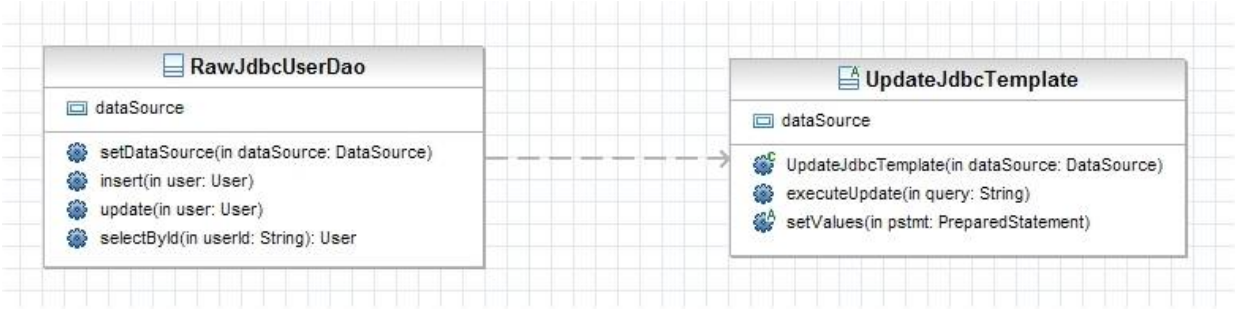
Sustainable Life, Programming, Programmer

Added by 박재성, last edited by 박재성 on 2013년 03월 18일

목차

변경되는 부분을 상속 대신 인자로 해결

- 앞에서 구현한 바와 같이 상속을 통해 변경되는 부분을 처리할 수도 있지만 굳이 상속을 사용할 필요는 없을 듯 하다.
- 상속 대신 메써드 인자로 전달해 해결이 가능하다.
- 먼저 query를 메써드가 아니라 인자로 전달하는 것이 가능하다.



UpdateJdbcTemplate.java

```
package net.javajigi.user;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public abstract class UpdateJdbcTemplate {
  private Connection conn;

  public UpdateJdbcTemplate(Connection conn) {
    this.conn = conn;
  }

  public void update(String query, User user) throws SQLException {
    PreparedStatement pstmt = conn.prepareStatement(query);
    setValues(user, pstmt);

    pstmt.executeUpdate();
  }

  abstract void setValues(User user, PreparedStatement pstmt) throws SQLException;
}
```

- 컴파일 에러가 발생하지 않도록 리팩토링 하려면 먼저 public void update(String query, User user) 메써드를 하나 추가한 후 RawJdbcUserDao에서 이 메써드를 사용하도록 변환한 후 기존의 public void update(User user) 메써드를 제거하면 된다. 이 문서에서는 이 과정은 생략한다.

RawJdbcUserDao.java

```
public class RawJdbcUserDao {
  private Connection conn;

  public RawJdbcUserDao(Connection conn) {
    this.conn = conn;
  }
```

```
  public void insert(User user) throws SQLException {
    UpdateJdbcTemplate template = new UpdateJdbcTemplate(conn) {
      void setValues(User user, PreparedStatement pstmt)
          throws SQLException {
        pstmt.setString(1, user.getUserId());
        pstmt.setString(2, user.getName());
        pstmt.setString(3, user.getEmail());
      }
    };
    String query = "insert into user values(?, ?, ?)";
    template.update(query, user);
  }

  public void update(User user) throws SQLException {
    UpdateJdbcTemplate template = new UpdateJdbcTemplate(conn) {
      void setValues(User user, PreparedStatement pstmt)
          throws SQLException {
        pstmt.setString(1, user.getName());
        pstmt.setString(2, user.getEmail());
        pstmt.setString(3, user.getUserId());
      }
    };
    String query = "update user set name=?, email=? where userid=?";
    template.update(query, user);
  }
}
```

- 위와 같이 구현할 경우 User 클래스 인자는 RawJdbcUserDao에서만 사용하고 있기 때문에 인자로 전달하지 않아도 된다. 따라서 User 인자를 제거하는 리팩토링 작업을 진행한다.

UpdateJdbcTemplate.java

```
package net.javajigi.user;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public abstract class UpdateJdbcTemplate {
  private Connection conn;

  public UpdateJdbcTemplate(Connection conn) {
    this.conn = conn;
  }

  public void update(String query) throws SQLException {
    PreparedStatement pstmt = conn.prepareStatement(query);
    setValues(pstmt);

    pstmt.executeUpdate();
  }

  abstract void setValues(PreparedStatement pstmt) throws SQLException;
}
```

RawJdbcUserDao.java

```
package net.javajigi.user;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class RawJdbcUserDao {
  private Connection conn;

  public RawJdbcUserDao(Connection conn) {
    this.conn = conn;
  }

  public void insert(final User user) throws SQLException {
    UpdateJdbcTemplate template = new UpdateJdbcTemplate(conn) {
      void setValues(PreparedStatement pstmt)
          throws SQLException {
        pstmt.setString(1, user.getUserId());
        pstmt.setString(2, user.getName());
        pstmt.setString(3, user.getEmail());
      }
    };
    String query = "insert into user values(?, ?, ?)";
```

```
    template.update(query);
  }

  public void update(final User user) throws SQLException {
    UpdateJdbcTemplate template = new UpdateJdbcTemplate(conn) {
      void setValues(PreparedStatement pstmt)
          throws SQLException {
        pstmt.setString(1, user.getName());
        pstmt.setString(2, user.getEmail());
        pstmt.setString(3, user.getUserId());
      }
    };
    String query = "update user set name=?, email=? where userid=?";
    template.update(query);
  }
  ...
}
```

- 이 정도까지 리팩토링을 완료하면 Insert, Update, Delete 쿼리에 대해서는 공통적으로 사용할 수 있는 Framework API를 만든 듯 하다. 다음은 Select 쿼리를 지원하기 위한 API를 만들어 나가 보자.