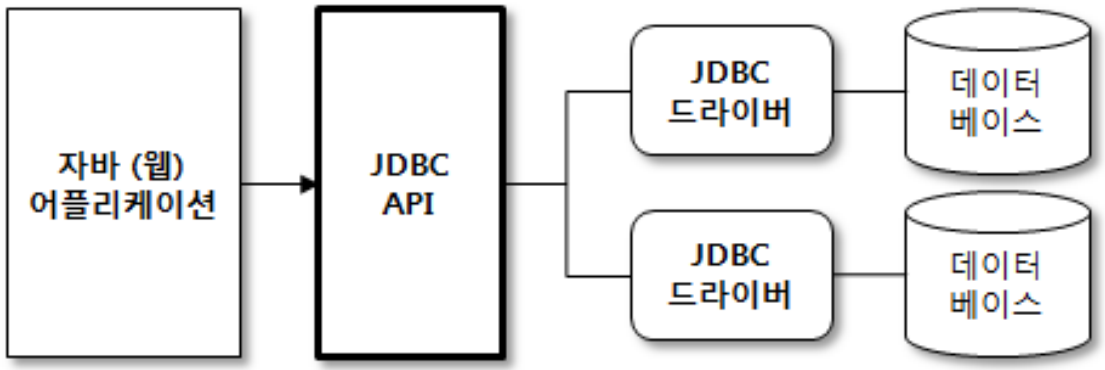
06 JDBC 프로그래밍

Contents

- JDBC 프로그래밍
- 트랙잭션(Transaction) 처리
- 데이터베이스 커넥션 풀 사용하기

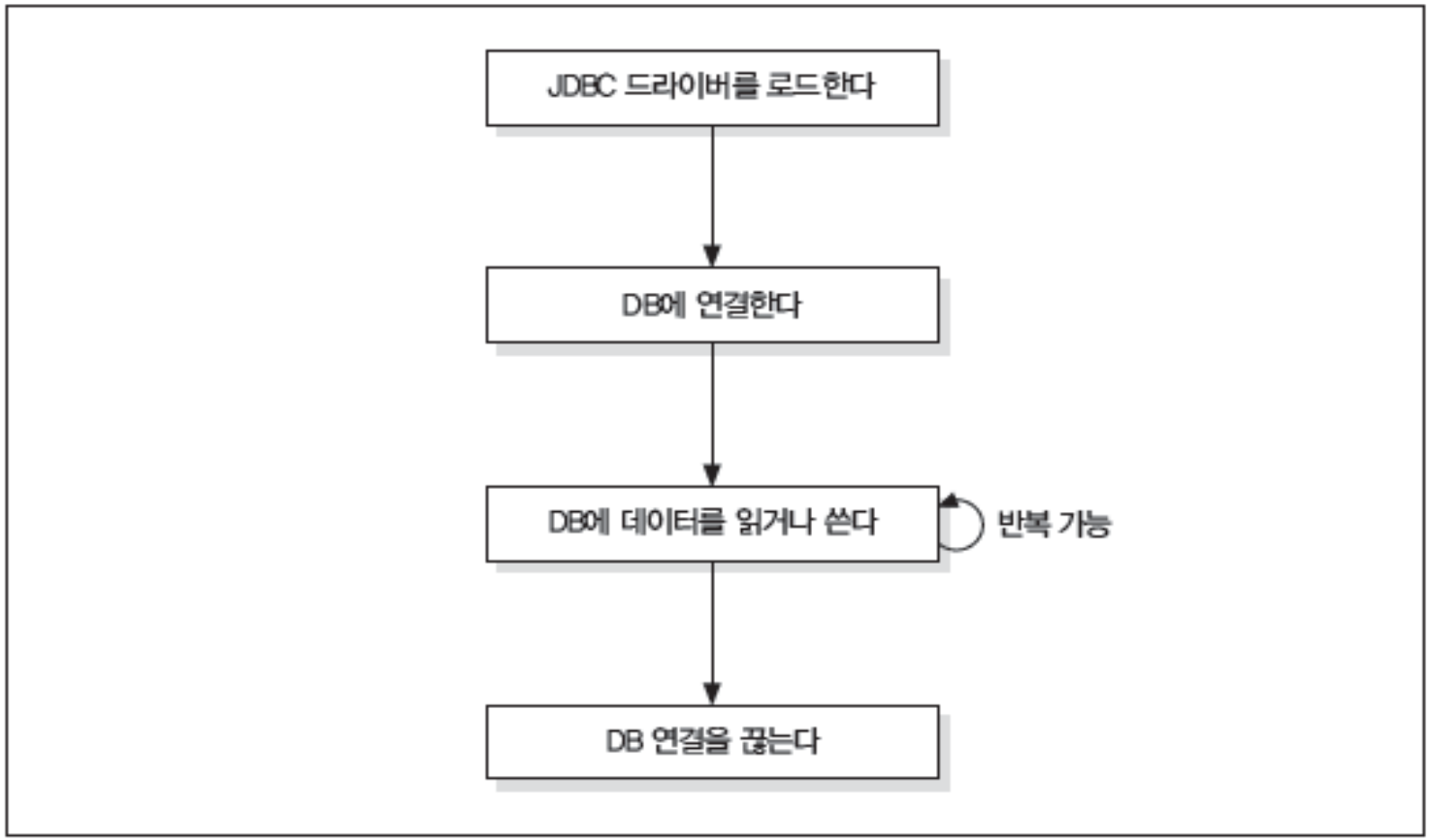
1. JDBC 프로그래밍

- Java Database Connectivity
- 자바에서 DB 프로그래밍을 하기 위해 사용되는 API
- JDBC API 사용 어플리케이션의 기본 구성



- JDBC 드라이버 : 각 DBMS에 알맞은 클라이언트
  - 보통 jar 파일 형태로 제공

1. JDBC 프로그래밍



1. JDBC 프로그래밍

```
// 1. JDBC 드라이버 로딩
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection conn = null; Statement stmt = null; ResultSet rs = null;
try {
    // 2. 데이터베이스 연결 - DB커넥션 생성
    DriverManager.getConnection(
        "jdbc:orcle:thin:@서버ip:1521:SID", "사용자ID", "비밀번호");
    // 3. Statement, PreparedStatement 생성
    stmt = conn.createStatement();
    // 4. 쿼리를 실행하여 결과 받기
    rs = stmt.executeQuery("select * from product");
    // 5. 쿼리 실행 결과 출력
    while(rs.next()) {
        String name = rs.getString(1);
    }
} catch(SQLException ex) {
    ex.printStackTrace();
} finally {
    // 6. 사용한 Statement 종료
    if (rs != null) try { rs.close(); } catch(SQLException ex) {}
    if (stmt != null) try { stmt.close(); } catch(SQLException ex) {}
    // 7. 커넥션 종료
    if (conn != null) try { conn.close(); } catch(SQLException ex) {}
}
```

1. JDBC 프로그래밍

```
// 1. JDBC 드라이버 로딩
Class.forName("com.mysql.jdbc.Driver");
Connection conn = null; Statement stmt = null; ResultSet rs = null;
try {
    // 2. 데이터베이스 연결 - DB커넥션 생성
    conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/mall", "user", "pass");
    // 3. Statement, PreparedStatement 생성
    stmt = conn.createStatement();
    // 4. 쿼리를 실행하여 결과 받기
    rs = stmt.executeQuery("select * from product");
    // 5. 쿼리 실행 결과 출력
    while(rs.next()) {
        String name = rs.getString(1);
    }
} catch(SQLException ex) {
    ex.printStackTrace();
} finally {
    // 6. 사용한 Statement 종료
    if (rs != null) try { rs.close(); } catch(SQLException ex) {}
    if (stmt != null) try { stmt.close(); } catch(SQLException ex) {}
    // 7. 커넥션 종료
    if (conn != null) try { conn.close(); } catch(SQLException ex) {}
}
```

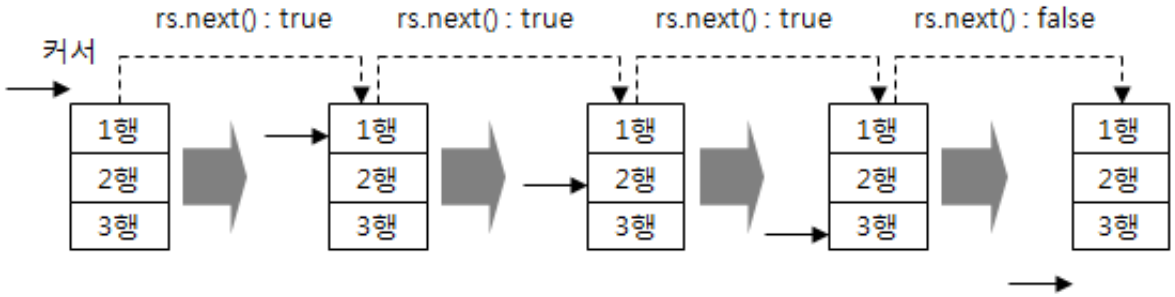
1. JDBC 프로그래밍

❖ Statement, PreparedStatement 제공하는 메소드

- executeQuery(String query) – SELECT 쿼리를 실행
- executeUpdate(String query) – INSERT, UPDATE, DELETE 쿼리를 실행

❖ ResultSet 동작

- next() 메서드로 데이터 조회 여부 확인

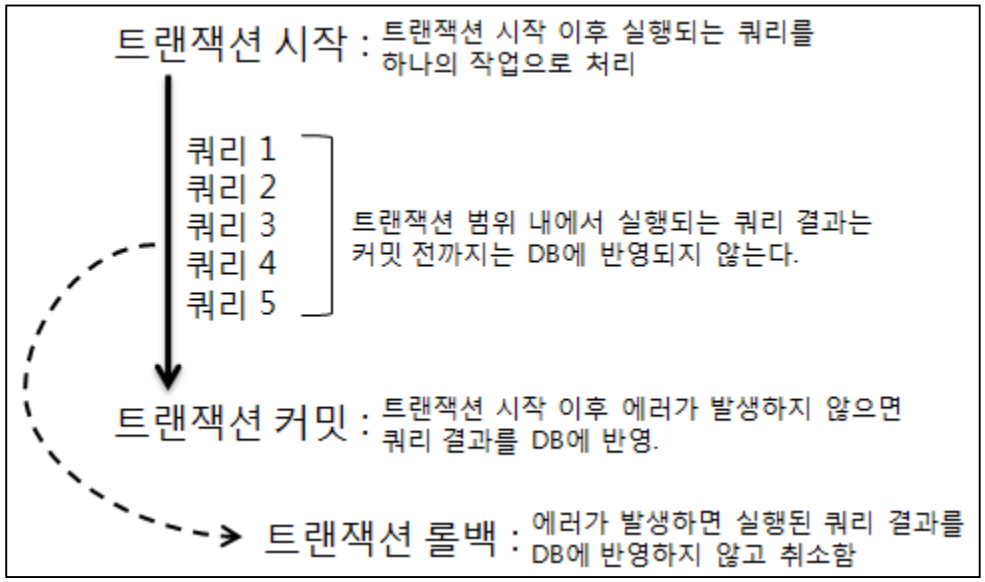


❖ ResultSet의 데이터 조회 위한 메소드(교재 p320 참고)

- getString()
- getInt(), getLong(), getFloat(), getDouble()
- getTimestamp(), getDate(), getTime()

2. 트랜잭션(Transaction)

- 하나의 작업을 완료하는데 있어 여러 개의 쿼리가 필요한 경우 데이터 무결성을 보장하기 위해 필요한 여러 쿼리를 하나의 작업 단위로 처리하는 것 – 트랜잭션



- 트랜잭션 구현 방법: 오토 커밋 해제, JTA(Java Transaction API)

2. 트랜잭션(Transaction)

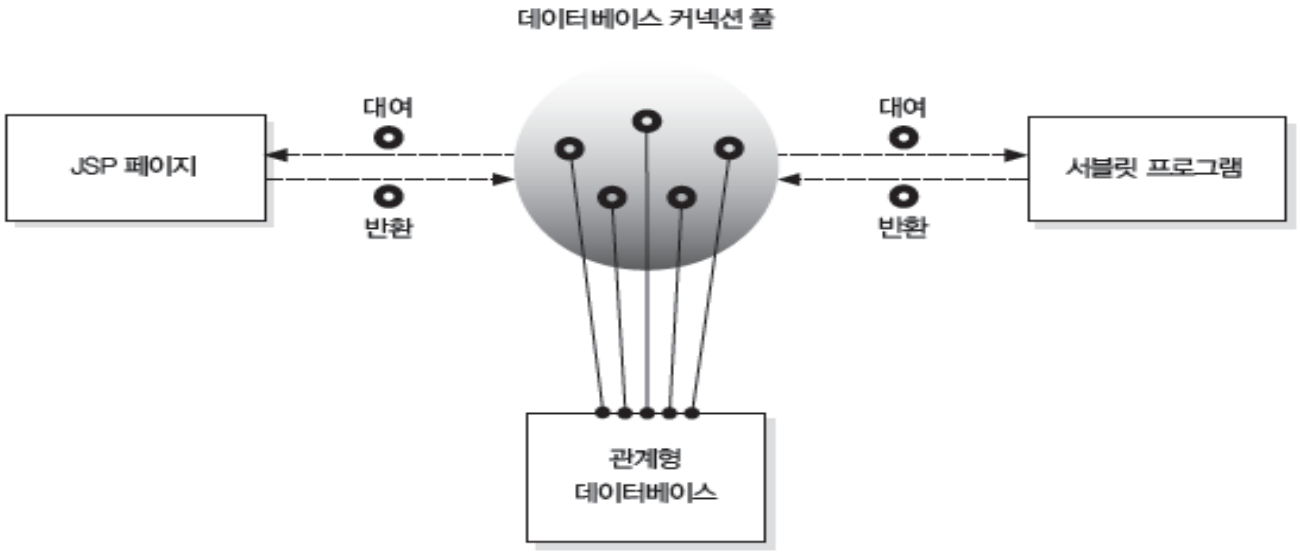
❖ **Connection.setAutoCommit(false) – 기본값은 true**

```
try {
    conn = DriverManager.getConnection(...);
    // 트랜잭션 시작
    conn.setAutoCommit(false);
    ... // 쿼리 실행
    ... // 쿼리 실행
    // 트랜잭션 커밋
    conn.commit();
} catch(SQLException ex) {
    if (conn != null) {
        // 트랜잭션 롤백
        conn.rollback();
    }
} finally {
    if (conn != null) {
        try {
            conn.close();
        } catch(SQLException ex) {}
    }
}
```

여러 개의 쿼리를 하나의 작업단위로 묶어 실행 - 트랜잭션(Transaction)

3. 데이터베이스 커넥션 풀 사용하기

- 데이터베이스에 동시에 접속할 수 있는 사용자 수는 한정되어 있고, 웹 서버에 동시에 수백, 수천의 사용자들이 접속할 수 있는 상황에서 커넥션의 생성과 연결 시간의 절약을 위해 사용, 커넥션의 수를 제한하고 재사용하기 위해 사용됨
- 웹 서버에 수백, 수천의 사용자 요청에 따라 데이터베이스로 새로운 접속을 하는 것은 현실적으로 불가능하므로, 데이터베이스와 몇 개의 접속을 미리 생성하여 데이터베이스 커넥션 풀(Database Connection Pool)에 저장해 놓고 필요할 때 임대해 쓰고 반환



3. JNDI를 이용한 커넥션 풀 사용하기

❖ JNDI(Java Naming and Directory Interface)란?

- JNDI(Java Naming and Directory Interface)란 자바에서 네이밍 서비스를 이용할 수 이도록 제공하는 인터페이스로 논리적인(가상의) 이름을 디렉토리 서비스의 파일 또는 자바 객체, 서버 등과 연결해주는 서비스를 말하며 DNS가 도메인 이름을 IP주소로 변환해 주는 서비스와 비슷한 개념으로 데이터 및 객체를 발견하고 참고하기 위한 자바 API이다.
- 대표적인 디렉토리 서비스에는 LDAP(Light weight Directory Access Protocol), Active Directory 서비스가 있다.
- JNDI는 대규모 애플리케이션 개발의 분산처리 환경에서 여러 가지 형태의 자원을 효율적으로 관리하고 참조할 수 있다는 장점이 있으며 J2EE 플랫폼의 일부이다.
- JNDI 인터페이스는 javax.naming 패키지에 존재하며 모든 리소스는 기본 네임스페이스인 java:con/env에 리소스 이름을 추가하는 형식으로 지정한다.
- JNDI를 이용한 데이터베이스 커넥션 풀은 커넥션 풀에서 사용하는 DataSource 객체를 네이밍 서비스를 이용해 컨테이너로 부터 제공 받는다.
- JNDI를 이용한 커넥션 풀의 설정은 톰캣의 context.xml에 설정하는 방법과 애플리케이션에서 별도의 context.xml 파일을 작성해 설정하는 방법이 있다.

3. JNDI를 이용한 커넥션 풀 사용하기

❖ 톰캣 서버의 context.xml 파일에 커넥션 풀 설정하기

- 톰캣의 context.xml에 DBCP 정보를 설정하고 애플리케이션에서 JNDI를 이용하기 위해 web.xml에 JNDI 객체를 참조하는 리소스 참조를 명시적으로 기술해야 한다.
- 톰캣 서버의 context.xml 파일에 아래와 같이 DBCP 정보를 설정한다.

```
<Context> <- 톰캣의 Context 파일의 루트 태그
   <Resource name="jdbc/membersDBPool"
      auth="Container"
      type="javax.sql.DataSource"
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/members"
      username="root"
      password="12345678“
      factory="org.apache.commons.dbcp.BasicDataSourceFactory"
      maxActive="10"
      maxIdle="5" />
</Context>
```

3. JNDI를 이용한 커넥션 풀 사용하기

- 애플리케이션에서 별도의 context.xml 파일에 커넥션 풀 설정하기
  - 별도의 context.xml 파일에 DBCP 정보를 설정하고 이클립스 프로젝트의 META-INF 폴더에 저장한다. 애플리케이션에서 JNDI를 이용하기 위해 web.xml에 JNDI 객체를 참조하는 리소스 참조를 명시적으로 기술해야 한다.
  - 별도의 context.xml 파일에 아래와 같이 DBCP 정보를 설정한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Resource name="jdbc/membersDBPool"
        auth="Container"
        type="javax.sql.DataSource"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/members"
        username="root"
        password="12345678“
        factory="org.apache.commons.dbcp.BasicDataSourceFactory"
        maxActive="10"
        maxIdle="5" />
</Context>
```

3. JNDI를 이용한 커넥션 풀 사용하기

❖ web.xml 파일에서 리소스 참조 설정하기

- 톰캣 서버의 context.xml 파일이나 별도의 context.xml 파일을 작성해 DBCP관련 정보를 설정을 한 후 web.xml 파일에 JNDI 연결에 필요한 리소스 참조를 <web-app> 루트 태그 아래에 다음과 같이 설정한다.
- <description>과 <res-type>은 생략할 수 있다.

```
<resource-ref>
    <description>DBCP Context Setting</description>
    <res-type>javax.sql.DataSource</res-type>
    <res-ref-name>jdbc/membersDBPool</res-ref-name>
    <res-auth>Container</res-auth>
</resource-ref>
```

3. JNDI를 이용한 커넥션 풀 사용하기

- **소스 코드에서 context.xml 에 설정된 DBCP 정보를 읽어 커넥션 얻기**
  - InitialContext 객체를 생성해 "java:/comp/env" 네임스페이스에 기술된 이름을 찾아 Context 객체를 리턴 받아 Context 객체를 이용해 "jdbc/membersDBPool" 이름을 가진 DBCP에서 DataSource 객체를 얻어 커넥션을 대여 받는다.
  - init.lookup("java:comp/env/jdbc/membersDBPool");와 같이 한 번으로 DataSource 객체를 얻을 수도 있다.
  - "java:/comp/env"는 정해진 기본 네임스페이스고 "jdbc/membersDBPool " 은 DBCP 이름으로 프로그래머가 임의로 지정할 수 있다.

```
Context initContext = new InitialContext();
Context envCtx = (Context)
initContext.lookup("java:/comp/env");
DataSource ds = (DataSource)
envCtx.lookup("jdbc/mallDBPool");
Connection conn = ds.getConnection();
```