  
        System.out.print(", owner: " + owner);  
        System.out.print(", species: " + species);  
        System.out.print(", sex: " + sex);  
        System.out.print(", birth: " + birth);  
        System.out.println(", death: " + death);  
    }  
    rs.close();  
}
```

예제: 데이터 검색과 출력(소스 계속)

```
private static void printData(ResultSet rs) throws SQLException {
    while(rs.next()) {
        //Retrieve by column name
        String name = rs.getString("name");
        String owner = rs.getString("owner");
        String species = rs.getString("species");
        String sex = rs.getString("sex");
        Date birth = rs.getDate("birth");
        Date death = rs.getDate("death");
        //Display values
        System.out.print("name: " + name);
        System.out.print(", owner: " + owner);
        System.out.print(", species: " + species);
        System.out.print(", sex: " + sex);
        System.out.print(", birth: " + birth);
        System.out.println(", death: " + death);
    }
}
```

데이터의 변경

□ 레코드 추가

```
stmt.executeUpdate("INSERT INTO pet VALUES ('Puffball', 'Diane', 'hamster', 'f',  
'1999-03-30', NULL);WrWn");
```

- DB에 변경을 가하는 조작은 executeUpdate() 메소드 사용
- SQL문 수행으로 영향을 받은 행의 개수 반환

□ 데이터 수정

```
stmt.executeUpdate("UPDATE pet set owner = 'Benny' where name='Puffball'");
```

- where문의 MySQL에서 처리되므로 문자열을 Unicode에서 ISO-8859-1로 변환에 주의

□ 데이터 삭제

```
stmt.executeUpdate("DELETE from pet where name='Puffball'");
```