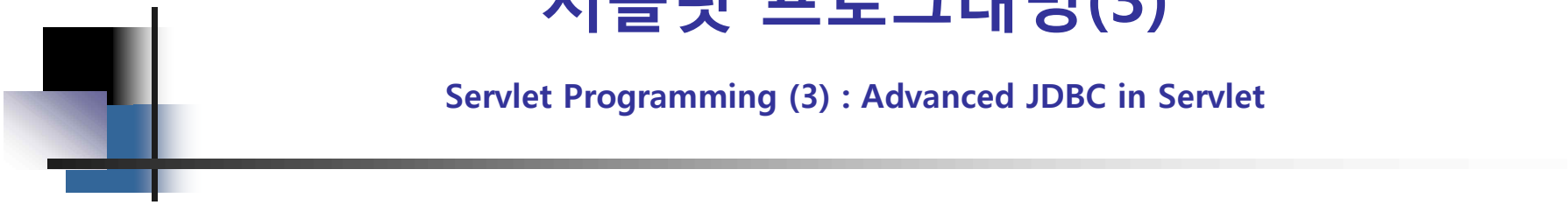


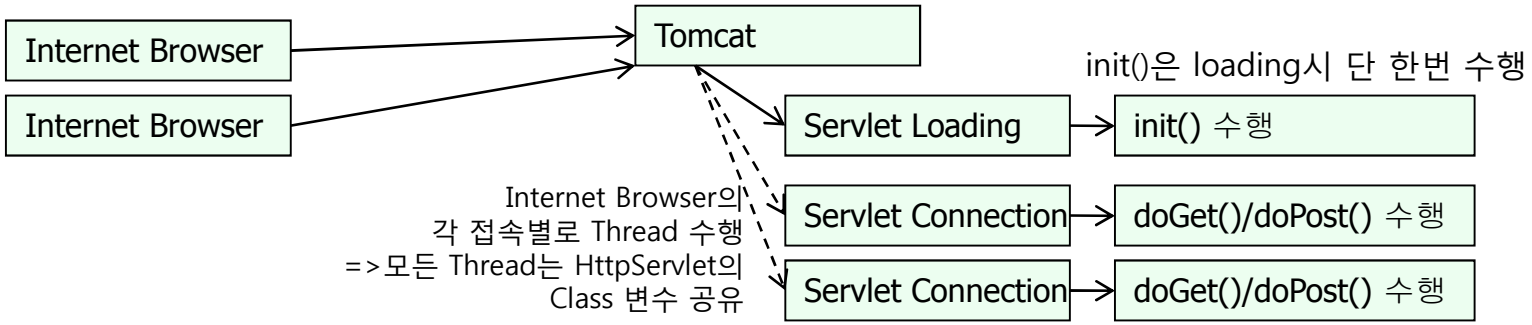
서블릿 프로그래밍(3)

Servlet Programming (3) : Advanced JDBC in Servlet

남 광 우

Reusing Database Objects

- Database Connection의 재사용
  - Servlet의 Thread 모델



  - doGet()에서 Connection을 매번 연결
    - Connection에 매 Internet Browser 접속당 연결해야하므로 상당한 시간이 소요됨
  - init()에서 Connection 연결하고 Thread간 공유
    - 단 한번만 Connection을 연결하므로 처리시간 단축
    - Connection Error 발생시 처리 방법을 고려해야 함
      - Connection Pool 기법 사용

Reusing Database Objects

- 예 1: init()을 이용한 Connection 공유

DBPhoneLookupReuse.java

```
public class DBPhoneLookupReuse extends HttpServlet {

  private Connection con = null;

  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
      // Load (and therefore register) the Sybase driver
      Class.forName(" com.mysql.jdbc.Driver ");

      // Get a Connection to the database
      con = DriverManager.getConnection("jdbc:mysql://localhost/test ?"user", "passwd");
    }
    catch (ClassNotFoundException e) {
      throw new UnavailableException(this, "Couldn't load database driver");
    }
    catch (SQLException e) {
      throw new UnavailableException(this, "Couldn't get db connection");
    }
  }
```

Reusing Database Objects

- 예 1: init()을 이용한 Connection 공유

DBPhoneLookupReuse.java

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
   res.setContentType("text/html");
   PrintWriter out = res.getWriter();

   out.println("<HTML><HEAD><TITLE>Phonebook</TITLE></HEAD>");
   out.println("<BODY>");

   HtmlSQLResult result =
     new HtmlSQLResult("SELECT NAME, PHONE FROM EMPLOYEES", con);

   // Display the resulting output
   out.println("<H2>Employees:</H2>");
   out.println(result);
   out.println("</BODY></HTML>");
 }

 public void destroy() {
   // Clean up.
   try {
     if (con != null) con.close();
   }
   catch (SQLException ignored) { }
 }
}
```

Reusing Database Objects

- 예 2 : Transaction의 사용

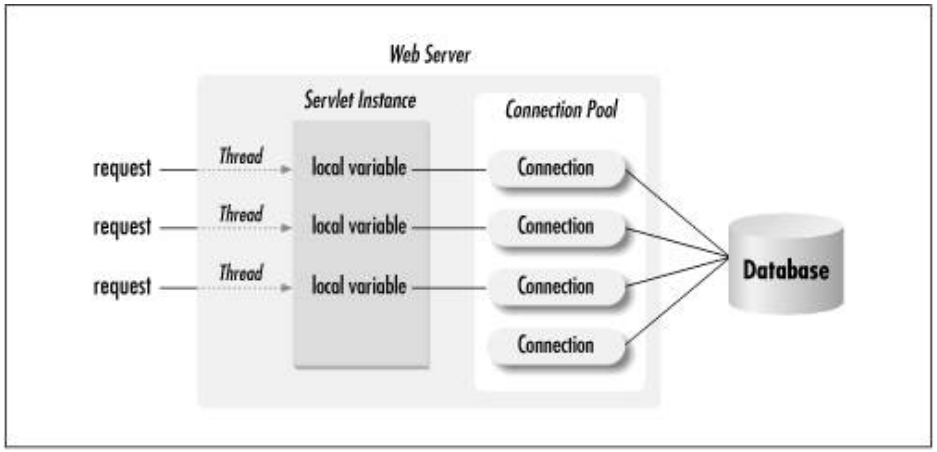
OrderHandler.java

```
public class OrderHandler extends HttpServlet {
 public void doPost(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
   res.setContentType("text/plain");
   PrintWriter out = res.getWriter();

   Connection con = null;
   try {
     Class.forName(" com.mysql.jdbc.Driver ");
     con = DriverManager.getConnection(" jdbc:mysql://localhost/test", "user", "passwd");

     // Turn on transactions
     con.setAutoCommit(false);

     Statement stmt = con.createStatement();
     stmt.executeUpdate("UPDATE INVENTORY SET STOCK = (STOCK - 10) WHERE PRODUCTID = 7");
     stmt.executeUpdate("UPDATE SHIPPING SET SHIPPED = (SHIPPED + 10) WHERE PRODUCTID = 7");

     chargeCard();  // method doesn't actually exist...
     con.commit();
     out.println("Order successful!  Thanks for your business!");
   }
   catch (Exception e) {
     try {
       con.rollback();
     }
     catch (SQLException ignored) { }
     out.println("Order failed. Please contact technical support.");
   }
```

```
finally {
      // Clean up.
      try {
        if (con != null) con.close();
      }
      catch (SQLException ignored) { }
    }
  }
}
```

Reusing Database Objects

- 커넥션 풀링(Connection Pooling)
  - 단일 Connection-다수 Servlet Thread의 문제
    - Connection이 문제가 발생했을 경우 다른 모든 Thread 사용 불가
    - 한 Connection에서 다수의 Transaction이 발생할 경우 처리 불가
  - Connection Pooling 기법
    - 미리 다수개의 JDBC Connection을 생성하여 Pool에 넣어놓은 후, Servlet Thread에서 요청이 있을 경우 사용중이 아닌 Connection중에 하나를 Servlet이 사용할 수 있도록 해줌
    - Connection의 사용이 끝난 Sevlet Thread는 Connection을 Connection Pool에 반환



Reusing Database Objects

- 예 3: Connection Pool 클래스의 구현

```
public class ConnectionPool {
  private Hashtable connections;
  private int increment;
  private String dbURL, user, password;

  public ConnectionPool(String dbURL,
                        String user,
                        String password,
                        String driverClassName,
                        int initialConnections,
                        int increment)
         throws SQLException, ClassNotFoundException {

    // Load the specified driver class
    Class.forName(driverClassName);

    this.dbURL = dbURL;
    this.user = user;
    this.password = password;
    this.increment = increment;

    connections = new Hashtable();

    // Put our pool of Connections in the Hashtable
    // The FALSE value indicates they're unused
    for(int i = 0; i < initialConnections; i++) {
      connections.put(DriverManager.getConnection(dbURL, user, password),
                      Boolean.FALSE);
    }
  }
```

ConnectionPool.java

Connection이 모두 사용중일 때 추가해서 생성할 Connection의 수
예 : 초기 5 증가 2

Connection을 initialConnection 수 만큼 생성하여 Hashtable을 이용한 connections에 저장해 놓음

Reusing Database Objects

예 3: Connection Pool 클래스의 구현(계속)

```
public Connection getConnection() throws SQLException {
  Connection con = null;

  Enumeration cons = connections.keys();

  synchronized (connnections) {
    while(cons.hasMoreElements()) {
      con = (Connection)cons.nextElement();

      Boolean b = (Boolean)connections.get(con);
      if (b == Boolean.FALSE) {
        // So we found an unused connection.
        // Test its integrity with a quick setAutoCommit(true) call.
        // For production use, more testing should be performed,
        // such as executing a simple query.
        try {
          con.setAutoCommit(true);
        }
        catch(SQLException e) {
          // Problem with the connection, replace it.
          con = DriverManager.getConnection(dbURL, user, password);
        }
        // Update the Hashtable to show this one's taken
        connections.put(con, Boolean.TRUE);
        // Return the connection
        return con;
      }
    }
  }
}
```

ConnectionPool.java

Hashtable을 이용한 connections에 저장해 놓은 Connection들중에서 사용중이 아닌 Connection을 발견함

Servlet이 사용할 Connection을 사용중인 것으로 설정.

Servlet에 Connection을 전송

Reusing Database Objects

- 예 3: Connection Pool 클래스의 구현

```
// If we get here, there were no free connections.
    // We've got to make more.
    for(int i = 0; i < increment; i++) {
      connections.put(DriverManager.getConnection(dbURL, user, password),
                      Boolean.FALSE);
    }

    // Recurse to get one of the new connections.
    return getConnection();
  }

public void returnConnection(Connection returned) {
    Connection con;
    Enumeration cons = connections.keys();
    while (cons.hasMoreElements()) {
      con = (Connection)cons.nextElement();
      if (con == returned) {
        connections.put(con, Boolean.FALSE);
        break;
      }
    }
  }
}
```

Connection이 모두 사용중일 경우
Increment만큼 추가로 Connection 생성

ConnectionPool.java

Connection의 사용이 모두 끝났을 경우
반환할때 불려지는 메소드

Reusing Database Objects

예 4: Connection Pool의 사용

OrderHandlerPool.java

```
public class OrderHandlerPool extends HttpServlet {
  private ConnectionPool pool;

  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
      pool = new ConnectionPool(" jdbc:mysql://localhost/test", "user", "passwd",
                                " com.mysql.jdbc.Driver", 10, 5);
    }
    catch (Exception e) {
      throw new UnavailableException(this, "Couldn't create connection pool");
    }
  }

  public void doPost(HttpServletRequest req, HttpServletResponse res)
                                throws ServletException, IOException {
    Connection con = null;

    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();

    try {
      con = pool.getConnection();

      // Turn on transactions
      con.setAutoCommit(false);

      Statement stmt = con.createStatement();
      stmt.executeUpdate("UPDATE INVENTORY SET STOCK = (STOCK - 10) WHERE PRODUCTID = 7");
      stmt.executeUpdate("UPDATE SHIPPING SET SHIPPED = (SHIPPED + 10) WHERE PRODUCTID = 7");
```

Connection 획득

Reusing Database Objects

- 예 4: Connection Pool의 사용

```
chargeCard();  // method doesn't actually exist...

      con.commit();
      out.println("Order successful!  Thanks for your business!");
    }
    catch (Exception e) {
      // Any error is grounds for rollback
      try {
        con.rollback();
      }
      catch (Exception ignored) { }
      out.println("Order failed. Please contact technical support.");
    }
    finally {
      if (con != null) pool.returnConnection(con);
    }
  }
}
```

OrderHandlerPool.java

Connection 반환

예제 : 삽입, 삭제, 갱신 화면 만들기

- 회원정보 관리