

JAVA 프로그래밍

Chapter 19 JDBC

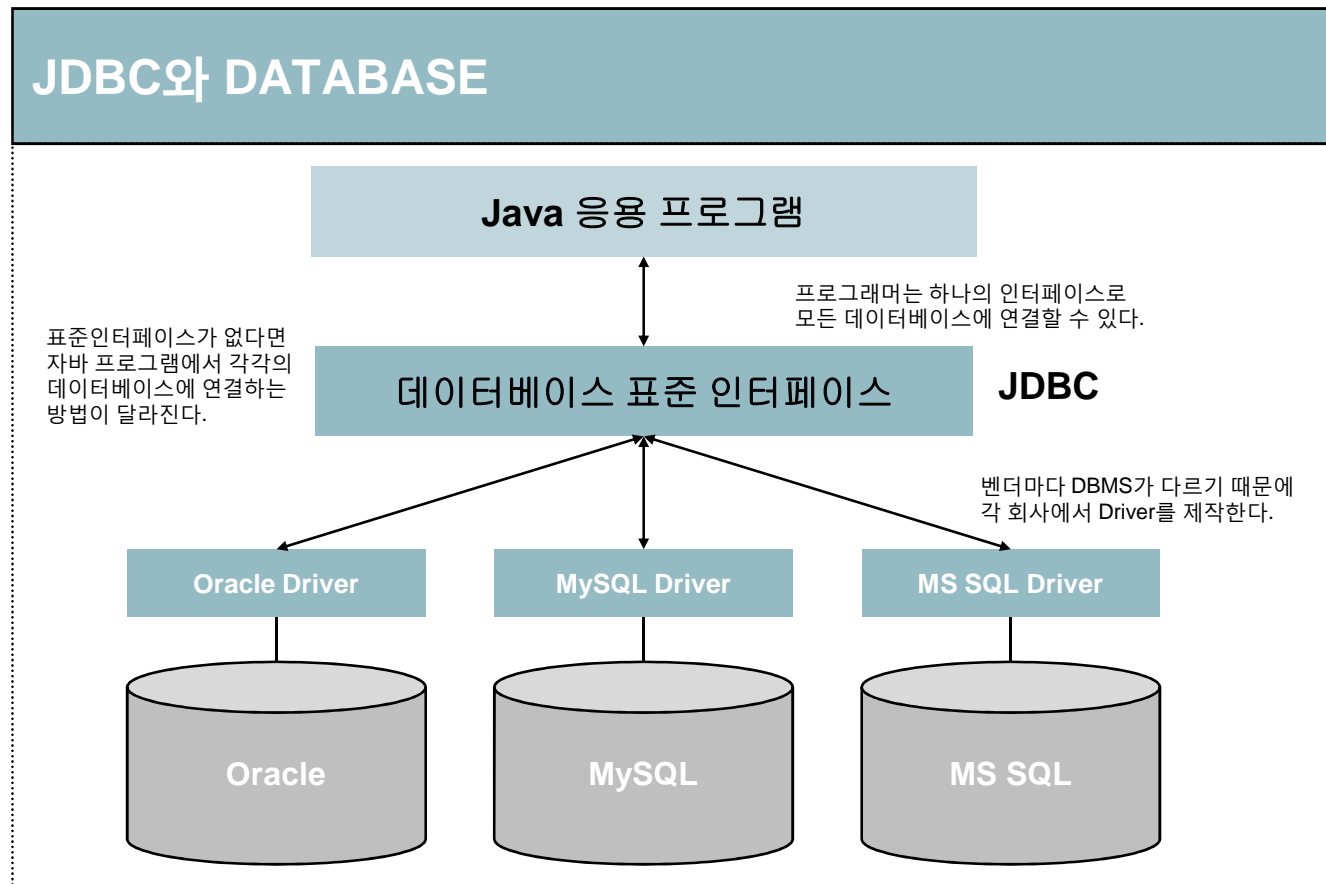
목차

- ❖ JDBC 란
- ❖ DBMS 설치
- ❖ JDBC를 통한 데이터베이스 연결방법
- ❖ JDBC 주요 클래스
- ❖ DAO & DTO
- ❖ Connection Factory 기법
- ❖ Connection Pooling

JDBC(Java Database Connectivity)

❖ JDBC(Java Database Connectivity)

- JDBC란 데이터베이스에 연결 및 작업을 하기 위한 자바 표준 인터페이스(API)이다.



MySQL 설치

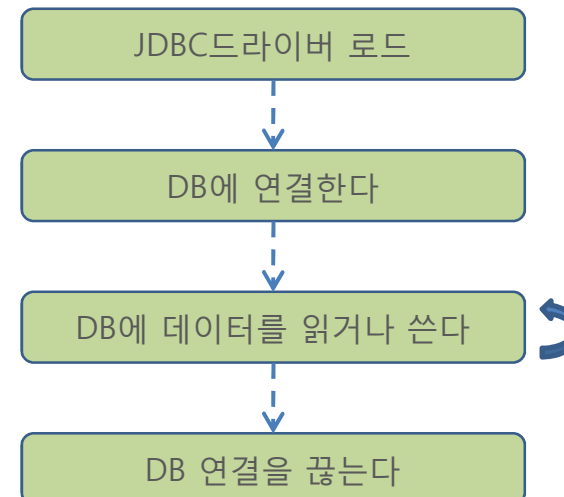
- ❖ MySQL 다운로드 사이트
 - <http://www.mysql.com>
- ❖ JDBC 드라이버 패키지 파일
 - Connector/J 5.x라는 이름으로 등록되어 있으며 zip 또는 tar 파일 형태로 묶여 있다.
 - 압축해제디렉터리\mysql-connector-java-5.x.x-bin.jar
- ❖ 클래스 패스 지정
 - mysql-connector-java-5.x.x-bin.jar 파일을 [자바 설치 디렉터리]\jre\lib\ext 디렉터리에 두면 자동으로 클래스 패스가 지정된다.
- ❖ MySQL Connector/J 5.x 드라이버 로딩
 - `Class.forName("com.mysql.jdbc.Driver");`
- ❖ MySQL Connector/J 5.x URL
 - `String url = "jdbc:mysql://localhost:3306/dbname";`

Oracle 설치

- ❖ Oracle 공식 사이트
 - <http://www.oracle.com>
- ❖ JDBC 드라이버 패키지 파일
 - 오라클 설치 폴더에서 가져오기
 - C:\oracle\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar
 - 오라클 다운로드 사이트에서 가져오기
 - <http://www.oracle.com/technetwork/database/application-development/jdbc/downloads/download-3320542.html>
- ❖ 클래스 패스 지정
 - ojdbc6.jar 파일을 [자바 설치 디렉터리]\jre\lib\ext 디렉터리에 두면 자동으로 클래스 패스가 지정된다.
 - 혹은 이클립스에서 lib 폴더 추가 후 classpath를 설정해서 한다.
- ❖ 드라이버 로딩
 - `Class.forName("oracle.jdbc.driver.OracleDriver");`
- ❖ URL
 - `String url = "jdbc:oracle:thin:@127.0.0.1:1521:SID";`

JDBC를 통한 데이터베이스 연결방법

- ❖ 1. 데이터 베이스 연결을 위한 드라이버 생성
 - `Class.forName("com.mysql.jdbc.Driver")`으로 드라이버 로딩
 - `Class.forName("oracle.jdbc.driver.OracleDriver")`으로 드라이버 로딩
- ❖ 2. 연결을 관리하는 Connection 객체 생성
 - DriverManager의 `getConnection()`을 이용해서 Connection 객체 생성
- ❖ 3. SQL 문 수행.
 - 작업을 처리할 Statement, PreparedStatement, CallableStatement 객체 생성
 - 쿼리 실행 및 결과 사용
- ❖ 4. 연결 종료
 - `Connection close()`



JDBC 주요 클래스

Connection 인터페이스

❖ DBMS와 연결을 관리하는 클래스

반환형	메소드	설명
void	close()	Connection 객체를 해제한다.
	commit()	트랜잭션으로 설정된 모든 자원을 커밋한다.
Statement	createStatement()	SQL문을 전송할 수 있는 Statement 객체를 생성한다.
	createStatement(int resultSetType, int resultSetConcurrency)	매개변수로 SQL문을 전송할 수 있는 Statement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 Statement 객체의 기능이 달라진다.
boolean	getAutoCommit()	Connection 객체의 현재 auto-commit 상태를 반환한다.
CallableStatement	prepareCall(String sql)	SQL문 전송과 Store Procedure를 호출할 수 있는 CallableStatement 객체를 생성한다.
	prepareCall(String sql, int resultSetType, int resultSetConcurrency)	매개변수로 CallableStatement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 CallableStatement 객체의 기능이 달라진다.

Connection 인터페이스(계속)

❖ DBMS와 연결을 관리하는 클래스

반환형	메소드	설명
PreparedStatement	prepareStatement(String sql)	SQL문을 전송할 수 있는 PreparedStatement 객체를 생성한다.
	prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	매개변수로 PreparedStatement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 PreparedStatement 객체의 기능이 달라진다.
void	rollback()	현재 트랜잭션에 설정된 모든 변화를 되돌린다.
	rollback(Savepoint savepoint)	Savepoint로 설정된 이후의 모든 변화를 되돌린다.
Savepoint	setSavepoint(String name)	현재 트랜잭션에서 name으로 Savepoint를 설정한다.

Connection 방법

- ❖ 1. 드라이버 로딩
 - `Class.forName("oracle.jdbc.driver.OracleDriver");`
- ❖ 2. `java.sql.DriverManager` 클래스를 이용해서 `Connection` 객체 생성
 - `String url = "jdbc:oracle:thin:@localhost:1521:xe";`
 - `String url = "jdbc:mysql://localhost:3306/dbname";`
 - `String user = "user";`
 - `String password = "password";`
 - `Connection conn = DriverManager.getConnection(url, user, password);`
- ❖ 3. 작업
 - 쿼리와 명령 처리
- ❖ 4. `Connection` 닫기
 - `conn.close();`

Statement

❖ Statement 의 종류

- Statement
- PreparedStatement
- CallableStatement

❖ Statement

- 정적쿼리 처리에 유리하다.

❖ PreparedStatement

- 객체를 생성하면서 기준 Query를 생성한다.
- setter()메소드를 가지고 조건만 바꾸면서 동적 쿼리를 수행한다.

❖ Statement 의 의미

- 자바에서 사용되는 3가지 종류의 스테이트먼트들은 데이터베이스로 쿼리를 담아서 보내는 그릇 정도로 생각하면 된다. 즉 스테이트먼트에 쿼리를 실어 데이터베이스로 보내버리면 데이터베이스에서 처리되는 것이다. 이 때 한번 사용하고 버리는 그릇은 Statement이며, 재사용 가능한 그릇은 PreparedStatement이다.

❖ CallableStatement

- 데이터베이스 내의 스토어드 프로시저(Stored Procedure)를 호출할 수 있는 스테이트먼트

Statement 인터페이스

- ❖ SQL문을 전송을 담당하는 인터페이스이다.
- ❖ Statement 객체는 Connection 인터페이스의 `createStatement()` 메서드를 사용하여 얻어 올 수 있다.
- ❖ `executeQuery(String sql)` : select SQL 구문일 경우
- ❖ `executeUpdate(String sql)` : insert, update, delete SQL 구문일 경우
- ❖ `execute(String sql)` : SQL문을 알지 못하는 경우

Statement 인터페이스

반환형	메소드	설명
void	addBatch(String sql)	Statement 객체에 SQL문을 추가한다. 이 메서드를 이용해서 SQL의 일괄처리를 할 수 있다.
	clearBatch()	Statement 객체에 모든 SQL문을 비운다.
	close()	Statement 객체를 해제한다.
boolean	execute(String sql)	매개변수인 SQL문을 수행한다. 만약, 수행한 결과가 ResultSet 객체를 반환하면 true, 어떠한 결과도 없거나, 갱신된 숫자를 반환하면 false를 반환한다.
int[]	executeBatch()	Statement 객체에 추가된 모든 SQL문을 일괄처리한다. 일괄처리된 각각의 SQL문에 대한 결과값을 int[]로 반환한다.
ResultSet	executeQuery(String sql)	매개변수인 SQL문을 수행하고 ResultSet 객체를 반환한다.
int	executeUpdate(String sql)	매개변수인 SQL문을 수행한다. SQL문은 INSERT문, UPDATE문, CREATE문, DROP문 등을 사용한다.
ResultSet	getResultSet()	ResultSet 객체를 반환한다.

ResultSet 인터페이스

❖ ResultSet

- ResultSet은 SQL문에 대한 결과를 처리할 수 있는 객체이다.
- Statement 인터페이스의 executeQuery() 메서드를 실행한 결과로 ResultSet 객체를 리턴 받는다.
- 모든 데이터를 한번에 가져올 수 없기 때문에 cursor의 개념을 가지고 있다.
- cursor란 ResultSet 객체가 가져올 수 있는 행을 지정해 준다.
- 처음 커서의 위치는 결과물(필드)에 위치하지 않기 때문에 cursor를 이동해야 한다.
- 커서를 이동하는 메서드가 ResultSet 의 next() 메서드이다.
- next() 메서드의 리턴 타입은 boolean 인데 이는 다음 행의 결과물(필드)이 있으면 true, 없으면 false를 리턴 한다.
- ResultSet 객체가 결과물(필드)을 가져올 수 있는 행으로 이동이 되었다면 이제는 실제 결과물(필드)을 가져와야 한다.
- ResultSet 인터페이스에는 결과물(필드)을 가져오는 수많은 메서드(getXXX())를 제공한다.
- getXXX() 메서드는 oracle의 자료형 타입에 따라 달라지게 된다.

ResultSet 인터페이스

반환형	메소드	설명
boolean	absolute(int row)	ResultSet 객체에서 매개변수 row로 커서를 이동한다. 만약, 매개변수 row로 커서를 이동할 수 있으면 true, 그렇지 않으면 false를 반환한다.
void	afterLast()	ResultSet 객체에서 커서를 마지막 row 다음으로 이동한다.
	beforeFirst()	ResultSet 객체에서 커서를 처음 row 이전으로 이동한다.
boolean	last()	ResultSet 객체에서 커서를 마지막 row 로 이동한다. 만약, ResultSet에 row가 있다면 true, 그렇지 않으면 false를 반환한다.
	next()	ResultSet 객체에서 현재 커서에서 다음 row로 커서를 이동한다. 만약, ResultSet에 다음 row가 있다면 true, 그렇지 않으면 false를 반환한다.
	previous()	ResultSet 객체에서 현재 커서에서 이전 row로 커서를 이동한다. 만약, ResultSet에 다음 row가 있다면 true, 그렇지 않으면 false를 반환한다.
void	close()	ResultSet 객체를 해체한다.

ResultSet 인터페이스

반환형	메소드	설명
boolean	first()	ResultSet 객체에서 커서를 처음 row로 이동한다. 만약 ResultSet에 row가 있다면 true, 그렇지 않으면 false를 반환한다.
InputStream	getBinaryStream(int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 InputStream으로 반환한다.
	getBinaryStream(String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 InputStream으로 반환한다.
Blob	getBlob(int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 Blob으로 반환한다.
	getBlob(String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 Blob으로 반환한다.
byte	getBytes(int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 byte으로 반환한다.
	getBytes(String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 byte으로 반환한다.

ResultSet 인터페이스

반환형	메소드	설명
Clob	getClob (int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 Clob으로 반환한다.
	getClob (String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 Clob으로 반환한다.
double	getDouble (int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 double으로 반환한다.
	getDouble (String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 double으로 반환한다.
int	getInt (int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 int으로 반환한다.
	getInt (String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 int으로 반환한다.
String	getString (int columnIndex)	ResultSet 객체의 현재 row에 있는 columnIndex의 값을 String으로 반환한다.
	getString (String columnName)	ResultSet 객체의 현재 row에 있는 columnName의 값을 String으로 반환한다.

PreparedStatement 인터페이스

- ❖ PreparedStatement는 SQL문을 작성할 때 컬럼 값을 실제로 지정하지 않고, 변수 처리 함으로서 DBMS를 효율적으로 사용한다.
- ❖ PreparedStatement의 SQL문은 SQL문의 구조는 같은데 조건이 수시로 변할 때 조건의 변수처리를 "?" 하는데 이를 바인딩 변수라 한다.
- ❖ 바인딩 변수는 반드시 컬럼 명이 아닌 컬럼 값이 와야 한다는 것이다.
- ❖ 바인딩 변수의 순서는 "?" 의 개수에 의해 결정이 되는데 시작 번호는 1 부터 시작하게 된다.
- ❖ 바인딩 변수에 값을 저장하는 메서드는 오라클의 컬럼 타입에 따라 지정해 주면 된다. 전에 ResultSet의 getXXX() 메서드와 유사하게 PreparedStatement 인터페이스에는 바인딩 변수에 값을 저장하는 setXXX() 메서드를 제공하고 있다.

CallableStatement-Return

- ❖ 스토어드 프로시저(Stored Procedure)
 - 데이터베이스 내에서 사용되어지는 함수
- ❖ CallableStatement의 특징
 - 데이터베이스의 스토어드 프로시저를 실행시킬 때 사용
 - 속도, 코드의 독립성, 보안성에 상당한 이점이 있음.
- ❖ CallableStatement의 생성
 - `CallableStatement cs = conn.prepareCall("{call sp_getAgeSelector(?,?)})");`
 - `cs.setString(1,"홍길동");`
- ❖ 출력용 반환 타입 세팅
 - `cs.registerOutParameter(2, java.sql.Types.INTEGER);`
- ❖ StoredProcedure 실행
 - `cs.execute();`
 - `cs.getInt(2);`

DAO & DTO

❖ DAO(Data Access Object)

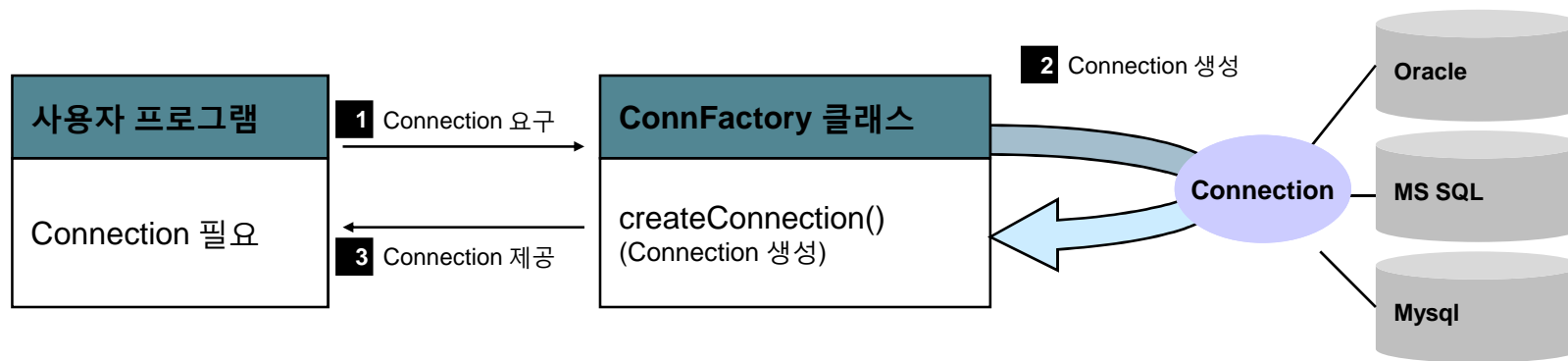
- 데이터베이스와 연계하여 처리할 프로그램을 정규화해 둔 클래스
- 입력, 수정, 삭제, 검색 등의 작업을 미리 캡슐화해 두어 사용하기 편하도록 만든 형태.

❖ DTO(Data Transfer Object)

- 데이터를 하나의 객체로 관리할 목적으로 만들어 둔 클래스의 객체
- Domain DTO와 Custom DTO로 구별할 수 있다.
- setter()와 getter() 메서드를 갖는다.

Connection Factory 기법

Connection Factory



Connection Pooling

❖ Connection Pool

- 미리 컨넥션을 일정 수만큼 생성시킨 뒤 컨넥션을 빌려주고 다시 반환 받는 형식으로 컨넥션을 관리한다.
- 컨넥션 풀에서 필요한 만큼 Connection을 미리 생성
- Connection이 필요할 때 컨넥션 풀에서 빌려서 사용
- 사용이 끝난 다음에는 Connection을 컨넥션 풀에 반환

❖ Connection Pooling 기법의 장점

- Connection Pooling 기법을 이용하면 데이터베이스에 연결하는 시간을 절약할 수 있다.

Connection Pooling

