프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 12. 데이터베이스 커넥션 풀과 트랜잭션

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- 데이터베이스 연결 성능을 향상시키기 위한 커넥션 풀의 개념을 이해한다.
- 아파치 톰캣 서버에 커넥션 풀을 설정한다.
- 웹 애플리케이션에서 커넥션 풀의 사용 방법을 배운다.
- 데이터베이스의 트랜잭션 개념을 이해한다.
- 웹 애플리케이션에서 JDBC 트랜잭션 처리 기법을 학습한다.

01. 커넥션과 트랜잭션

1. 커넥션이란

- 커넥션이란 애플리케이션과 데이터베이스의 연결로서, 애플리케이션에서 데이터베이스에 접속하고 접속을 종료하는 일련의 과정을 말한다.
- 동시 접속 사용자가 늘어나면 프로그램에서 데이터베이스를 연결하고 종료하는 일련의 과정은 시스템에 많은 부하를 주게 된다.
- 또한 많은 데이터베이스 공급 업체가 데이터베이스의 동시 접속 사용자 수에 따른 라이선스 정책을 사용하고 있으므로 비효율적인 커넥션 관리는 비용적 손실도 초래하게 된다.
- 따라서 대규모 시스템일수록 커넥션 관리는 애플리케이션 개발에서 중요한 위치를 차지한다.

> 예를 들어, 게시판 목록을 보여주는 페이지가 있다고 하자. 게시판 목록을 보여주는 JSP에서 데이터베이스에 접속하고, SQL문을 보내서 결과를 수신한 다음 연결을 종료한다. 이때 연결을 종료하지 않으면 데이터베이스에 불필요한 연결이 유지되어 데이터베이스 서버 자원을 낭비하게 된다. (연결을 종료하지 않으면 데이터베이스의 최대 커넥션을 초과해 애플리케이션에서 데이터베이스에 접속하지 못하는 경우가 종종 생기기도 한다).

01. 커넥션과 트랜잭션

2. 트랜잭션이란

- 트랜잭션은 여러 작업을 하나의 단위로 처리하는 것을 말하는데, 대부분의 데이터베이스가 이러한 트랜잭션 처리를 위한 메커니즘을 제공하고 있다.
- 트랜잭션이 필요한 이유는 복잡한 데이터베이스 관련 작업들이 서로 연관되어 있을때 하나의 작업이 최종적으로 완료되기 위해 다른 작업들이 모두 정상적으로 완료되어야 하거나 혹은 잘못될 경우 앞에서 처리한 작업을 원래대로 되돌릴 필요가 있기 때문이다.
- 이러한 트랜잭션 작업을 프로그래머가 코딩으로 처리할 경우 로직의 실수에 따라 큰 문제가 발생할 수도 있어 데이터베이스 자체의 트랜잭션 기능과 JavaEE에서의 선언적 트랜잭션 기법 등이 요구된다.

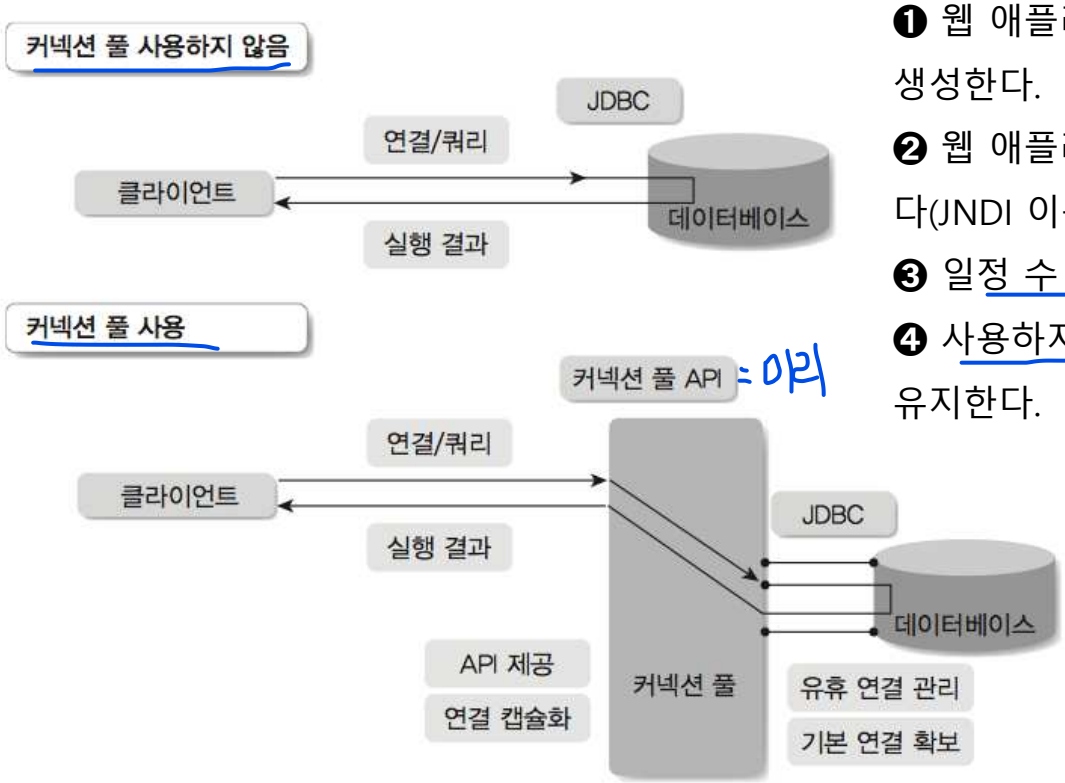
> 예를 들어, 사용자가 쇼핑몰에서 물건을 구입한다고 했을 때 상품 주문 테이블에서는 주문 정보가 들어가야 하고, 회원 테이블에 포인트가 추가되어야 한다. 또한 택배 회사에서 사용할 수 있도록 배송 테이블에 주문 내역과 주소 등도 입력되어야 한다. 이때 일련의 작업은 모두 성공되어야 하며 중간에 하나라도 잘못된다면 데이터가 맞지 않는 문제가 발생한다.

02. 데이터베이스 커넥션 관리

1. 데이터베이스 커넥션 풀이란

- 데이터베이스 커넥션 풀(Database Connection Pool)이란 애플리케이션에서 필요로 하는 시점에 커넥션을 만드는 것이 아니라 미리 일정한 수의 커넥션을 만들어놓고 필요한 시점에 애플리케이션에 제공하는 서비스 및 관리 체계를 말한다.

커넥션 풀의 구조와 동작



❶ 웹 애플리케이션 서버가 시작될 때 일정 수의 커넥션을 미리 생성한다.

❷ 웹 애플리케이션 요청에 따라 생성된 커넥션 객체를 전달한다(JNDI 이용).

❸ 일정 수 이상의 커넥션이 사용되면 새로운 커넥션을 만든다.

❹ 사용하지 않는 커넥션은 종료하고 최소한의 기본 커넥션을 유지한다.

[그림 12-1] 커넥션 풀 동작 구조

02. 데이터베이스 커넥션 관리

커넥션 풀 구현 유형

[표 12-1] 데이터베이스 커넥션 풀 구현 유형

유형	설명
직접 구현	개발자가 직접 javax.sql.DataSource 인터페이스를 구현하거나, 직접 새로운 형태의 커넥션 풀을 구현하는 방법이다. 학습을 위한 목적이 아니라면 일반적으로는 권장되지 않는다.
아파치 자카르타 DBCP API를 이용한 구현 (commons-dbcp)	아파치 그룹의 공개된 데이터베이스 커넥션 풀 API인 DBCP를 이용하는 방법이다. 프로그래밍에 자신이 있고 서블릿, 리스너 등의 활용에도 능숙하다면 이를 이용해 자신만의 커넥션 풀을 구성할 수 있다.
애플리케이션 서버 제공	애플리케이션에서 제공되는 커넥션 풀을 사용하는 방법이다. 최근의 웹 애플리케이션 서버들은 javax.sql.DataSource 인터페이스를 따르는 커넥션 풀을 제공하므로 호환에는 큰 문제가 없다. 이 경우 JNDI 네이밍 서비스(Naming Service)를 통해 커넥션 풀을 사용할 수 있다.
프레임워크 제공	애플리케이션 서버와는 별도로 스프링이나 스트러츠 같은 애플리케이션 프레임워크에서 제공하는 커넥션 풀을 사용하는 방법이다. 웹 애플리케이션 이외의 개발에도 사용할 수 있다는 점을 제외하고는 기본적으로 애플리케이션 서버가 제공하는 방식과 다르지 않으므로 개발 성격에 따라 프레임워크에서 제공되는 커넥션 풀을 사용하는 것도 괜찮은 방법이다. 자세한 내용은 해당 프레임워크 개발문서를 참고하도록 한다.

02. 데이터베이스 커넥션 관리

2. DBCP API를 통한 커넥션 풀 구현

DBCP API 설치하기

- 톰캣 7.0에서 부터는 자체적으로 DBCP 를 내하고 있어 아파치 DBCP(commons-dbcp) 를 따로 설치할 필요가 없다.

톰캣에 DataSource 설정하기

- 톰캣 서버 설정 파일인 server.xml 혹은 context.xml 에서 설정한다.(context.xml 권장)
- DB 연결을 위한 정보와 커넥션풀 운영 관련 정보로 구성된다.
- **context.xml 파일 수정** – 교재 p.514 참고

```
01 <Resource name="jdbc/mysql" auth="Container"
02     type="javax.sql.DataSource"
03     driverClassName="com.mysql.jdbc.Driver"
04     url="jdbc:mysql://127.0.0.1/jspdb"
05     username="jspbook" password="1234"
06     maxActive="5" maxIdle="3" maxWait="-1" />
```

02. 데이터베이스 커넥션 관리

[표 12-3] <Resource> 태그 속성

매개변수	설명
username	데이터베이스 접속을 위한 계정 이름이다.
password	데이터베이스 접속을 위한 계정 비밀번호다.
driverClassName	데이터베이스 JDBC 드라이버 클래스 이름이다(예를 들어, 오라클에서는 oracle.jdbc.OracleDriver로 사용한다).
url	데이터베이스 접속에 필요한 정보다(예를 들어, 오라클에서는 jdbc:oracle:thin:@localhost:1521로 사용한다).
maxActive	데이터베이스 최대 연결 개수다. 즉 클라이언트 요청에 따라 커넥션을 생성할 최대 개수다. maxActive 이상의 연결이 동시에 발생할 경우 우선순위에 따라 연결을 처리해야 하므로, 일부 클라이언트 연결에서는 지연이 발생할 수 있다. 0인 경우 시스템 성능 및 데이터베이스 서버 설정에 따라 무제한으로 연결할 수 있다.
maxIdle	데이터베이스 최대 유휴 연결 개수다. 즉, 클라이언트 사용이 없을 때에도 항상 유지할 연결의 최대 개수를 의미한다. -1인 경우 무제한으로 연결된다.

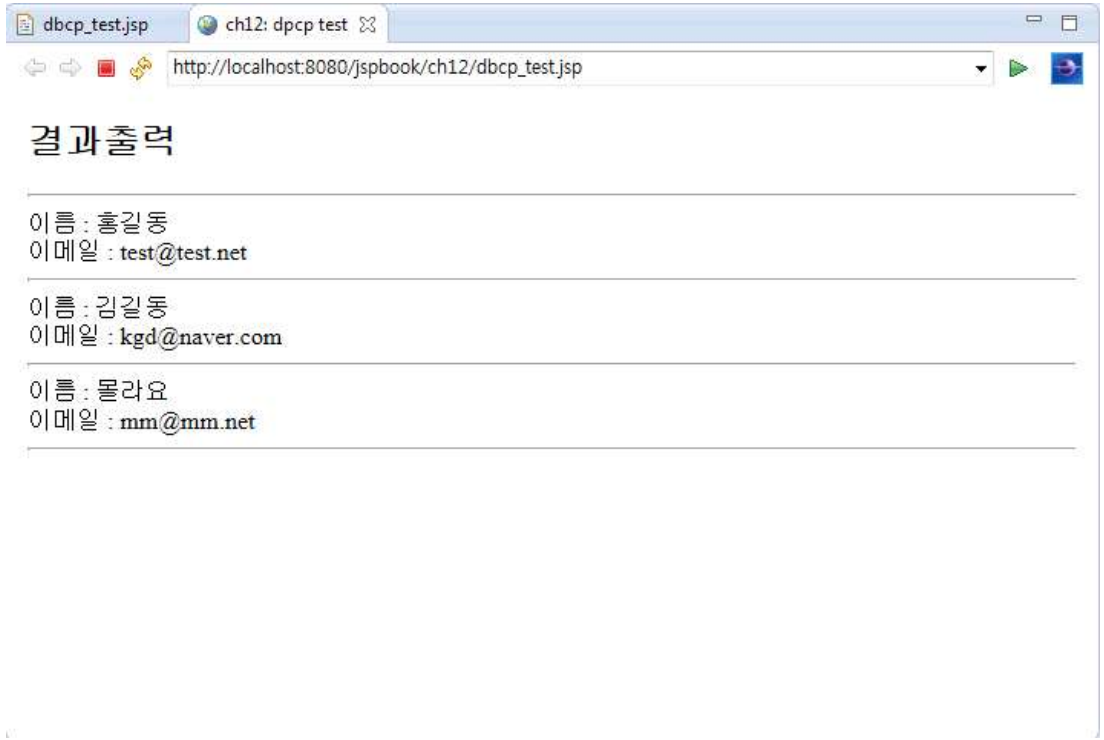
- **web.xml 설정** – 교재 p.515 참고
  - 자바 프로그램에서 네이밍서비스로 커넥션 풀에 접근할 수 있도록 JNDI 이름을 설정함.

```
01 <resource-ref>
02     <res-ref-name>jdbc/mysql</res-ref-name>
03     <res-auth>Container</res-auth>
04
05 </resource-ref>
```

02. 데이터베이스 커넥션 관리

DBCP API 설정 테스트하기

- JNDI 이름을 통해 Connection 객체를 가지고 오는 부분만 차이가 있고 나머지는 일반적인 JDBC 프로그램과 동일함.
- 여기서는 JSP 에서 JSTL sql 라이브러리를 이용해 간단하게 테스트 함.
- **[실습] dbcp_test.jsp** – 교재 p.516 참고



[그림 12-2] DBCP 테스트

03. [응용실습]커넥션 풀 : DBCP 설정 및 클라이언트 구현

실습 개요

- 8장의 jdbctest.jsp 를 DBCP 버전으로 수정하는 과정을 통해 DBCP 의 구현상의 차이점과 JNDI 사용법을 배운다.
- [실습] jdbctest_dbcp.jsp – 교재 p.518 ~ 520 참고
  - 기존 8장의 jdbctest.jsp 소스를 ch12로 복사해 이름을 jdbctest_dbcp.jsp로 변경한다.

[그림 12-3] jdbctext_dbcp.jsp 실행 결과

03. [응용실습]커넥션 풀 : DBCP 설정 및 클라이언트 구현

주요 소스코드 분석

- 커넥션을 가져오는 부분을 제외하고는 기존 소스와 동일함.
- new InitialContext()로 Context 객체를 확보하고 JNDI 이름으로 DataSource 객체를 참조한다.
- JNDI 접근은 java:/comp/env 로 Context 객체에 접근한 다음 web.xml에 등록한 JDNI 이름인 jdbc/mysql로 DataSource 객체를 얻는다.
- Connection 객체는 DataSource의 getConnection() 메서드로 구하게 된다.

```
09    try{
10        Context initContext = new InitialContext();
11        Context envContext =
12        (Contet)initContext.lookup("java:/ comp/env");
13        DataSource ds = (DataSource)envContext.lookup("jdbc/mysql");
14        // 커넥션 얻기
15        Conn = ds.getConnection();
```

04. 트랜잭션 처리

1. 트랜잭션이란

- 데이터베이스 연동 프로그램에서 커넥션과 함께 중요한 부분은 트랜잭션(Transaction)이다.
- 트랜잭션은 데이터베이스에서 일련의 작업을 하나로 묶어 처리하는 것을 의미하는데, 사실상 커넥션보다 더 중요하다.
- 다음 사례를 통해서 트랜잭션의 의미를 생각해보자.

> 한 고객이 통장A에서 통장B로 계좌를 이체하려고 한다. 실제 이러한 과정은 통장A에서 돈을 인출한 후, 통장B로 입금하는 과정을 거친다. 그런데 만약 통장A에서는 정상적으로 돈이 인출되었지만 통장B로 입금을 처리하는 부분에 문제가 생길 경우, 통장A에서 인출된 돈은 어떻게 처리해야 할까?

저자 한마디

데이터 무결성

데이터 무결성은 데이터베이스에 요구되는 중요 특성 중 하나로, 데이터베이스에 저장된 데이터는 실제 존재해야 하며 모순이 없어야 함을 의미한다. 일반적으로 데이터베이스에서 무결성을 보장하기 위한 방법으로 제약(constraint) 조건을 사용하는데, 여기에는 개체 무결성, 참조 무결성, 키 제약, 도메인 제약 같은 것이 있다.

트랜잭션과 Java EE

트랜잭션은 너무나도 중요하기 때문에 사실 개발자가 트랜잭션을 관리하는 것은 상당한 부담이 된다. 이는 개발자에게, 또 개발을 의뢰한 고객에게도 좋은 일이 아닐 것이다. 따라서 이러한 트랜잭션은 개발자가 프로그램 내에서 임의로 처리하는 것이 아니라, 애플리케이션 서버 차원에서 더욱 안정적이고 체계적인 관리가 이루어질 수 있어야 한다. Java EE 스펙에는 이러한 트랜잭션에 대한 요구사항이 들어 있으며, Java EE 스펙을 구현한 애플리케이션 서버에서는 서버 차원에서의 선언적 트랜잭션 관리 등을 수행할 수 있다. 물론, 이런 경우 JSP만으로는 불가능하며, EJB나 그에 준하는 퍼시스턴스 프레임워크(Persistence Framework) 등을 사용해야 한다.

04. 트랜잭션 처리

2. commit과 rollback

- commit과 rollback은 데이터베이스에서 트랜잭션 지원을 위해 제공하는 기능이다.

> • commit : 데이터베이스에 트랜잭션이 완료되었음을 알리는 명령이다. Commit이 완료되기까지 결과는 현재 커넥션에서만 유효하고, 다른 커넥션에서는 처리 내용을 확인할 수 없다. 즉 commit이 완료되어야만 다른 커넥션에서도 변경된 데이터를 확인할 수 있다.
> • rollback : 트랜잭션을 취소하는 명령으로, 현재 연결에서 수행한 결과를 원래대로 되돌리는 역할을 한다. 트랜잭션이 잘못되었을 경우 rollback을 이용해 모든 작업을 원래 상태로 되돌릴 수 있다.

- **[실습] commit, rollback 사용** - 교재 p.525 ~ 527 참고
  - 앞에서 만들었던 jdbctest_dbcp.jsp 와 MySQL Workbench를 통해 서로 다른 두 프로그램에서 jdbc_test 테이블에 데이터를 등록 혹은 삭제 하면서 commit, rollback 명령의 동작, 그리고 트랜잭션의 개념을 이해한다.

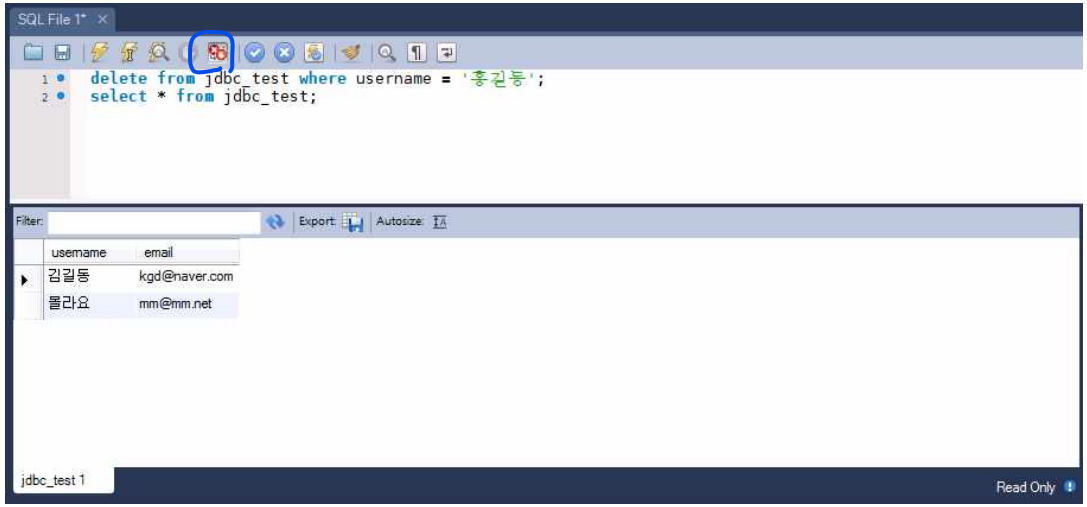
❶ 데이터 삭제

```
delete from jdbc_test where username='홍길동';
```

04. 트랜잭션 처리

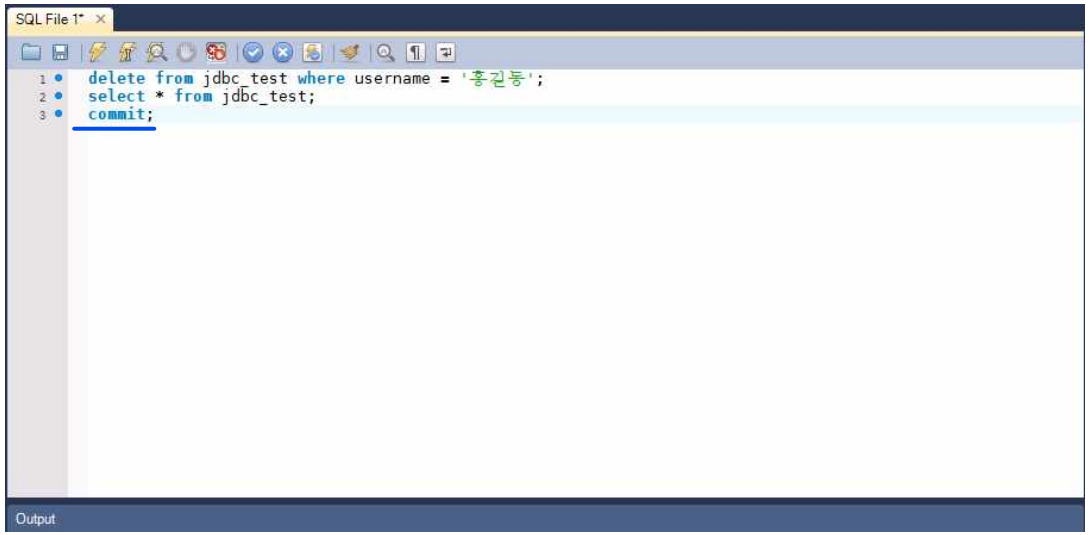
❷ 데이터 삭제 후 확인을 위한 쿼리 실행

```
select * from jdbc_test;
```



[그림 12-6] 게시물 삭제

❸ SQL 명령 입력 화면에서 commit 입력



[그림 12-7] commit 실행 결과

04. 트랜잭션 처리

❹ 실행결과 확인

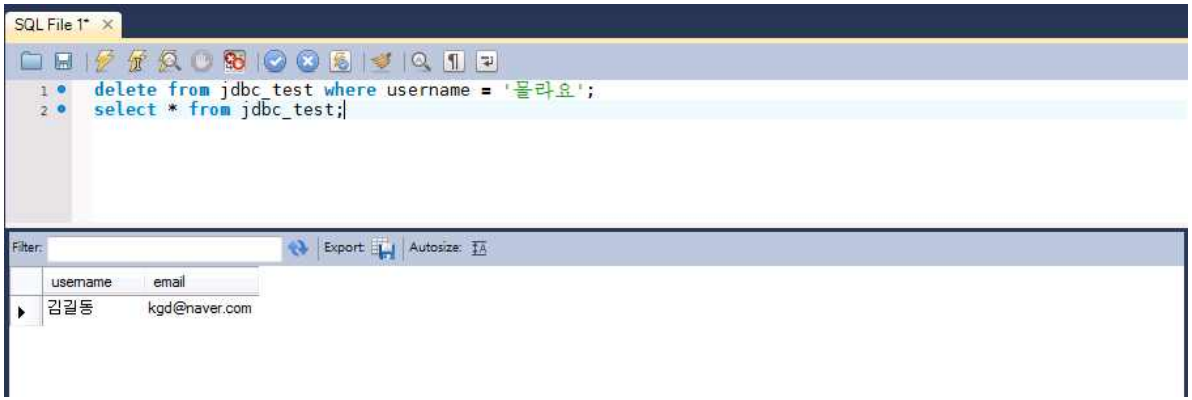
[그림 12-8] 삭제 확인

04. 트랜잭션 처리

❶ '몰라요'를 삭제한 후 select문을 통해 데이터를 조회한 후, rollback을 통해 삭제된 데이터를 복구

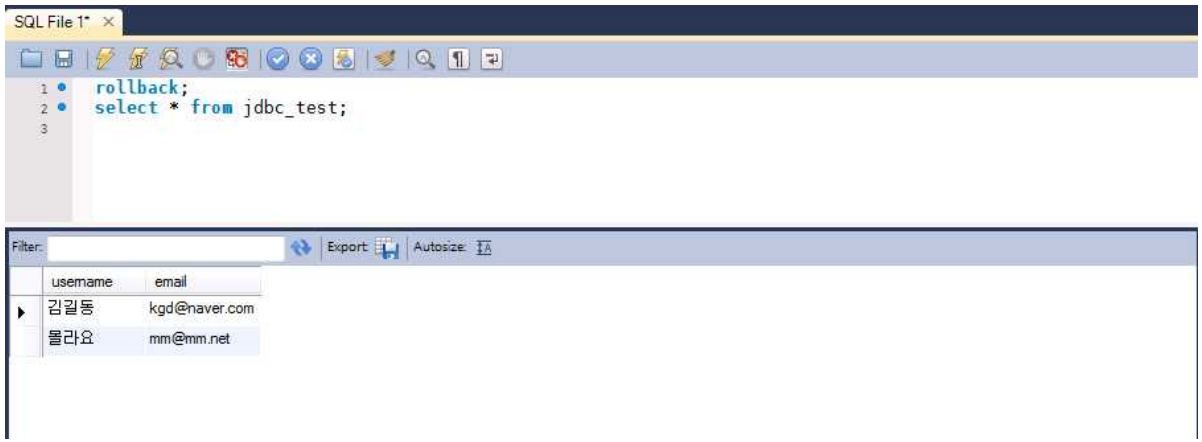
```
delete from jdbc_test where username='몰라요';
```

❷ 데이터 삭제 후 select를 수행하여 데이터가 삭제된 것을 확인



[그림 12-9] 삭제 데이터 확인

❸ rollback 명령 수행 후, select문을 수행해 삭제된 데이터가 복구된 것을 확인한다.



[그림 12-10] 삭제 데이터 rollback 확인

04. 트랜잭션 처리

3. JDBC에서 트랜잭션 처리

- JDBC에서는 프로그램적인 방법을 통해 트랜잭션 처리를 지원한다.
- 기본적으로는 auto commit 모드를 해제한 다음 프로그램에서 commit 과 rollback 시기를 정해 처리하도록 한다.

auto commit 해제 방법

- auto commit 모드를 해제하면 프로그램 내에서 update문이나 delete문을 수행해도 결과를 반영하지 않으므로 트랜잭션 관리에 주의를 기울여야 한다. 따라서 필요한 부분에서 auto commit을 해제하고, 사용이 끝난 다음에는 다시 auto commit 모드로 전환해주는 것이 좋다.

```
conn.setAutoCommit(false);
```

- 반대로 설정하기 위해서는 인자를 true로 주면 된다.

04. 트랜잭션 처리

JSP에서 commit과 rollback

- JSP에서 commit과 rollback은 Connection 클래스의 메서드로 제공된다.

```
conn.commit();
conn.rollback();
```

일괄갱신

- Statement에서 일괄 갱신

```
conn.setAutoCommit(false);
Statement stmt = conn.createStatement();
stmt.addBatch("insert into table1 values(1,'test')");
stmt.addBatch("update table2 set memo='test' ");
int[] cnt = stmt.executeBatch();
```

- PreparedStatement에서 일괄 갱신

```
PreparedStatement pstmt =
conn.prepareStatement("update table1 set memo=?
where num=?");
pstmt.setString(1,"테스트1");
pstmt.setString(2,"1");
pstmt.addBatch();
pstmt.setString(1,"테스트2");
pstmt.setString(2,"2");
pstmt.addBatch();
int[] cnt = pstmt.executeBatch();
```

05. [응용실습]트랜잭션 관리 : 계좌이체 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

실습 개요

- 두 테이블을 이용해 계좌 이체를 하는 예제 구현을 통해 트랜잭션 구현 방법을 익힌다.
- 잔액을 보여주고 임의 금액을 Bank1 에서 Bank2 로 이체 한다. 이때 Bank2 가 80 이상이 될 경우 "수용 한도가 초가 되었다"는 메시지를 보여주고 이체된 내역을 원래대로 되돌린다.

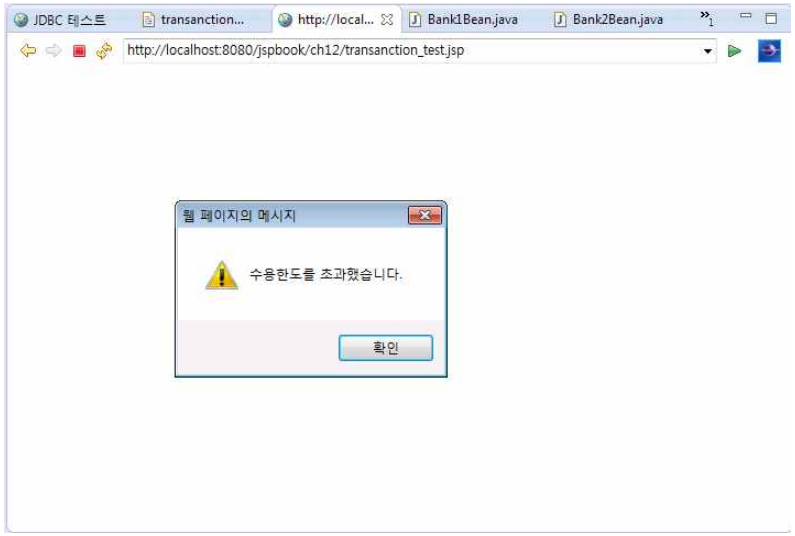
이체 화면 생성

- **이체 화면 생성(transaction_test.jsp)** – 교재 p.531 ~ 532 참고

빈즈 클래스 작성

- 빈즈 클래스 ❶ 작성(Bank1Bean.java) – 교재 p.533 ~ 536 참고
- 빈즈 클래스 ❷ 작성(Bank2Bean.java) – 교재 p.537 ~ 539 참고

[그림 12-11] 이체 화면



[그림 12-12] 한도를 초과할 때 나타나는 메시지

05. [응용실습]트랜잭션 관리 : 계좌이체 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

주요 소스코드 분석(transaction_test.jsp)

- 계좌 현황을 보여주고 이체를 수행하는 jsp
- 계좌 DB 처리를 위한 빈즈 클래스 선언.
- Bank1Bean을 이용해 이체를 처리하고 false 리턴시 alert 메시지 출력.

```
04 <jsp:useBean scope="page" id="bb" class="jspbook.ch12.Bank1Bean" />
05 <jsp:useBean scope="page" id="bb2" class="jspbook.ch12.Bank2Bean" />
06
07 <%
08     if(request.getMethod().equals("POST")){
09         if(bb.transfer(Integer.parseInt(request.getParameter("bal"))))
10             out.println("<script>alert('정상처리 되었습니다.')</script>");
11         else
12             out.println("<script>alert('수용한도를 초과했습니다.')</script>");
13     }
14     bb.getData();
15     bb2.getData();
16 %>
```

05. [응용실습]트랜잭션 관리 : 계좌이체 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

주요 소스코드 분석(Bank1Bean.java)

- Bank1Bean과 Bank2Bean 은 기본적으로 동일한 구조임.
- Bank1Bean의 transfer() 메서드만 살펴봄.
- Auto Commit 해제

```
073     conn.setAutoCommit(false);
```

- 계좌 이체를 위한 sql 문(bank1 에서 차감)

```
075     .prepareStatement("update bank1 set balance = balance-? where aid=101");
```

- 계좌 이체를 위한 sql 문(bank2 에 더하기)

```
079     .prepareStatement("update bakd2 set balance = balance+? where aid=201");
```

05. [응용실습]트랜잭션 관리 : 계좌이체 구현

주요 소스코드 분석(Bank1Bean.java)

- 조건에 따른 commit / rollback

```
090     if (rs.getInt(1) > 40) {
091         conn.rollback();
092         return false;
093     }
094     else
095         conn.commit();
096     }
```

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음