1. JDBC 를 배워야 할까요?

JDBC 는 Java Database Connectivity 의 **약자** 로 Java 프로그래밍 언어와 광범위한 데이터베이스 간의 데이터베이스 독립적 연결을 위한 표준 Java API.

JDBC 라이브러리에는 일반적으로 데이터베이스 사용과 관련된 아래 언급된 각 작업에 대한 API 가 포함되어 있습니다.

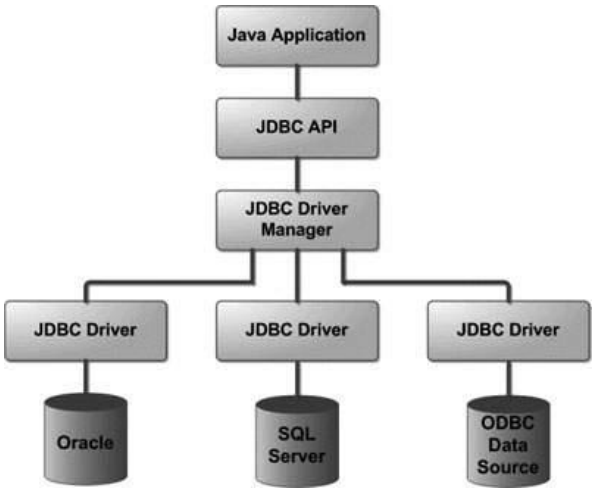
- 데이터베이스에 연결
- SQL 문 만들기
- 데이터베이스에서 SQL 쿼리 실행
- 결과 레코드 보기 및 수정

2. JDBC 의 응용

기본적으로 JDBC 는 기본 데이터베이스에 대한 이식 가능한 액세스를 허용하는 완전한 인터페이스 세트를 제공하는 사양입니다. Java 는 다음과 같은 다양한 유형의 실행 파일을 작성하는 데 사용할 수 있습니다.

- 자바 애플리케이션
- 자바 서블릿
- 엔터프라이즈 JavaBeans(EJB).

3. JDBC 아키텍처



4. 공통 JDBC 구성 요소

JDBC API 는 다음 인터페이스와 클래스를 제공

- DriverManager - 이 클래스는 데이터베이스 드라이버 목록을 관리
- 드라이버 - 이 인터페이스는 데이터베이스 서버와의 통신 처리
- 연결 - 데이터베이스에 접속하기 위한 모든 방법이 정의된 인터페이스
- Statement - 이 인터페이스에서 생성된 개체를 사용하여 SQL 문을 데이터베이스에 제출
- ResultSet – 이 개체는 Statement 개체를 사용하여 SQL 쿼리 실행 후 데이터베이스에서 검색된 데이터 보유
- SQLException - 이 클래스는 데이터베이스 응용 프로그램에서 발생하는 모든 오류 처리

5. JDBC - SQL 구문

데이터베이스 생성

데이터베이스 삭제

테이블 만들기

테이블 삭제

데이터 삽입

데이터 검색

데이터 변경

데이터 삭제

6. JDBC - 환경 설정

자바 설치

데이터베이스 설치: Oracle XE 11g

데이터베이스 사용자 생성(데이터베이스 생성)

테이블 만들기

데이터 레코드 생성

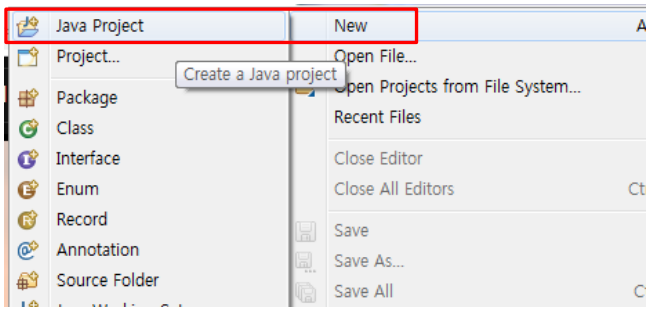
오라클 데이터베이스 버전별 Jdbc 파일

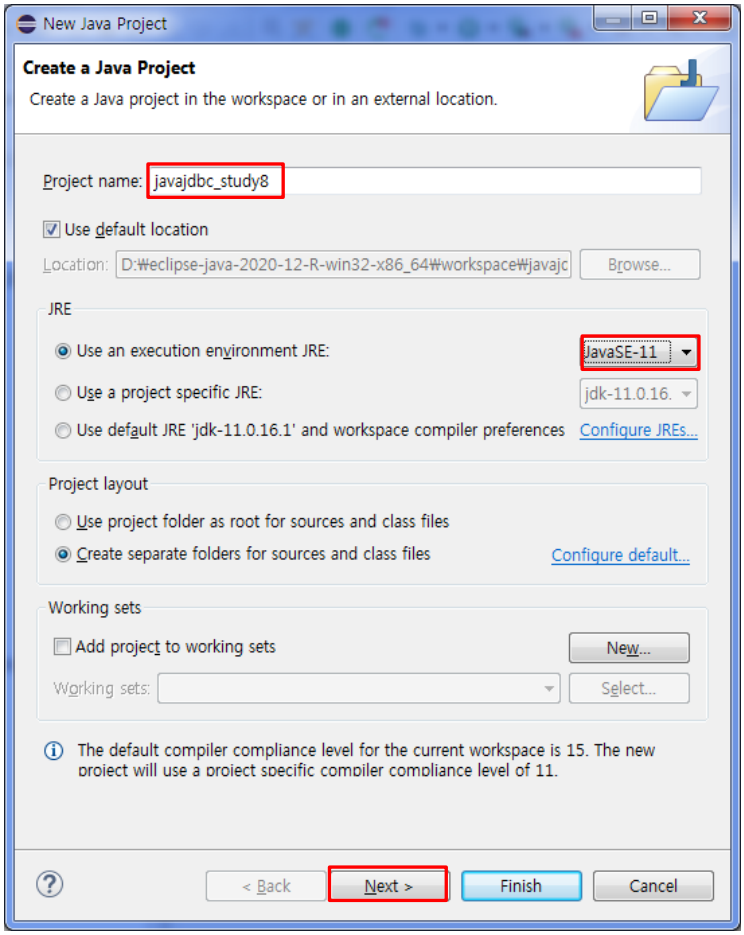
Oracle Database version	JDBC jar files specific to the release
21.1	ojdbc11.jar: JDK11, JDK12, JDK13, JDK14, JDK15 ojdbc8.jar: JDK8, JDK11, JDK12, JDK13, JDK14, JDK15
19.x	ojdbc10.jar: JDK10, JDK11 ojdbc8.jar: JDK8, JDK9, JDK11
18.3	ojdbc8.jar: JDK8, JDK9, JDK10, JDK11
12.2 or 12cR2	ojdbc8.jar: JDK8
12.1 or 12cR1	Ojdbc7.jar: JDK7, JDK8 Ojdbc6.jar: JDK6
11.2 or 11gR2	Ojdbc6.jar: JDK6, JDK7, JDK8 Ojdbc5.jar: JDK5

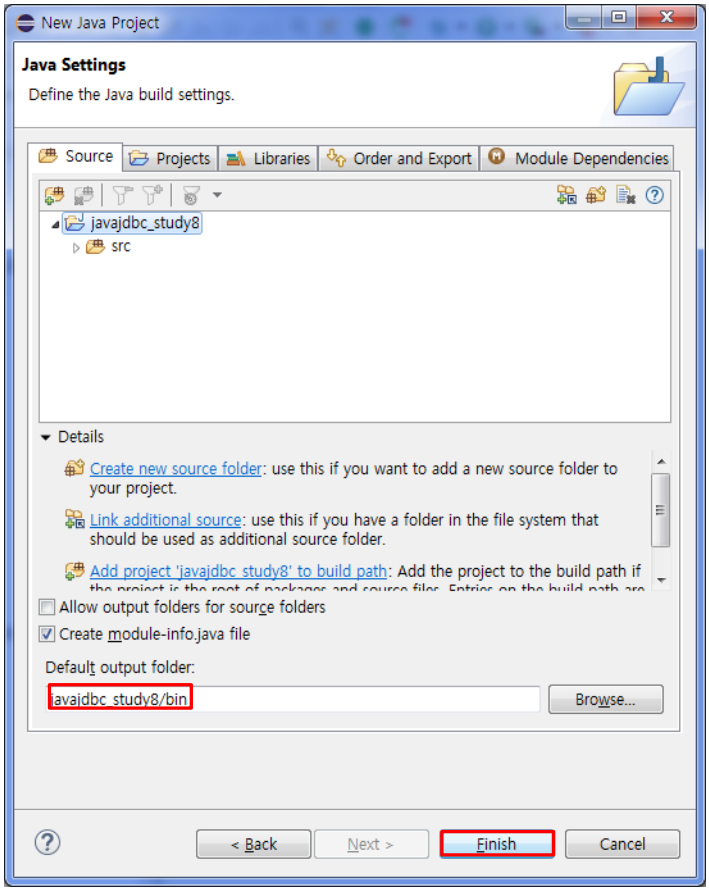
7. JDBC 애플리케이션 생성

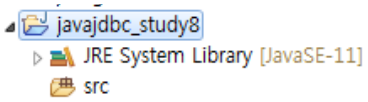
가. 구축 전 오라클 설치 및 JDBC 드라이버 설치

- 프로젝트 만들기

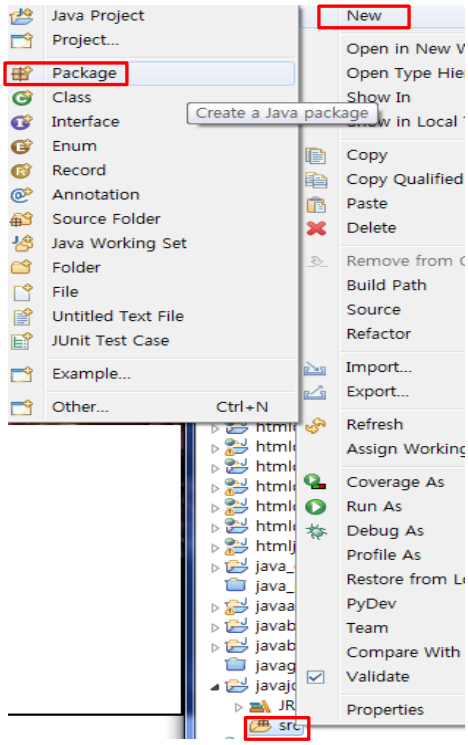


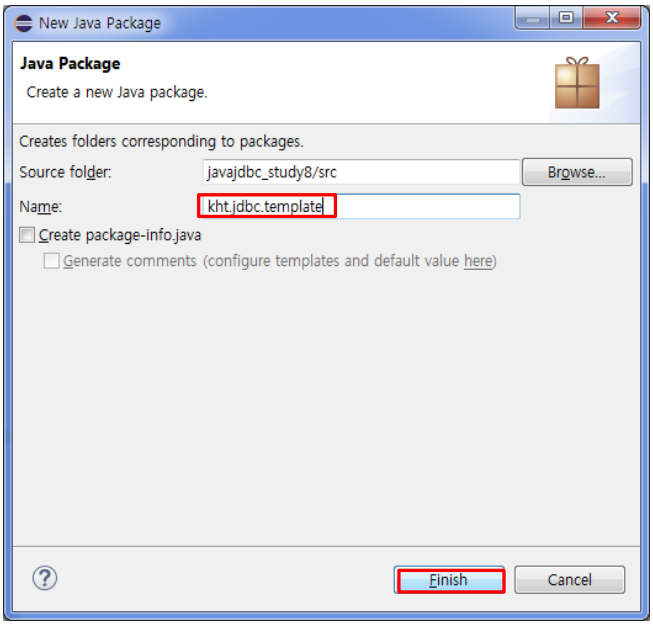


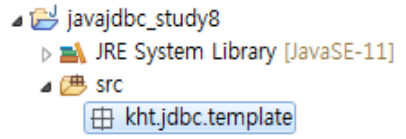

New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
Module Dependencies
javajdbc_study8
src
Details
Create new source folder: use this if you want to add a new source folder to your project.
Link additional source: use this if you have a folder in the file system that should be used as additional source folder.
Add project 'javajdbc_study8' to build path: Add the project to the build path if
Allow output folders for source folders
Create module-info.java file
Default output folder:
javajdbc_study8/bin
Browse...
< Back
Next >
Finish
Cancel


javajdbc_study8
JRE System Library [JavaSE-11]
src

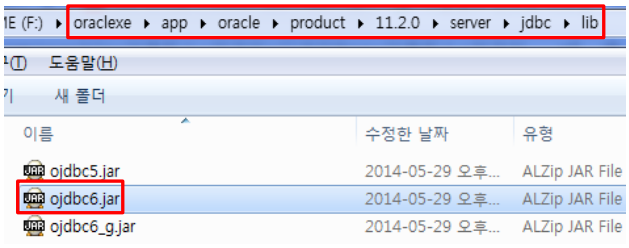
- 패키지 만들기

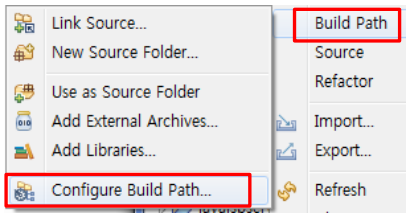

Java Project
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Example...
Other... Ctrl+N
Create a Java package
New
Open in New W
Open Type Hie
Show In
Copy
Copy Qualified
Paste
Delete
Remove from
Build Path
Source
Refactor
Import...
Export...
Refresh
Assign Working
Coverage As
Run As
Debug As
Profile As
Restore from L
PyDev
Team
Compare With
Validate
Properties

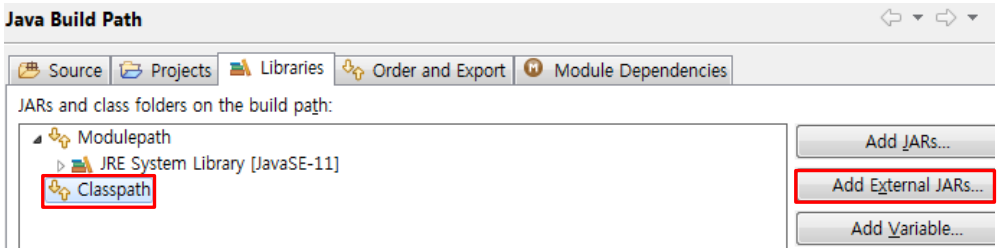

New Java Package
Java Package
Create a new Java package.
Creates folders corresponding to packages.
Source folder:
javajdbc_study8/src
Browse...
Name:
kht.jdbc.template
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel

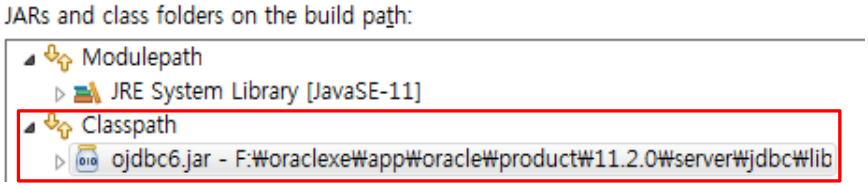


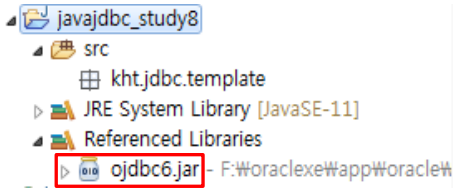
- 오라클 JDBC 드라이버 설정











나. JDBC 애플리케이션 구축 6 단계

- 패키지 가져오기 - JDBC 클래스가 포함된 패키지를 포함. 대부분의 경우 *import java.sql.** 사용
- 연결 열기 - *DriverManager.getConnection()* 메서드로 데이터베이스와의 물리적 연결을 나타내는 Connection 객체 생성
- 쿼리 실행 - SQL 문을 작성하고 데이터베이스에 제출
- 결과 집합에서 데이터 추출 - 결과 집합에서 데이터를 검색에 적절한 *ResultSet.getXXX() 메서드를 사용*
- 환경 정리 - 모든 데이터베이스 리소스를 명시적으로 닫아야 함

8. JDBC - 데이터 베이스 연결

가. 적절한 드라이버를 설치한 후 JDBC를 사용하여 데이터베이스 연결을 설정

나. JDBC 연결 설정 프로그래밍의 4단계
- JDBC 패키지 가져오기
- JDBC 드라이버 등록: JVM이 원하는 드라이버 구현물을 메모리에 로드하여 JDBC 요청을 이행할 수 있도록 함
- 데이터베이스 URL: 연결하려는 데이터베이스를 가리키는 올바른 형식 주소 생성
- 연결 개체 만들기: DriverManager 개체의 getConnection() 메소드에 대한 호출로 실제 데이터베이스 연결을 설정

다. JDBC 패키지 가져오기
- import java.sql.*; // 표준 JDBC 프로그램을 위한 패키지

라. JDBC 드라이버 등록 - 방법1

- 가장 일반적인 동적인 등록 방법
- 자바의 Class.forName() 메소드 사용
- JDK7부터 생략 가능한 단계
- 예

```
try {
    Class.forName( “oracle.jdbc.driver.OracleDriver” );
}
catch(ClassNotFoundException ex)
{
    System.out.println( “Error: unable to load driver class!” );
    System.exit(1);
}
```

마. 데이터베이스 URL
- DriverManager.getConnection() 메소드 사용
- 데이터베이스별 연결 URL

RDBMS	JDBC 드라이버 이름	URL 형식
MySQL	com.mysql.jdbc.드라이버	Jdbc:mysql://호스트이름/DB 이름
Oracle	com.jdbc.driver.OracleDriver	Jdbc:oracle:thin:@호스트이름:포트번호:DB이름
DB2	com.ibm.db2.jdbc.net.DB2Driver	Jdbc:db2:호스트이름:포트번호/DB 이름
Sybase	com.sybase.jdbc.SybDriver	Jdbc:sysbase:Tds:호스트이름:포트번호/DB 이름

바. 연결 개체 만들기

```
String URL = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
String USER = "oracle_test";
String PASS = "woseven"
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

```
jdbc:oracle:driver:username/password@database
String URL = "jdbc:oracle:thin:oracle_test/woseven@127.0.0.1:1521:xe";
Connection conn = DriverManager.getConnection(URL);
```

사. JDBC 연결 닫기
conn.close();

9. JDBC - Statement(정적SQL)와 PreparedStatement(동적SQL)

가. Statement: 정적 SQL 문장을 실행하는데 사용하는 개체
- 예

```
Statement stmt = null;
try {
    stmt = conn.createStatement();
    …
}
catch(SQLException) { }
finally { }
```

나. SQL 문장을 실행 방법
- boolean execute(String SQL): 실행 가능 여부 확인 -> DDL
- int executeUpdate(String SQL): 영향을 받는 행 수 반환 -> INSERT/UPDATE/DELET
- ResultSet executeQuery(String SQL): ResultSet 개체 반환 -> SELECT

다. Statement 객체 종료: stmt.close();

라. PreparedStatement: 동적 SQL 문장을 실행하는데 사용하는 객체
- 예

```
PreparedStatement pstmt = null;
try {
    String SQL = “update empKht Set name = ? WHERE emp_id = ?” ;
    Pstmt = conn.preparedStatement(SQL);
```

- ?: 매개변수 마커: set데이터타입() 메소드를 이용해 값을 매개변수에 바인딩

마. PreparedStatement 객체 종료: pstmt.close();