웹 어플리케이션에서의 효율적인 데이터 베이스 커넥션 관리를 위한 자바 컴포넌트의 구현

김종현() 황대준
성균관대학교 공과대학 전기전자컴퓨터공학부
hyounfkaywon.ac.kr, djhwang@yurim.skku.ac.kr

The Implementation of Java Components for Effective Database Connectivity in Web Applications

Jong-Hyoun Kim Dae-Joon Hwang
The School of Electrical and Computer Science, Sung Kyun Kwan University

요 약

현재 JDBC를 통한 웹 어플리케이션의 제작은 데이터베이스 커넥션에 대한 관리 기능의 부재, 다중 데이터베이스에 대한 효율적인 접근 방법의 부재, 예외 처리(try~ catch) 구문의 부담, 질의 결과(ResultSet)에 대한 subset을 처리 방법의 어려움이 존재하는 JDBC의 한계로 인해 개발에의 상당한 까다로움과 고려해야 할 사항이 다수 존재한다. 본 논문에서는 이러한 어려움을 해결함과 동시에 개발 비용과 개발 시간을 단축하고 사용자 요구에의 응답 시간을 최소화할 수 있는 데이터베이스 connectivity를 위한 컴포넌트를 설계하고 그것을 구현한다.
본 컴포넌트는 다중 데이터베이스를 지원하고, 데이터베이스에 대한 커넥션을 관리하며, JDBC 드라이버의 등록과 로딩을 관장하며, 예외 상황을 내부적으로 처리하고 ResultSet을 효율적으로 다룰 수 있는 기능 및 다중 생성자 권한으로 관련 테이블에의 접근 용이성을 제공한다.
본 컴포넌트를 이용한 웹 어플리케이션을 제작에 필요한 코드의 양은 그렇지 않았을 때에 비해서 약 30%의 절감 효과가 있다. 더불어, 디버깅에 필요한 시간을 최소화 할 수 있어 개발 기간을 약 2/3 수준으로 단축할 수 있으며 시스템 성능 향상에 도움을 준다. 이것은 실제 프로젝트(한국통신 웹사이트-www.kt.co.kr-, 존슨앤존슨 웹사이트-www.cleanandclear.co.kr-)를 통해 검증되었다.

1. 서론

데이터베이스에의 연결과 그것의 사용을 위해 마이크로소프트사의 ODBC가 단일화된 인터페이스를 제공하는 것과 같이 자바 기술은 JDBC를 통해 단일화된 인터페이스를 제공한다. JDBC를 통해 데이터베이스에 접근하여 데이터를 처리하기 위해서는 기본적으로 다음의 6 단계를 필요로 한다[1,2].

첫째, JDBC 드라이버 로딩 하기
둘째, 데이터베이스와 커넥션 맺기
셋째, SQL 문장 생성 하기
넷째, SQL 문장 실행 하기
다섯째, 실행 결과 처리 하기
여섯째, 사용한 자원 반납하기

이와 같은 기본 단계에 따른 개발은 데이터베이스 관리나 접근 측면에서 비효율적임과 동시에 개발 속도를 저하시킨다. 2장에서는 이와 관련된 연구를 소개하고, 웹 어플리케이션을 개발할 때의 JDBC의 한계와 그 해결책을 알아보고 그것의 구현 내용을 3장에서 제시한다. 4장에서는 해결책으로부터 얻게 되는 이득과 향후 연구에 대해서 기술한다.

2. 관련연구 및 제품

[3]은 데이터베이스 커넥션 풀을 위한 랩퍼 클래스를 제시함으로써 성능 향상을 시도한다. 이 제안은 사용자 요구에의 응답 시간을 단축시키는 데 많은 기여를 했지만 다중 데이터베이스 사용 및 단일 데이터베이스 관리 시스템에서의 다중 데이터베이스 생성자에 의한 테이블 접근 방법에 대한 문제 해결법은 제시되어 있지 않다.
자바 기술을 이용한 웹 사이트 구축에의 문제점에 대한 많은 해결점은 Bea 사의 웹로직 제품에서 찾을 수 있다. 데이터베이스 커넥션 관리 문제 역시 해결법 역시 구현 되어 있는 데, 이것의 문제는 무엇보다도 웹로직 제품에 종속 되어 있다는 것이며 또한 다중 데이터베이스 생성자에 의한 테이블 접근 방법이 웹 페이지 단위로 처리해야 하는 단점이 있다는 것이다[4].

3. JDBC의 한계와 문제점 그리고 그것의 해결책

JDBC는 데이터베이스에의 접근과 그것의 이용을 위한 단일화된 인터페이스로 개발자에게 편리함을 제공하는 반면 그것의 문제점 역시 존재한다. JDBC를 통한 웹 어플리케이션의 개발에 따른 문제점 및 필수 해결 사항은 다음의 5가지로 요약할 수 있다.무엇보다도 첫째, 데이터베이스 커넥션에 대한 관리 기능이 제공되지 않기 때문에 개발된 어플리케이션이 실제 적용될 때에 시스템 성능에 지대한 영향을 미칠 수 있다. 물론, 이 어플리케이션이 웹 어플리케이션일 경우는 그 영향이 더욱 크다.
둘째, 해당 JDBC 드라이버를 어플리케이션 당 한 번만 로딩할 수 있는 방안이 모색되어야 하며 동시에 사용할 드라이버의 변경이 손쉽게 이루어질 수 있어야 하며 그것 또한 내부적으로 관리되어야 한다.
셋째, 웹 어플리케이션은 다수의 DBMS를 사용하거나 단일 DBMS하에서 다수의 데이터베이스에 접근하는 경우가 빈번하게 발생하므로 다중 데이터베이스에 대한 효율적인 접근 방법이 제공되어야 한다.
넷째, 예외 처리(try~ catch)는 내부적으로 해결되어야 한다. 모든 웹 페이지에게 예외 처리를 맡기는 것은 매우 버거운 작업일 뿐더러 어플리케이션 개발 속도를 저하 시킨다. 예외 상황의 처리를 내부적으로 실행함으로써 웹 페이지를 보다 간결하고 보다 신속하게 작성할 수 있도록 해야 하며 발생할 수 있는 오류를 보다 단순화할 수 있게 해야 한다.
다섯째, JDBC가 제공하는 ResultSet은 결과에 대한 subset을 처리하는 데 매우 어렵게 만들어져 있다.
여섯째, 단일 데이터베이스 관리 시스템에서의 다중 데이터베이스 생성자에 의한 테이블 접근 방법이 간편하게 제공되어야 한다.
본 논문에서는 이와 같은 문제점을 해결하고 필수 요구사항을 충족시키는 그림 1과 같은 데이터베이스 커넥션과 액세스에 대한 컴포넌트 풀(component pool)을 제안한다.

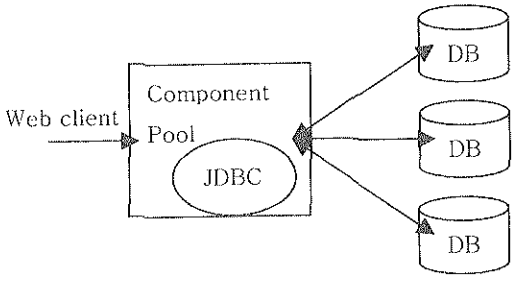

그림 1. Component Pool

컴포넌트 풀은 다중 데이터베이스를 지원하고, 데이터베이스에 대한 커넥션을 관리하며, JDBC 드라이버의 등록과 로딩을 관장하며, 예외 상황을 내부적으로 처리하고 ResultSet을 효율적으로 다룰 수 있는 기능 및 다중 생성자 권한으로 관련 테이블에의 접근 용이성을 제공한다. 컴포넌트 풀 내의 클래스 연관 관계가 그림 2에 있다.

DBConnectionManager : 현재 사용 중인 데이터베이스들에 대한 정보를 보관하며 데이터베이스 각각에 대한 JDBC 드라이버를 등록하고 로딩한다.
DBConnectionPool : 임의의 데이터베이스에 대한 커넥션 풀을 관리하는 클래스로써 freeConnectionList와 usingConnectionList를 갖고 있다.
ConnectionManager : DBAccess 클래스와 DBConnectionManager 클래스의 중개자로서 해당 JDBC 드라이버의 등록과 로딩을 요청하고 데이터베이스에 대한 커넥션의 획득과 반납을 관장한다.
OracleConnectionManager : 오라클용 JDBC 드라이버와 해당 클래스 그리고 데이터베이스에 대한 접근 방법과 권한이 기술된다.

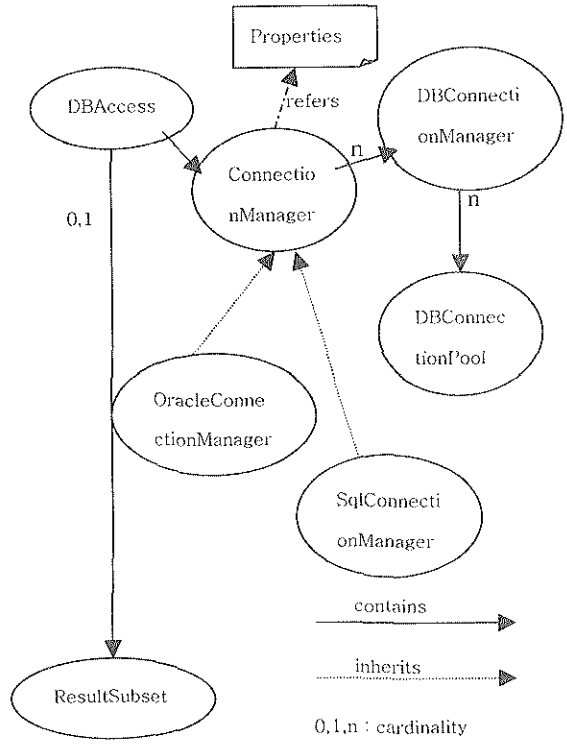

그림 3. Component Pool 내의 클래스 연관관계

SqlConnectionManager : Sql 서버용 JDBC 드라이버와 해당 클래스 그리고 데이터베이스에 대한 접근 방법과 권한이 기술된다.
DBAccess : JDBC API에 대한 예외 상황 처리를 관장한다.
ResultSubset : ResultSet를 페이지 단위로 관리한다.
Properties : 다중 데이터베이스, JDBC 드라이버 URL과 다중 생성자를 기술하고 최대 커넥션 수를 정의한다

4. 구현

본 컴포넌트는 kaywon.db.dbpool 과 kaywon.db.dbaccess 의 패키지로 구현되어 유용한 유틸리티 클래스와 함께 배포되고 있으며 그림 3는 본 컴포넌트에 대한 JSP 페이지에서의 사용 예를 보여 준다.

```
<%@ page import=" kaywon.db.dbpool.*,
kaywon.db.dbaccess.*" %>
<jsp:useBean id=" DBMgr" scope=application
class=" OracleConnectionManager" />
<%
DBAccess access = new DBAccess();
access.setConnection(DBMgr);
String sql = " ....." ;
access.execute(sql);
ResultSubset rs = access.getResultSubset(20);
while (rs.next()) { // doing something }
access.close();
%>
```

그림 3. 컴포넌트 클래스 사용 예

5. 향후 연구과제

앞에서 제시한 컴포넌트를 이용한 웹 어플리케이션을 제작에 필요한 코드의 양은 그렇지 않았을 때에 비해서 약 30%의 절감 효과가 있다. 더불어, 디버깅에 필요한 시간을 최소화 할 수 있어 개발 기간을 약 2/3 수준으로 단축할 수 있다. 이것은 실제 프로젝트(한국통신 웹사이트-www.kt.co.kr-, 존슨앤존슨 웹사이트-www.cleanandclear.co.kr-)의 결과로 도출된 자료이다. 자바 기반의 웹 어플리케이션 제작의 질과 안정성 그리고 확장성의 열쇠는 관련 부분에 대한 컴포넌트화의 정도에 달려 있다라고 말할 수 있다. 이를 기반으로 향후 웹용 컴포넌트 개발과 연구에 많은 시간을 투자할 계획이다.

6. 참고문헌 및 사이트

1.http://java.sun.com/j2se/1.3/docs/guide/jdbc/spec/jdbc-spec.frame.html
2.Database Programming with JDBC and JAVA, O'REILLY, George Reese
3.http://www.webdevelopersjournal.com/columns/connection_pool.html
4.http://www.bea.com/products/weblogic/server/datasheet.shtml