목 차

Ⅰ. **UDS 개요** **2**
UDS 개요 3
환경 설정 4

Ⅱ. **데이터 질의문 인터페이스 사용 방법** **7**
기본 UDS 8
JDBC 기반 UDS 13
오즈 서버 Connection을 사용하는 UDS 16
오즈 서버 Connection을 여러 개 사용하는 UDS 21

Ⅲ. **DataAction 인터페이스 사용 방법** **27**
기본 DataAction 28
마스터-디테일 DataAction 42

Ⅳ. **패러미터 넘겨주는 인터페이스 사용 방법** **6**

Ⅴ. **HttpRequest 접근을 위한 인터페이스 사용 방법** **68**

Ⅵ. **ResultSet 사용 방법** **74**

본 매뉴얼은 오즈 서버, 오즈 애플리케이션 디자이너, 오즈 리포트 디자이너/뷰어, 오즈 쿼리 디자이너 및 자바 프로그램을 사용할 수 있는 중급 이상 개발자를 대상으로 작성된 매뉴얼입니다.

Ⅰ. UDS 개요

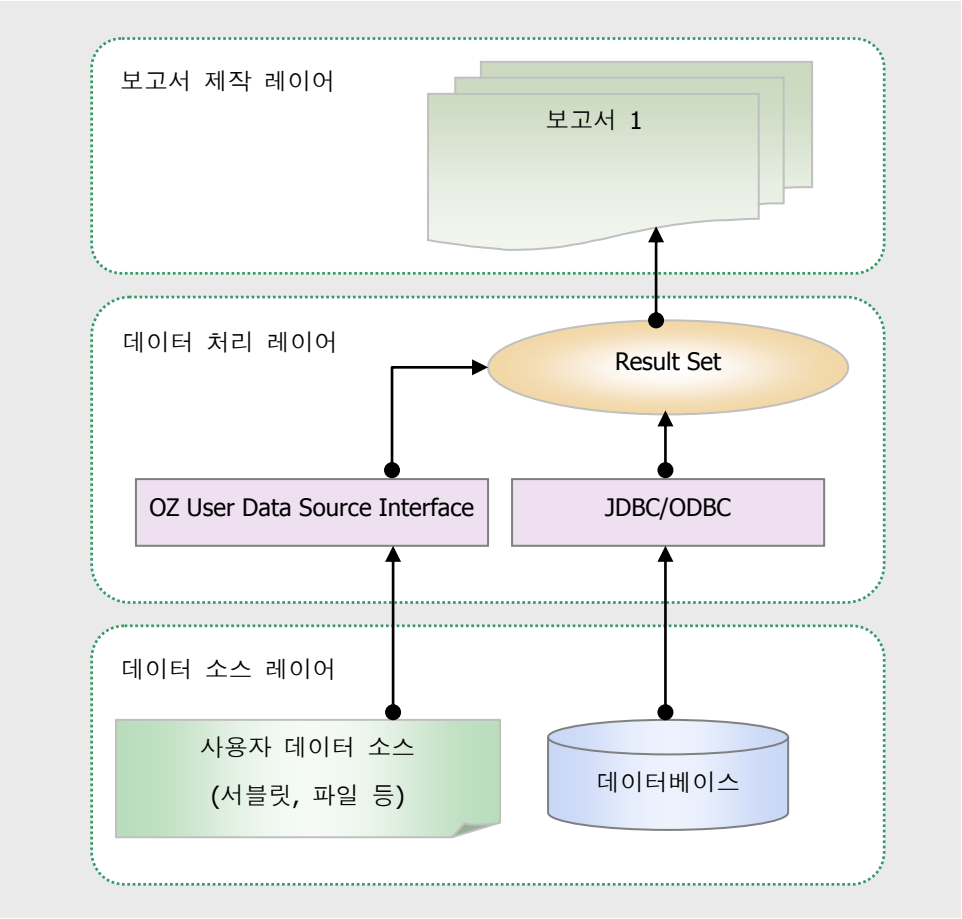
- UDS 개요
- 환경 설정

UDS 개요

UDS(User Data Store)는 Non-DB 데이터 연동을 위한 Java Interface로 데이터베이스에 저장되어 있지 않은 일반 텍스트 파일, CSV, XML 등의 파일 데이터는 물론 EJB, Servlet, ASP, JSP, Applet 등 다양한 형태의 외부 데이터 소스로부터 데이터 셋을 생성하여 보고서를 생성할 수 있도록하는 오즈만의 인터페이스입니다.

UDS 개발자는 자바를 이용하여 다양한 형태의 사용자 데이터를 받아 들이는 부분과 이 데이터를 오즈가 인식할 수 있는 ResultSet 객체로 만들어서 오즈로 넘겨주는 부분을 구현해야 합니다. 이렇게 구현된 자바 프로그램은 클래스 파일로 만들어 보고서와 연동할 수 있으며, 별도의 설정 없이 일반 SQL 데이터 셋과 동일하게 사용할 수 있습니다.

오즈는 사용자 클래스로 패러미터 스트링을 전달할 수 있으며, 패러미터 값에 따라 유동적으로 변화하는 데이터를 전달받을 수 있습니다.



환경 설정

본 매뉴얼에 있는 예제를 실행해보기 위해서는 다음과 같은 환경 설정이 필요합니다.

■ jar 파일 등록

UDS 기능이 구현된 jar 파일(servlet.jar, UDSTestPackage.jar)을 클라이언트 프로그램이 설치된 경로의 lib 폴더에 복사하고, launch.cfg 파일을 편집기로 열어 servlet.jar 파일과 UDSTestPackage.jar 파일의 경로를 추가합니다.

```
CLASSPATH
= .\lib\ozsfw60.jar;.\lib\servlet.jar;.\lib\UDSTestPackage.jar; ...
JRE_PATH = ..\jre15\bin\client\jvm.dll
```

■ ODBC 등록

본 매뉴얼의 예제에 사용될 AppExample.mdb를 ODBC 데이터 원본 관리자의 [시스템 DSN] 탭에 "AppExample" 이름으로 등록합니다.

■ 오즈 서버(타입 : 서블릿) 설정

- 클래스 파일 등록

  오즈 서버의 WEB-INF/classes 디렉토리에 uds_test 폴더를 새로 만든 후 TestUDS_RDBPoolRef.class, TestUDS_RDBPoolRefEX.class 파일을 복사하고, 오즈 서버의 WEB-INF/classes 디렉토리에 oz/uds 폴더를 새로 만든 후 OZUDSSample.class, OZUDSSampleResultSet.class 파일을 복사합니다.

  오즈 서버의 WEB-INF 폴더에 있는 web.xml 파일을 편집 프로그램으로 열어 클래스를 등록합니다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
   "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>

 <servlet>
   <servlet-name>httpserver</servlet-name>
   <servlet-class>oz.server.OZServlet</servlet-class>
 </servlet>
 <servlet>
```

```
  <servlet-name>ozhello</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet>
  <servlet-name>uds_test.TestUDS_RDBPoolRef</servlet-name>
  <servlet-class>uds_test.TestUDS_RDBPoolRef</servlet-class>
</servlet>
<servlet>
  <servlet-name>uds_test.TestUDS_RDBPoolRefEX</servlet-name>
  <servlet-class>uds_test.TestUDS_RDBPoolRefEX</servlet-class>
</servlet>
<servlet>
  <servlet-name>oz.uds.OZUDSSample</servlet-name>
  <servlet-class>oz.uds.OZUDSSample</servlet-class>
</servlet>
<servlet>
  <servlet-name>oz.uds.OZUDSSampleResultSet</servlet-name>
  <servlet-class>oz.uds.OZUDSSampleResultSet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>httpserver</servlet-name>
  <url-pattern>/server</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>httpserver</servlet-name>
  <url-pattern>/server/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>ozhello</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>uds_test.TestUDS_RDBPoolRef</servlet-name>
   <url-pattern>/uds_test.TestUDS_RDBPoolRef</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>uds_test.TestUDS_RDBPoolRefEX</servlet-name>
   <url-pattern>/uds_test.TestUDS_RDBPoolRefEX</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>oz.uds.OZUDSSample</servlet-name>
```

```
    <url-pattern>/oz.uds.OZUDSSample</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>oz.uds.OZUDSSampleResultSet</servlet-name>
    <url-pattern>/oz.uds.OZUDSSampleResultSet</url-pattern>
  </servlet-mapping>

</web-app>
```

- db.properties 설정

  오즈 서버의 conf 폴더에 있는 db.properties 파일에 ODBC 정보를 추가합니다.

```
AppExample.vendor=odbc
AppExample.dsn=AppExample
AppExample.user=
AppExample.password=
AppExample.maxconns=5
AppExample.initconns=2
AppExample.timeout=5

Member.vendor=odbc
Member.dsn=AppExample
Member.user=
Member.password=
Member.maxconns=5
Member.initconns=2
Member.timeout=5
```

Ⅱ. 데이터 질의문 인터페이스 사용 방법

- 기본 UDS
- JDBC 기반 UDS
- 오즈 서버 Connection을 사용하는 UDS
- 오즈 서버 Connection을 여러 개 사용하는 UDS

기본 UDS

기본 UDS인 DefaultUserDataStore 클래스는 OZUserDataStore와 RawDataSource 인터페이스 상속과 함께 로그 내용을 표시하는 기능이 추가된 클래스입니다. UDS를 사용할 경우에는 DefaultUserDataStore 클래스를 상속받아 사용해야 합니다.

```
interface OZUserDataStore
interface RawDataSource
   │
   └── abstract class DefaultUserDataStore
```

■ **oz.uds.UserDataStore 인터페이스**

```
package oz.uds;

import java.sql.*;

public interface OZUserDataStore {
public void init() throws OZUserDataStoreException;
   public ResultSet getResultSet(String argument) throws
OZUserDataStoreException;
   public void freeResultSet(ResultSet rst);
   public void close();
}
```

– init

Prototype	`public void init() throws OZUserDataStoreException`
Definition	사용자 데이터 셋을 초기화합니다. UDS가 처음 생성된 후 한번 호출합니다.

– getResultSet

Prototype	`public ResultSet getResultSet(String argument) throws OZUserDataStoreException`
Definition	오즈 서버나 디자이너로부터 실행문을 넘겨받아 해당 실행문에 맞는 사용자 데이터를 ResultSet 객체로 변환하여 반환합니다.
Argument	*argument* ResultSet 객체를 만들 실행문

- freeResultSet

Prototype	`public void freeResultSet(ResultSet rst)`
Definition	ResultSet을 종료합니다. 오즈 서버나 디자이너에 getResultSet으로 받은 ResultSet을 모두 사용한 후에는 freeResultSet 메소드를 호출하여 ResultSet을 종료하여야 합니다.
Argument	*rst* 종료할 ResultSet 객체

- close

Prototype	`public void close()`
Definition	사용자 데이터 셋을 종료합니다. UDS를 모두 사용한 후 마지막으로 호출하여야 합니다.

■ oz.uds.basic.RawDataSource 인터페이스

```
package oz.uds.basic;

import oz.uds.*;
public interface RawDataSource {
   Object getRawData(String command) throws OZUserDataStoreException;
}
```

- getRawData

Prototype	`Object getRawData(String command) throws OZUserDataStoreException`
Definition	명령어가 unformat 형태의 데이터 예를 들어 int, String 등이 아닌 포맷의 데이터인 경우 사용합니다.
Argument	*command* 명령어

■ DefaultUserDataStore를 상속받아 UDS를 구현한 Example - TestUDS.java

```
package uds_test;

import oz.uds.*;
import oz.uds.basic.DefaultUserDataStore;
import java.sql.*;

public class TestUDS extends DefaultUserDataStore{
  static {
    log = true;
  }
```

```
  public TestUDS() {
    System.out.println("create TestUDS...");
  }

  public synchronized void init() {}

  public synchronized ResultSet getResultSet(String command) {
    // 일반 ResultSet을 리턴할 수 있으나 직접 ResultSet을 구현할 수 있음
    if(command.equals("null")) return null;
    else return new TestUDSResultSet(command);
  }

  public synchronized Object getRawData(String command) throws
OZUserDataStoreException {
    return null;
  }
}
```

- **oz.uds.rs.DefaultResult을 상속받아 ResultSet을 구현한 TestUDSResultSet 샘플 - TestUDSResultSet.java**

```
package uds_test;

import oz.uds.rs.*;

import java.sql.ResultSetMetaData;
import java.sql.SQLException;

//oz.uds.rs.DefaultResultSet을 상속받아 구현한다.
//implements java.sql.ResultSet을 구현할 수 있으나, 그렇게 되면 getString()..
//등등의 메소드를 전부 구현해야 하므로 DefaultResultSet에 이미 구현되어 있음
//만약에 ResultSet 메소드를 변경 사용할 경우 재정의를 하면됨
public class TestUDSResultSet extends DefaultResultSet {
    private int cursor = 0, num_rows = 10;
    private String command = new String("tempResult");
    TestUDSResultSet(String _command) {
        super();
        if (_command != null) command = _command;
    }

    public boolean next() throws SQLException {
        // ResultSet을 직접 구현한다.
        // 본 예는 10개 데이터만 next할 수 있도록 구현한 것임
        cursor++;
```

```
        if (cursor < num_rows)
            return true;
        else
            return false;
    }

    public String getString(String columnName) throws SQLException {
        return command + Integer.toString(cursor);
    }

    public ResultSetMetaData getMetaData() throws SQLException {
        // 일반적인 경우에는 jdbc의 ReusltSetMetaData를 넘길 수 있으며,
        // 직접 개발도 가능하다.
        return new TestUDSResultSetMeta("field1");
    }
}
```

- **oz.uds.rs.DefaultResultMeta을 상속받아 ResultSet을 구현한 TestUDSResultSet 샘플 - TestUDSResultSetMeta.java**

```
package uds_test;

import oz.uds.rs.*;

import java.sql.SQLException;
import java.sql.Types;

//oz.uds.rs.DefaultResultSetMeta를 상속받아 ResultSetMeta를 직접 구현한 샘플
//DefaultResultSetMeta 클래스는 java.sql.ResultSetMeta interface를 구현한 클래스임
//만약 함수중에 내용을 다르게 하려면 해당 메소드를 재정의하면 됨
public class TestUDSResultSetMeta extends DefaultResultSetMeta{
    private String columnName = new String("tempField");
    public TestUDSResultSetMeta(String _columnName) {
        columnName = _columnName;
    }
    public int getColumnCount() throws SQLException{
        return 1;
    }
    public String getColumnName(int column) throws SQLException{
        return columnName;
    }
    public String getColumnLabelName(int column) throws SQLException{
        return columnName;
    }
    public int getColumnType(int column) throws SQLException{
```

```
        return Types.VARCHAR;
    }
    public String getColumnLabel(int int0) throws SQLException{
      return columnName;
    }
}
```

■ 적용 예

기본 UDS의 데이터를 보고서에 출력하는 예제를 살펴보겠습니다.

➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "CLASS_TestUDS"로, 클래스 파일 경로는 "TestUDS.class"로 지정하여 사용자 데이터 스토어를 추가하고, 스토어에 "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 실행문을 입력하고 쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class1.odi" 파일로 저장합니다.

➢ 보고서 디자인하기

리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class1.odi" 파일을 추가합니다.

보고서에 데이터 밴드를 추가하고 데이터 밴드의 "ODI 이름" 속성을 "class1"로, "데이터셋" 속성을 "SET_1"로 설정합니다.

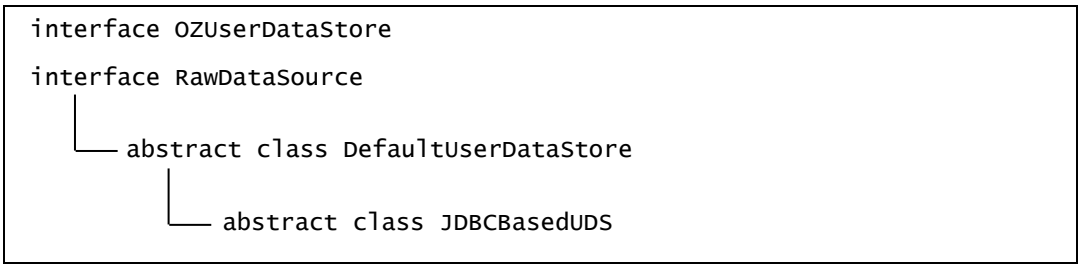
데이터 트리 창에서 "SET_1"의 "field1" 필드를 선택한 후 데이터 밴드로 드래그&드롭하여 추가합니다.

➢ 미리보기

툴바의 미리보기 아이콘을 클릭하여 보고서를 미리보기합니다.

JDBC 기반 UDS

JDBC 기반 UDS인 JDBCBasedUDS 클래스는 DefaultUserDataStore 클래스에서 상속받아 getjdbcName(), getjdbcURL() 메소드가 추가된 클래스입니다.



■ **JDBCBasedUDS를 상속받아 UDS를 구현한 Example - TestJDBCUDS.java**

아래 샘플처럼 getjdbcName()와 getjdbcURL()을 정의해주면 init()시에 해당 이름과 URL의 Connection을 가져옵니다.

```
package uds_test;

import oz.uds.*;
import oz.uds.basic.*;
import java.sql.*;

public class TestJDBCUDS extends JDBCBasedUDS {
   public void init() throws OZUserDataStoreException
   {
      super.init();
      DefaultUserDataStore.log = true;
   }
   public String getjdbcName() {
      return "sun.jdbc.odbc.JdbcOdbcDriver";
   }
   public String getjdbcURL() {
      return "jdbc:odbc:AppExample";
   }
   // JDBCBasedUDS 클래스에 이미 아래와 같이 구현되어 있음
   public       ResultSet       getResultSet(String       command)       throws
OZUserDataStoreException {
      if(log)log("################ getResultSet ##################");
      try {
```

```
            command = processCommand(command);
            ResultSet rst = stmt.executeQuery(command);
            return rst;
        }
        catch (SQLException e) {
            if(log)e.printStackTrace(ps);
            throw new OZUserDataStoreException(e.getMessage());
        }
    }
    // JDBCBasedUDS 클래스에 이미 아래와 같이 구현되어 있음
    public Object getRawData(String command) throws
OZUserDataStoreException {
        return null;
    }
    // JDBCBasedUDS 클래스에 이미 아래와 같이 구현되어 있음
    public void close() {
        super.close();
        try {
            if(stmt != null)stmt.close();
            stmt = null;
            if(conn != null)conn.close();
            conn = null;
        }
        catch (SQLException e) {
            if(log)e.printStackTrace(ps);
        }
    }
}
```

■ 적용 예

JDBC 기반 UDS의 데이터를 보고서에 출력하는 예제를 살펴보겠습니다.

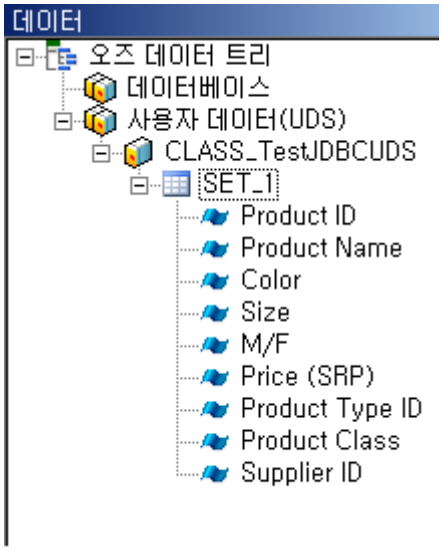
➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "CLASS_TestJDBCUDS"로, 클래스 파일 경로는 "TestJDBCUDS.class"로 지정하여 사용자 데이터 스토어를 추가하고, 스토어에 "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 아래와 같이 입력하고

```
select * from Product
```

쿼리문 실행 아이콘()을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class2.odi" 파일로 저장합니다.

- 보고서 디자인하기

  리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class2.odi" 파일을 추가합니다.

  보고서에 데이터 밴드를 추가하고 데이터 밴드의 "ODI 이름" 속성을 "class1"로, "데이터셋" 속성을 "SET_1"로 설정합니다.

  데이터 밴드에 테이블을 추가하고 테이블에 표시할 필드를 추가합니다.

- 미리보기

  툴바의 미리보기 아이콘을 클릭하여 보고서를 미리보기합니다.

오즈 서버 Connection을 사용하는 UDS

UDS 클래스에서 오즈 서버 DB Connection Pool의 Connection을 공유해서 사용하기 위해서는 oz.uds.OZRDBPoolRef.java 인터페이스를 구현해야 합니다.

■ **oz.uds.OZRDBPoolRef 인터페이스**

```
package oz.uds;

/**
  * RDB를 사용하는 UDS 에서 서버의 ConnectionPool 을 사용할 때
  * 이 interface 를 implements 해야 한다.
  * 이 클래스를 implements 하는 UDS 클래스의 close() 메서드에서
  * 임의로 connection을 close() 하면 안된다.
*/

public interface OZRDBPoolRef {
    /**
     * 현재 UDS 가 사용할 DB 의 Alias 이름
     * 이 이름은, db.properties 에 정의되어 있어야 한다.
     * @return alias name
     */
    public String getDBAlias();

    /**
     * 서버의 ConnectionPool로부터 얻어온 Connection을
     * UDS 의 Connection 에 연결한다.
     * obj[0] - java.sql.Connection : Connection
     * obj[1] - java.lang.String    : Session ID of Connection
     * @param obj Connection Info.
     */
    public void setDBConnection(Object[] obj);
}
```

- getDBAlias

Prototype	public String getDBAlias()
Definition	UDS가 사용할 DB Pool의 Alias 이름을 리턴합니다. 이 메소드는 서버에서 호출합니다.

- setDBConnection

Prototype	`public void setDBConnection(Object[] obj)`	
Definition	UDS가 사용할 Connection을 서버 DB Pool로부터 설정합니다. 이 메소드는 서버에서 호출합니다.	
Argument	*obj*	Alias에 해당되는 Connection이 DB Pool에 있을 때 설정할 Connection 정보 obj[0] : Connection obj[1] : Session ID

※ 참고사항 : 서버에서 메소드 호출 순서

① getDBAlias(해당 Connection을 DB Pool에서 찾기)

② setDBConnection(검색된 Connection을 UDS에 설정)

③ init(초기화)

■ 제약 사항

- OZRDBPoolRef를 통해 할당받는 Connection은 하나의 ODI에서 동일한 RDB를 사용하는 모든 Store들이 공유해서 사용하므로, Store별로 Auto Commit 속성을 설정했을 경우에는 주의해야 합니다.
- OZRDBPoolRef 인터페이스를 implements한 클래스에서 사용하는 Connection은 서버의 Pool에서 관리되므로 close() 메소드를 호출하여 임의로 Connection을 Close해서는 안됩니다.
- OZRDBPoolRef 인터페이스를 implements한 클래스는 서버와 쿼리 디자이너에서 초기화하는 방법이 각각 다르므로 init(), close()를 주의해서 작성해야 합니다.
- 일반적으로 init에서 Connection을 생성하던 UDS와 달리 OZRDBPoolRef 인터페이스를 implements한 UDS는 Connection을 setDBConnections()에 의해 서버로부터 받습니다.
- UDS가 서버에서 생성될 때에는 setDBConnection() → init() 순서로 호출됩니다.

■ OZRDBPoolRef를 구현한 UDS Example - TestUDS_RDBPoolRef.java

```
package uds_test;

import java.sql.*;

import oz.uds.*;
import oz.uds.basic.DefaultUserDataStore;
import oz.uds.rs.*;
```

```
public class TestUDS_RDBPoolRef extends DefaultUserDataStore implements
OZRDBPoolRef {

   private Connection conn;
   private Statement stmt;

   /**
    * init 에서 연결할 DB 정보와 동일한 alias 이름을 리턴한다.
    * Connection 정보는 서버의 db.properties 에도 동일한 정보가 존재해야 된다.
    * @return alias name
    */
   public String getDBAlias() {
      return "AppExample";
   }

   /**
    * 서버의 ConnectionPool 로 부터 얻어온 Connection 을
    * UDS 의 Connection 에 연결한다.
    * obj[0] - java.sql.Connection : Connection
    * obj[1] - java.lang.String    : Session ID of Connection
    * @param obj Connection Info.
    */
   public void setDBConnection(Object[] obj) {
      try {
         conn = (Connection)obj[0];
      }
      catch(Exception e){
         e.printStackTrace();
      }
}

   public void init() throws OZUserDataStoreException {
      try {
         stmt = conn.createStatement();
      }
      catch(Exception e){
         e.printStackTrace();
         throw new OZUserDataStoreException(e.getMessage());
      }
   }

   public void close() {
      try {
         if(stmt != null) {
```

```
            try {
                stmt.close();
                stmt = null;
            }
            catch(Exception e) {
            }
        }
}
      catch(Exception e){
         e.printStackTrace();
      }
   }

   public Object getRawData(String string) throws OZUserDataStoreException {
      return null;
   }

   public       ResultSet       getResultSet(String       command)       throws
OZUserDataStoreException {
      try {
         System.out.println("COMMAND :" + command);
         return stmt.executeQuery(command);
      }
      catch(Exception e) {
         e.printStackTrace();
         throw new OZUserDataStoreException(e.toString());
      }
   }

   public void freeResultSet(ResultSet rst) {
      if(rst != null) {
         try {
            rst.close();
         }
         catch(Exception e) {
         }
      }
   }
}
```

■ 적용 예

오즈 서버의 Connection을 사용하는 UDS의 데이터를 보고서에 출력하는 예제를 살펴보겠습니다.

➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "CLASS_TestUDS_RDBPoolRef"로, 클래스 파일 경로는 "TestUDS_RDBPoolRef.class"로 지정하여 사용자 데이터 스토어를 추가하고, 스토어에 "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 실행문을 입력하고 쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class3.odi" 파일로 저장합니다.

➢ 보고서에 디자인하기

리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class3.odi" 파일을 추가합니다.

보고서에 데이터 밴드를 추가하고 데이터 밴드의 "ODI 이름" 속성을 "class3"으로, "데이터셋" 속성을 "SET_1"로 설정합니다.

데이터 밴드에 테이블을 추가하고 테이블에 표시할 필드를 추가합니다.

디자인한 보고서를 "class3.ozr"로 저장한 후 ODI와 OZR을 서버 리파지토리에 업로드합니다.

➢ 웹에서 보고서 미리보기

보고서를 호출하는 웹 페이지를 작성한 후 웹 브라우저에서 웹 페이지를 실행하여 보고서를 바인딩합니다.

오즈 서버 Connection을 여러 개 사용하는 UDS

UDS 클래스에서 오즈 서버의 DB Connection Pool로부터 여러 개의 Connection을 공유해서 사용하기 위해서는 oz.uds.OZRDBPoolRefEX.java 인터페이스를 구현해야 합니다.

■ **oz.uds.OZRDBPoolRefEX 인터페이스**

```
package oz.uds;

import java.util.HashMap;

/**
* Description: RDB를 사용하는 UDS에서 서버의 ConnectionPool로부터
* 복수개의 Connection을 사용할 때 이 interface 를 implements 해야 한다.
* 이 클래스를 implements 하는 UDS 클래스의 close() 메서드에서
* 임의로 connection을 close() 하면 안된다.
*/

public interface OZRDBPoolRefEX {
    /**
     * 현재 UDS 가 사용할 DB 의 Alias 이름들
     * 이 이름은, db.properties 에 정의되어 있어야 한다.
     * @return alias name
     */
    public String[] getDBAliasArray();
    /**
     * 서버의 ConnectionPool 로 부터 얻어온 Connection 들을
     * UDS 의 Connection 에 연결한다.
     * map - KEY:alias, VALUE:Object[2]
     *     obj[0] - java.sql.Connection : Connection
     *     obj[1] - java.lang.String    : Session ID of Connection
     */
    public void setDBConnections(HashMap map);
}
```

- getDBAliasArray

Prototype	public String getDBAliasArray()
Definition	UDS가 사용할 DB Pool의 앨리어스 이름을 가져옵니다.

- setDBConnections

Prototype	`public void setDBConnections(HashMap map)`	
Definition	UDS가 사용할 Connection을 서버 DB Pool로부터 검색하여 HashMap 형태로 설정합니다. 서버에서 앨리어스 이름의 Connection을 찾지 못했을 경우 null입니다.	
Argument	*map*	key : alias name value : Object[2] [0] : Connection [1] : Session ID

※ 참고사항 : 서버에서 메소드 호출 순서

④ getDBAliasArray(해당 Connection을 DB Pool에서 찾기)

⑤ setDBConnections(검색된 Connection을 UDS에 설정)

⑥ init(초기화)

■ 제약 사항

- OZRDBPoolRefEX를 통해 할당받는 Connection은 하나의 ODI에서 동일한 RDB를 사용하는 모든 Store들이 공유해서 사용하므로, Store별로 Auto Commit 속성을 설정했을 경우에는 주의해야 합니다.
- OZRDBPoolRefEX 인터페이스를 implements한 클래스에서 사용하는 Connection은 서버의 Pool에서 관리되므로 close() 메소드를 호출하여 임의로 Connection을 Close해서는 안됩니다.
- OZRDBPoolRefEX 인터페이스를 implements한 클래스는 서버와 쿼리 디자이너에서 초기화하는 방법이 각각 다르므로 init(), close()를 주의해서 작성해야 합니다.
- 일반적으로 init에서 Connection을 생성하던 UDS와 달리 OZRDBPoolRef 인터페이스를 implements한 UDS는 Connection을 setDBConnections()에 의해 서버로부터 받습니다.
- UDS가 서버에서 생성될 때에는 setDBConnections() → init() 순서로 호출됩니다.

■ OZRDBPoolRefEX를 구현한 UDS Example - TestUDS_RDBPoolRefEX.java

```
package uds_test;

import java.sql.*;
import java.util.*;

import oz.uds.*;
import oz.uds.basic.DefaultUserDataStore;
```

```
import oz.uds.rs.*;

public class TestUDS_RDBPoolRef extends DefaultUserDataStore implements
OZRDBPoolRefEX {

   private Connection conn1;
   private Connection conn2;
   private Connection conn3;

   private Statement stmt1;
   private Statement stmt2;
   private Statement stmt3;

   String[] aliases = {"AppExample", "Member", "Unknown" };

   /**
    * 현재 UDS 가 사용할 DB 의 Alias 이름들.
    * 이 이름은, db.properties 에 정의되어 있어야 한다.
    * @return alias name
    */
   public String[] getDBAliasArray() {
      return aliases;
   }

   /**
    * 서버의 ConnectionPool 로 부터 얻어온 Connection 들을
    * UDS 의 Connection 에 연결한다.
    * map [KEY:alias, VALUE:Object[2]
    *    obj[0] - java.sql.Connection : Connection
    *    obj[1] - java.lang.String    : Session ID of Connection
    * @param obj Connection Info.
    */
   public void setDBConnections(HashMap map) {
      try {
         Object[] obj = null;

         obj = (Object[])map.get(aliases[0]);
         if(obj != null) conn1 = (Connection)obj[0];

         obj = (Object[])map.get(aliases[1]);
         if(obj != null) conn2 = (Connection)obj[0];

         obj = (Object[])map.get(aliases[2]);
         if(obj != null) conn3 = (Connection)obj[0];
```

```
        }
        catch(Exception e){
            e.printStackTrace();
        }
}

    public void init() throws OZUserDataStoreException {
        try {
            stmt1 = conn1.createStatement();
            stmt2 = conn2.createStatement();
            stmt3 = conn3.createStatement();
        }
        catch(Exception e){
            e.printStackTrace();
            throw new OZUserDataStoreException(e.getMessage());
        }
    }

    public void close() {
        try {
            if (stmt1 != null) stmt1.close();
            if (stmt2 != null) stmt2.close();
            if (stmt3 != null) stmt3.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    public Object getRawData(String string) throws OZUserDataStoreException {
        return null;
    }

    public ResultSet getResultSet(String command) throws
OZUserDataStoreException {
        try {
            /**
             * 여기서 stmt1,2,3을 command에 의해 처리해주면 된다.
             * 샘플에서는 stmt1만 사용하도록 한다.
             */
            System.out.println("COMMAND :" + command);
            return stmt1.executeQuery(command);
        }
        catch(Exception e) {
```

```
            e.printStackTrace();
            throw new OZUserDataStoreException(e.toString());
        }
    }

    public void freeResultSet(ResultSet rst) {
        if(rst != null) {
            try {
                rst.close();
            }
            catch(Exception e) {
            }
        }
    }
}
```

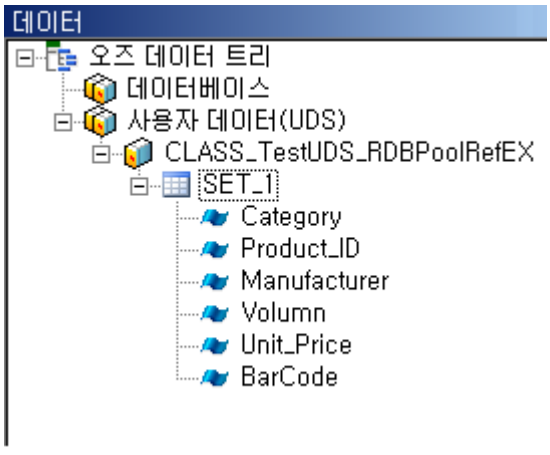
■ 적용 예

오즈 서버 Connection을 여러 개 사용하는 UDS의 데이터를 보고서에 출력하는 예제를 살펴보겠습니다.

➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "CLASS_TestUDS_RDBPoolRefEX"로, 클래스 파일 경로는 "TestUDS_RDBPoolRefEX.class"로 지정하여 사용자 데이터 스토어를 추가하고, 스토어에 "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 실행문을 입력하고 쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class4.odi" 파일로 저장합니다.

- 보고서에 디자인하기

  리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class4.odi" 파일을 추가합니다.
  보고서에 데이터 밴드를 추가하고 데이터 밴드의 "ODI 이름" 속성을 "class4"로, "데이터셋" 속성을 "SET_1"로 설정합니다.
  데이터 밴드에 테이블을 추가하고 테이블에 표시할 필드를 추가합니다.
  디자인한 보고서를 "class4.ozr"로 저장한 후 ODI와 OZR을 서버 리파지토리에 업로드합니다.

- 웹에서 보고서 미리보기

  보고서를 호출하는 웹 페이지를 작성한 후 웹 브라우저에서 웹 페이지를 실행하여 보고서를 바인딩합니다.

Ⅲ. DataAction 인터페이스 사용 방법

- 기본 DataAction
- 마스터-디테일 DataAction

기본 DataAction

UDS에서 DataAction을 처리하기 위해서는 oz.uds.OZFrameworkDataStore 인터페이스를 구현하여야 합니다.

■ **oz.uds.OZFrameworkDataStore 인터페이스**

```
package oz.uds;

import java.sql.*;

public interface OZFrameworkUserDataStore extends OZUserDataStore
{
    /**
     * insert a row.
     *
     * it can be implemented as following SQL like logic.
     *
     *    INSERT INTO 'table' (src_fields[0], field_namses[1], ...)
     *                VALUES(src_values[0], src_value[1], ...)
     *
     * extra arugment can be used to convert spacial types (TO_DATE etc) or
     * modify data.
     *
     * @param src_fields
     *    field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param src_values
     *    the target row's field data array faired with 'src_fields'.
     *    it's size can be 0 to the number of fields in the table.
     * @param ext
     *    extra argument if needs.
     * @return
     *    number of inserted rows. (always 1 if normal)
     * @throws OZUserDataStoreException
     * @since 3.0
     */
    public abstract int insertRow(String cmd, String[] src_fields, Object[]
src_values, String ext) throws OZUserDataStoreException;
    /**
     * delete row(s).
```

```
     *
     * it can be implemented as following SQL like logic.
     *
     *    DELETE FROM 'table' WHERE dst_fields[0] = dst_values[0]
     *                        AND dst_fields[1] = dst_values[1]
     *                        AND ...
     *                        AND extra condition
     *
     * @param dst_fields
     *    field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param dst_values
     *    the target row's field data array faired with 'field_names'.
     *    it's size can be 0 to the number of fields in the table.
     *    the type of value is one of following.
     *
     *        [Java types]        [SQL types]
     *        java.io.InputStream  BINARY VARBINARY LONGVARBINARY BLOB
     *        java.lang.String     CHAR LONGVARCHAR VARCHAR CLOB
     *        java.sql.Date        DATE
     *        java.sql.Time        TIME
     *        java.sql.Timestamp   TIMESTAMP
     *        java.lang.Integer    SMALLINT TUNYINT INTEGER
     *        java.lang.Double     BIGINT FLOAT DOUBLE REAL NUMERIC DECIMAL
     *        java.lang.Boolean    BIT
     *
     * @param ext
     *    extra condition if needs.
     * @return
     *    number of deleted rows.
     * @throws OZUserDataStoreException
     * @since 3.0
     */
    public    abstract    int    deleteRow(String    cmd,    String[]    dst_fields,
Object[] dst_values, String ext) throws OZUserDataStoreException;
    /**
     * update row(s).
     *
     * it can be implemented as following SQL like logic.
     *
     *    UPDATE 'table' SET src_fields[0] = src_values[0],
     *                       src_field[1] = src_values[1],
     *                       ...
     *          WHERE dst_fields[0] = dst_values[0]
```

```
     *           AND dst_fields[1] = dst_values[1]
     *           AND ...
     *           AND ext
     *
     * extra arugment can be used to convert spacial types (TO_DATE etc) or
     * modify data.
     *
     * @param dst_fields
     *    destination row's field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param dst_values
     *    destination row's field data array faired with 'dst_fields'.
     *    it's size can be 0 to the number of fields in the table.
     * @param src_fields
     *    the source field name array to update.
     *    it's size varies from 0 to the number of fields in the table.
     * @param src_values
     *    the source field data array aligned with 'dst_fields' to update.
     *    it's size can be 0 to the number of fields in the table.
     * @param argument
     *    extra argument if needs.
     * @return
     *    number of updated rows.
     * @throws OZUserDataStoreException
     * @since 3.0
     */
    public abstract int updateRow(String cmd,
                            String[] dst_fields, Object[] dst_values,
                            String[] src_fields, Object[] src_values,
                            String ext)
       throws OZUserDataStoreException;
    /**
     * commit all data actions
     *
     * called after all data action transactions are successfully completed.
     *
     * @since 3.0
     */
    public abstract void commit() throws OZUserDataStoreException;

    /**
     *
     * rollback all data actions
     *
```

```
     * called if exception occured while doing data actions.
     *
     * @since 3.0
     */
    public abstract void rollback()throws OZUserDataStoreException;
}
```

- insertRow

Prototype	public abstract int insertRow(String cmd, String[] src_fields, Object[] src_values, String ext) throws OZUserDataStoreException	
Definition	행을 삽입하고 데이터를 추가합니다.	
Argument	*cmd*	행 삽입할 쿼리문
	src_fields	행 삽입할 필드명 Array
	src_values	행 삽입할 필드의 값 Array
	ext	추가할 여분의 내용

- deleteRow

Prototype	public abstract int deleteRow(String cmd, String[] dst_fields, Object[] dst_values, String ext) throws OZUserDataStoreException	
Definition	행을 삭제합니다.	
Argument	*cmd*	행 삭제할 쿼리문
	dst_fields	행 삭제할 필드명 Array
	dst_values	행 삭제할 필드의 값 Array
	ext	추가할 여분의 내용

- updateRow

Prototype	public abstract int updateRow(String cmd, String[] dst_fields, Object[] dst_values, String[] src_fields, Object[] src_values, String ext) throws OZUserDataStoreException	
Definition	행을 변경합니다.	
Argument	*cmd*	행 변경할 쿼리문
	dst_fields	행 변경할 타겟 필드명 Array
	dst_values	행 변경할 타겟 필드의 값 Array

	src_fields	행 변경할 소스 필드명 Array
	src_values	행 변경할 소스 필드의 값 Array
	ext	추가할 여분의 내용

- commit

Prototype	public abstract void commit() throws OZUserDataStoreException
Definition	DataAction을 Commit(실행)합니다.

- rollback

Prototype	public abstract void rollback() throws OZUserDataStoreException
Definition	Commit한 내용을 rollback합니다.

■ **OZFrameworkUserDataStore를 구현한 UDS Example - DefaultOZFrameworkUDS.java**

```
package oz.uds.AppExample;

import oz.uds.*;
import oz.uds.basic.*;

public class DefaultOZFrameworkUDS extends JDBCBasedUDS implements
OZFrameworkUserDataStore
{
   public void init()
      throws OZUserDataStoreException
   {
      super.init();
      DefaultUserDataStore.log = true;
   }

   public String getjdbcName() {
      return "sun.jdbc.odbc.JdbcOdbcDriver";
   }

   public String getjdbcURL() {
      return "jdbc:odbc:AppExample";
   }

   public int insertRow(String cmd,
                  String[] src_fields, Object[] src_values,
```

```
                    String ext)
      throws OZUserDataStoreException
   {
      StringBuffer info = new StringBuffer();
      info.append("TEST_DAC_UDS : insertRow ----------------------------------
\n");
      info.append("CMD :" + cmd + "\n");
      for(int i=0; i<src_fields.length; i++) {
         info.append("SF"+i+"["+src_fields[i]+"] :" + src_values[i] + "\n");
      }
      info.append("Ext :" + ext + "\n\n");
      info.append("------------------------------------------------------------
\n");
      System.out.println(info.toString());

      try {
         return stmt.executeUpdate(cmd);
      }
      catch(java.sql.SQLException sqlexception) {
         throw new OZUserDataStoreException(sqlexception.getMessage());
      }
   }

   public int deleteRow(String cmd, String[] dst_fields, Object[] dst_values,
String ext)
      throws OZUserDataStoreException
   {
      StringBuffer info = new StringBuffer();
      info.append("TEST_DAC_UDS : deleteRow ----------------------------------
\n");
      info.append("CMD :" + cmd + "\n");
      for(int i=0; i<dst_fields.length; i++) {
         info.append("DF"+i+"["+dst_fields[i]+"] :" + dst_values[i] + "\n");
      }
      info.append("Ext :" + ext + "\n\n");
      info.append("------------------------------------------------------------
\n");
      System.out.println(info.toString());

      try {
         return stmt.executeUpdate(cmd);
      }
      catch(java.sql.SQLException sqlexception) {
         if(DefaultUserDataStore.log) {
```

```
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
         }
         throw new OZUserDataStoreException(sqlexception.getMessage());
      }
   }

   public int updateRow(String cmd,
                  String[] dst_fields, Object[] dst_values,
                  String[] src_fields, Object[] src_values,
                  String ext)
      throws OZUserDataStoreException
   {
      StringBuffer info = new StringBuffer();
      info.append("TEST_DAC_UDS  :  updateRow ----------------------------------
\n");
      info.append("CMD :" + cmd + "\n");
      for(int i=0; i<src_fields.length; i++) {
         info.append("SF"+i+"["+src_fields[i]+"] :" + src_values[i] + "\n");
      }
      for(int i=0; i<dst_fields.length; i++) {
         info.append("DF"+i+"["+dst_fields[i]+"] :" + dst_values[i] + "\n");
      }
      info.append("Ext :" + ext + "\n\n");
      info.append("------------------------------------------------------------
\n");
      System.out.println(info.toString());

      try {
         return stmt.executeUpdate(cmd);
      }
      catch(java.sql.SQLException sqlexception) {
         if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
         }
         throw new OZUserDataStoreException(sqlexception.getMessage());
      }
   }

   public boolean isInsertRowAvailable() {
      return true;
   }

   public String insertCommand(String cmd,
                        String[] src_fields, Object[] src_values,
```

```
                        String ext)
    throws OZUserDataStoreException
{
    StringBuffer result = new StringBuffer("INSERT INTO " + ext + "(");

    for( int i = 0; i < src_fields.length; i++) {
        if (i == (src_fields.length-1)) {
            result.append(src_fields[i]);
        }
        else {
            result.append(src_fields[i]).append(", ");
        }
    }

    result.append(") VALUES ( ");

    for( int i = 0; i < src_values.length; i++) {
        if (i == (src_values.length-1)) {
            result.append(src_values[i]);
        }
        else {
            result.append(src_values[i]).append(", ");
        }
    }

    return result.append(")").toString();
}


public boolean isDeleteRowAvailable() {
    return true;
}

public String deleteCommand(String cmd,
                        String[] dst_fields, Object[] dst_values,
                        String ext)
    throws OZUserDataStoreException
{
    StringBuffer result = new StringBuffer("DELETE FROM " + ext + " WHERE ");

    for( int i = 0; i < dst_fields.length; i++) {
        if (i == 0) {
            result.append(dst_fields[i]).append(" = ").append(dst_values[i]);
        }
```

```
         else {
            result.append("     AND     ").append(dst_fields[i]).append("     =
").append(dst_values[i]);
         }
      }

      return result.toString();
   }

   public boolean isUpdateRowAvailable() {
      return true;
   }

   public String updateCommand(String cmd,
                           String[] dst_fields, Object[] dst_values,
                           String[] src_fields, Object[] src_values,
                           String ext)
      throws OZUserDataStoreException
   {
      StringBuffer result = new StringBuffer("UPDATE " + ext + " SET ");

      for( int i = 0; i < src_fields.length; i++) {
         if (i == (src_fields.length-1)) {
            result.append(src_fields[i]).append(" = ").append(src_values[i]);
         }
         else {
            result.append(src_fields[i]).append("                              =
").append(src_values[i]).append(", ");
         }
      }

      result.append(" WHERE ");

      for( int i = 0; i < dst_fields.length; i++) {
         if (i == 0) {
            result.append(dst_fields[i]).append(" = ").append(dst_values[i]);
         }
         else {
            result.append("     AND     ").append(dst_fields[i]).append("     =
").append(dst_values[i]);
         }
      }

      return result.toString();
```

```
}

/**
 * commit all data actions
 *
 * called after all data action transactions are successfully completed.
 *
 * @since 3.0
 */
public void commit()
    throws OZUserDataStoreException
{
    if(DefaultUserDataStore.log) {
        log("############### commit ##################");
    }

    try {
        conn.commit();
    }
    catch(java.sql.SQLException sqlexception) {
        if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
        }
        throw new OZUserDataStoreException(sqlexception.getMessage());
    }
}

public boolean isCommitAvailable()
    throws OZUserDataStoreException
{
    return true;
}

/**
 *
 * rollback all data actions
 *
 * called if exception occured while doing data actions.
 *
 * @since 3.0
 */
public void rollback()
    throws OZUserDataStoreException
{
```

```
        if(DefaultUserDataStore.log) {
            log("################ rollback ##################");
        }

        try {
            conn.rollback();
        }
        catch(java.sql.SQLException sqlexception) {
            if(DefaultUserDataStore.log) {
                sqlexception.printStackTrace(DefaultUserDataStore.ps);
            }
            throw new OZUserDataStoreException(sqlexception.getMessage());
        }
    }

    public boolean isRollBackAvailable()
        throws OZUserDataStoreException
    {
        return true;
    }

}
```

■ 적용 예

오즈 애플리케이션의 테이블을 이용하여 데이터를 삽입, 삭제, 변경하는 예제를 살펴보겠습니다.

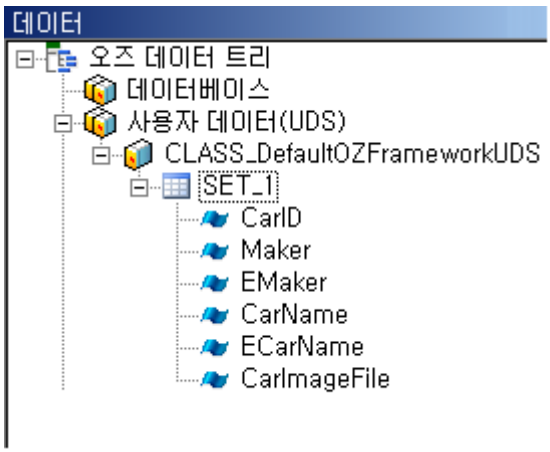
➢ ODI 만들기

쿼리 디자이너를 실행하여 "CLASS_DefaultOZFrameworkUDS"으로 사용자 데이터 스토어를 추가하고, "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 아래와 같이 입력하고

```
select * from CarSales
```

쿼리문 실행 아이콘()을 클릭하여 필드 리스트를 생성합니다.



데이터 셋에 행 삽입, 삭제, 변경 쿼리문을 입력합니다.

- 행 삽입 쿼리

```
insert into CarSales
(#@ARG_SF1#,#@ARG_SF2#,#@ARG_SF3#,#@ARG_SF4#,#@ARG_SF5#,#@ARG_SF6#)
values
('#@ARG_SV1#','#@ARG_SV2#','#@ARG_SV3#','#@ARG_SV4#','#@ARG_SV5#','#@ARG_SV6#')
```

- 행 삭제 쿼리

```
DELETE FROM CarSales
WHERE #@ARG_DF1# = '#@ARG_DV1#'
```

- 행 변경 쿼리

```
UPDATE CarSales SET
#@ARG_SF1# = '#@ARG_SV1#', #@ARG_SF2# = '#@ARG_SV2#',
#@ARG_SF3# = '#@ARG_SV3#', #@ARG_SF4# = '#@ARG_SV4#',
#@ARG_SF5# = '#@ARG_SV5#', #@ARG_SF6# = '#@ARG_SV6#'
WHERE #@ARG_DF1# = '#@ARG_DV1#'
```

ODI 파일을 "class5.odi" 파일로 저장합니다.

➢ 폼 디자인하기

애플리케이션 디자이너를 실행하여 프로젝트를 새로 만든 후 "class5.odi" 파일을 추가합니다.

보드에 테이블을 추가하고 테이블의 "ODIKey" 속성을 "class5"로, "DataSet" 속성을 "SET_1"로, "AllowInsert", "AllowDelete", "AllowUpdate" 속성을 "True"로,

"CellSelectionMode" 속성을 "Single"로 설정합니다.

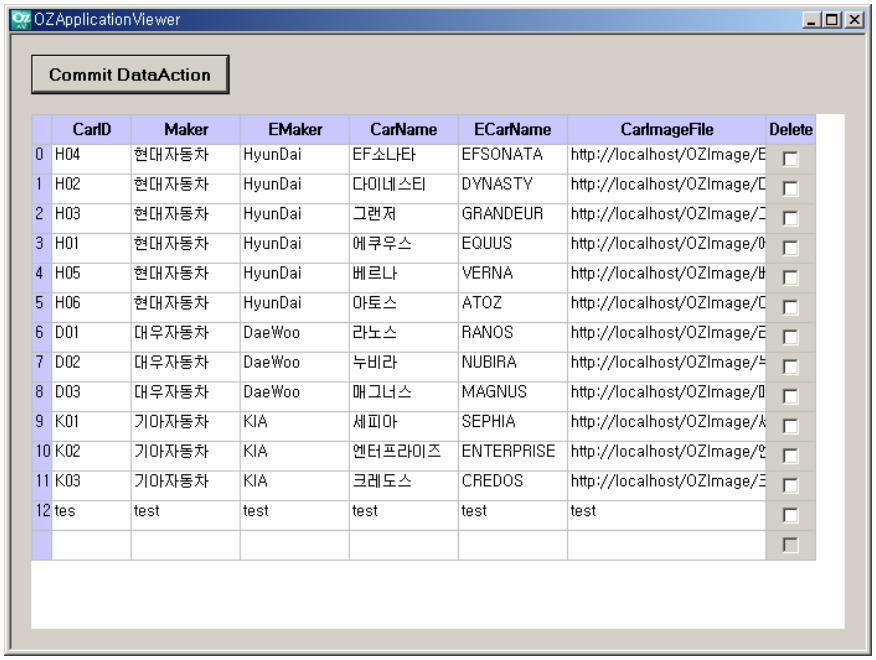
테이블 마법사를 실행하여 "SET_1" 데이터 셋의 필드와 "#OZDeleteFlag#"를 추가하고 추가된 모든 항목을 선택하여 "ColumnEditable" 속성을 "True"로 설정합니다.

보드에 버튼을 추가하고 버튼의 OnClick 이벤트에 아래와 같이 스크립트를 입력합니다.

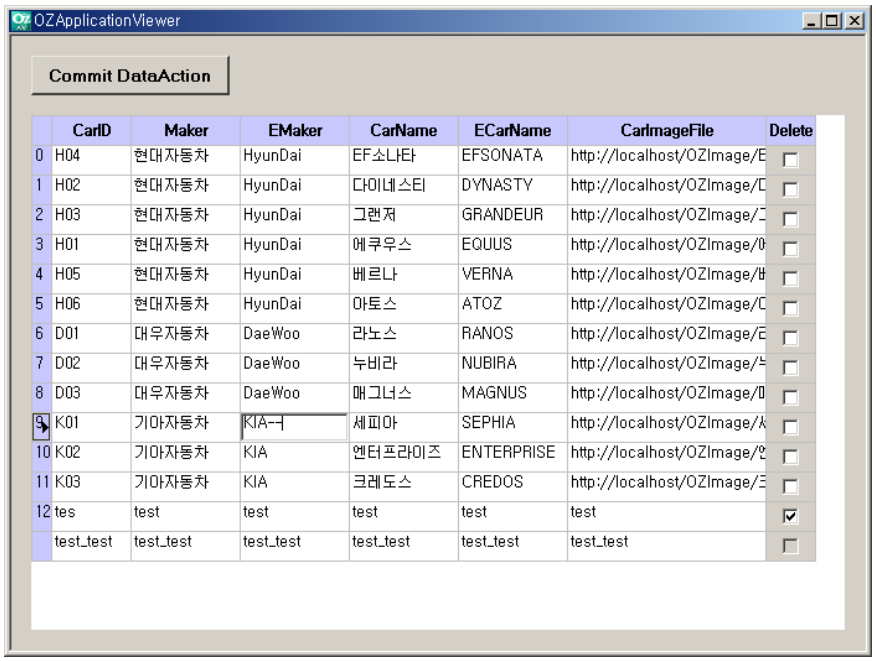
```
Table1.CommitQueuedActions();
Table1.GetDataSet().RefreshDataSet();
```

- DataAction 실행하기

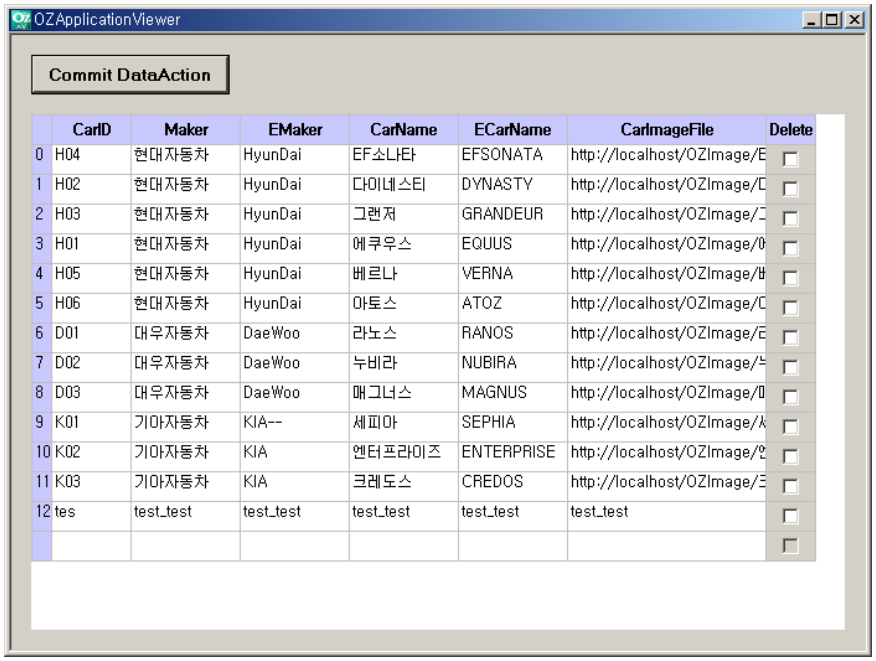
툴바의 미리보기 아이콘을 클릭하여 폼을 미리보기합니다.



맨 마지막에 새로 추가된 행에 데이터를 입력하여 추가합니다. Delete 체크박스를 체크하여 삭제할 행을 선택합니다. 데이터를 변경할 셀을 더블 클릭하여 값을 변경합니다.



[Commit DataAction] 버튼을 클릭하여 DataAction을 실행합니다.



마스터-디테일 DataAction

UDS에서 일반적으로 DataAction 을 처리하기 위해서는 oz.uds.OZFrameworkDataStore 인터페이스를 구현합니다. 하지만 oz.uds.OZFrameworkDataStore 인터페이스는 마스터셋의 값을 사용할 수 없습니다.

그리하여 마스터셋 값을 UDS 클래스에서 바로 사용하기 위해서는 OZUDSDataActionRef 인터페이스를 구현해야합니다.

또한 OZFrameworkDataStore는 insert/delete/update/commit의 결과를 int형으로 리턴하는 반면 OZUDSDataActionRef는 insert/delete/update/commit의 결과를 String형으로 리턴하므로 OZUDSDataActionRef를 사용할 경우 리턴 값을 더 폭넓게 활용할 수 있습니다.

- **oz.uds.OZUDSDataActionRef 인터페이스**

```
package oz.uds;

import java.util.Hashtable;

public interface OZUDSDataActionRef extends OZUserDataStore
{
    /**
     * insert a row.
     *
     * it can be implemented as following SQL like logic.
     *
     *    INSERT INTO 'table' (src_fields[0], field_namses[1], ...)
     *                VALUES(src_values[0], src_value[1], ...)
     *
     * extra arugment can be used to convert spacial types (TO_DATE etc) or
     * modify data.
     *
     * @param src_fields
     *    field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param src_values
     *    the target row's field data array faired with 'src_fields'.
     *    it's size can be 0 to the number of fields in the table.
     * @param ext
     *    extra argument if needs.
     * @param param
```

```
     *    Field name and value of Parameter and MasterSet
     * @return
     *    number of inserted rows. (always 1 if normal)
     * @throws OZUserDataStoreException
     * @since 3.0
     */
    public abstract String insertRow(String cmd,
                            String[] src_fields, Object[] src_values,
                            String ext, Hashtable param)
                            throws OZUserDataStoreException;

    /**
     * delete row(s).
     *
     * it can be implemented as following SQL like logic.
     *
     *    DELETE FROM 'table' WHERE dst_fields[0] = dst_values[0]
     *                        AND dst_fields[1] = dst_values[1]
     *                        AND ...
     *                        AND extra condition
     *
     * @param dst_fields
     *    field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param dst_values
     *    the target row's field data array faired with 'field_names'.
     *    it's size can be 0 to the number of fields in the table.
     *    the type of value is one of following.
     *
     *       [Java types]        [SQL types]
     *       java.io.InputStream  BINARY VARBINARY LONGVARBINARY BLOB
     *       java.lang.String     CHAR LONGVARCHAR VARCHAR CLOB
     *       java.sql.Date        DATE
     *       java.sql.Time        TIME
     *       java.sql.Timestamp   TIMESTAMP
     *       java.lang.Integer    SMALLINT TUNYINT INTEGER
     *       java.lang.Double     BIGINT FLOAT DOUBLE REAL NUMERIC DECIMAL
     *       java.lang.Boolean    BIT
     *
     * @param ext
     *    extra condition if needs.
     * @param param
     *    Field name and value of Parameter and MasterSet
     * @return
```

```
     *    number of deleted rows.
     * @throws OZUserDataStoreException
     * @since 3.0
     */
    public abstract String deleteRow(String cmd,
                            String[] dst_fields, Object[] dst_values,
                            String ext, Hashtable param)
                            throws OZUserDataStoreException;

    /**
     * update row(s).
     *
     * it can be implemented as following SQL like logic.
     *
     *    UPDATE 'table' SET src_fields[0] = src_values[0],
     *                       src_field[1] = src_values[1],
     *                       ...
     *          WHERE dst_fields[0] = dst_values[0]
     *            AND dst_fields[1] = dst_values[1]
     *            AND ...
     *            AND ext
     *
     * extra arugment can be used to convert spacial types (TO_DATE etc) or
     * modify data.
     *
     * @param dst_fields
     *    destination row's field name array.
     *    it's size varies from 0 to the number of fields in the table.
     * @param dst_values
     *    destination row's field data array faired with 'dst_fields'.
     *    it's size can be 0 to the number of fields in the table.
     * @param src_fields
     *    the source field name array to update.
     *    it's size varies from 0 to the number of fields in the table.
     * @param src_values
     *    the source field data array aligned with 'dst_fields' to update.
     *    it's size can be 0 to the number of fields in the table.
     * @param argument
     *    extra argument if needs.
     * @param param
     *    Field name and value of Parameter and MasterSet
     * @return
     *    number of updated rows.
     * @throws OZUserDataStoreException
```

```
     * @since 3.0
     */
    public abstract String updateRow(String cmd,
                              String[] dst_fields, Object[] dst_values,
                              String[] src_fields, Object[] src_values,
                              String ext, Hashtable param)
                             throws OZUserDataStoreException;

    /**
     * commit all data actions
     *
     * called after all data action transactions are successfully completed.
     *
     * @since 3.0
     */
    public abstract String commit()
       throws OZUserDataStoreException;

    /**
     * rollback all data actions
     *
     * called if exception occured while doing data actions.
     *
     * @since 3.0
     */
    public abstract void rollback()
       throws OZUserDataStoreException;
}
```

- insertRow

Prototype	public abstract String insertRow(String cmd, String[] src_fields, Object[] src_values, String ext, Hashtable param) throws OZUserDataStoreException	
Definition	행을 삽입하고 데이터를 추가합니다.	
Argument	*cmd*	행 삽입할 쿼리문
	src_fields	행 삽입할 필드명 Array
	src_values	행 삽입할 필드의 값 Array
	ext	추가할 여분의 내용
	param	마스터셋 패러미터의 필드명과 필드 값 (OZMap 형태로 설정하여야 함)

- deleteRow

Prototype	`public abstract String deleteRow(String cmd, String[] dst_fields, Object[] dst_values, String ext, Hashtable param) throws OZUserDataStoreException`	
Definition	행을 삭제합니다.	
Argument	*cmd*	행 삭제할 쿼리문
	dst_fields	행 삭제할 필드명 Array
	dst_values	행 삭제할 필드의 값 Array
	ext	추가할 여분의 내용
	param	마스터셋 패러미터의 필드명과 필드 값 (OZMap 형태로 설정하여야 함)

- updateRow

Prototype	`public abstract String updateRow(String cmd, String[] dst_fields, Object[] dst_values, String[] src_fields, Object[] src_values, String ext, Hashtable param) throws OZUserDataStoreException`	
Definition	행을 변경합니다.	
Argument	*cmd*	행 변경할 쿼리문
	dst_fields	행 변경할 타겟 필드명 Array
	dst_values	행 변경할 타겟 필드의 값 Array
	src_fields	행 변경할 소스 필드명 Array
	src_values	행 변경할 소스 필드의 값 Array
	ext	추가할 여분의 내용
	param	마스터셋 패러미터의 필드명과 필드 값 (OZMap 형태로 설정하여야 함)

- commit

Prototype	`public abstract String commit() throws OZUserDataStoreException`
Definition	DataAction을 Commit(실행)합니다.

- rollback

Prototype	`public abstract void rollback() throws OZUserDataStoreException`

Definition	Commit한 내용을 rollback합니다.

■ OZMap

insertRow, deleteRow, updateRow 함수에 아규먼트 중 Hashtable param은 OZMap 형태로 구현되어야 합니다.

OZMap 형태는 다음과 같습니다.

```
Key[String]:Set이름, Value[OZMap]:Set값(FieldName, FieldValue)
```

OZMap은 순서가 있는 Map으로 입력한 순서대로 관리되고, Key는 중복되지 않게 설정하여야 합니다.

OZMap의 메소드는 다음과 같습니다.

- clear

Prototype	`public synchronized void clear()`
Definition	Map의 모든 값을 삭제하고 초기화합니다.

- containsKey

Prototype	`public synchronized boolean containsKey(Object key)`
Definition	Map에 해당 Key가 존재하는지 체크합니다. 해당 Key가 존재하면 true를 리턴합니다.
Argument	*key* Key

- containsValue

Prototype	`public boolean containsValue(Object value)`
Definition	Map에 해당 값이 존재하는지 체크합니다. 해당 값이 존재하면 true를 리턴합니다.
Argument	*value* 값

- elements

Prototype	`public Iterator elements()`
Definition	Map에 Value의 Iterator를 리턴합니다.

- get

Prototype	`public Object get(Object key)`

Definition	Key에 매핑되는 값을 리턴합니다. Key에 매핑되는 값이 없으면 null을 리턴합니다.
Argument	*key* Key

- isEmpty

Prototype	`public boolean isEmpty()`
Definition	Map이 비었는지 여부를 체크하여, 비어 있으면 true를 리턴합니다.

- keys

Prototype	`public Iterator keys()`
Definition	Map에서 Key의 Iterator를 리턴합니다.

- put

Prototype	`public synchronized void put(Object key, Object value) throws IllegalArgumentException`
Definition	Map에 Key에 매핑될 값을 설정합니다.
Argument	*key* Map Key
	value Key에 매핑될 값

- remove

Prototype	`public synchronized Object remove(Object key)`
Definition	Map에서 Key에 매핑된 값을 삭제하고, 삭제된 Key에 매핑된 값을 리턴합니다.
Argument	*key* Key

- size

Prototype	`public int size()`
Definition	Map에 Key Set 사이즈를 리턴합니다.

- toString

Prototype	`public String toString()`
Definition	Key에 맵핑된 값을 String 형태로 리턴합니다.

■ OZUDSDataActionRef를 구현한 UDS Example - UDSDataActionMARef.java.java

```
package oz.uds.AppExample;

import java.util.*;

import oz.uds.*;
import oz.uds.basic.*;

import oz.util.OZMap;

/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
 * @author not attributable
 * @version 1.0
 */

public class UDSDataActionMARef extends JDBCBasedUDS implements
OZUDSDataActionRef
{
   public void init()
      throws OZUserDataStoreException
   {
      super.init();
      DefaultUserDataStore.log = true;
   }

   public String getjdbcName() {
      return "sun.jdbc.odbc.JdbcOdbcDriver";
   }

   public String getjdbcURL() {
      return "jdbc:odbc:AppExample";
   }

   public String insertRow(String cmd,
                  String[] src_fields, Object[] src_values,
                  String ext, Hashtable param)
      throws OZUserDataStoreException
   {
      StringBuffer info = new StringBuffer();
      info.append("TEST_DAC_UDS  :  insertRow ----------------------------------
```

```
\n");
       info.append("CMD :" + cmd + "\n");
       for(int i=0; i<src_fields.length; i++) {
          info.append("SF"+i+"["+src_fields[i]+"] :" + src_values[i] + "\n");
       }
       info.append("Ext :" + ext + "\n\n");
       Enumeration sets = param.keys();
       String key = null;
       while(sets.hasMoreElements()) {
          key = (String)sets.nextElement();
          OZMap map = (OZMap)param.get(key);
          Iterator iterK = map.keys();
          Iterator iterV = map.elements();
          while(iterK.hasNext() && iterV.hasNext()) {
             info.append("[" + key + "] "+iterK.next()+":" +iterV.next()+"\n");
          }
       }
       info.append("------------------------------------------------------------
\n");
       System.out.println(info.toString());

       try {
          //insert Action이 있을 경우 리턴할 문자열
         return "insert:"+stmt.executeUpdate(cmd);
       }
       catch(java.sql.SQLException sqlexception) {
          throw new OZUserDataStoreException(sqlexception.getMessage());
       }
   }

   public String deleteRow(String cmd, String[] dst_fields, Object[] dst_values,
String ext, Hashtable param)
       throws OZUserDataStoreException
   {
       StringBuffer info = new StringBuffer();
       info.append("TEST_DAC_UDS  :  deleteRow  ---------------------------------
\n");
       info.append("CMD :" + cmd + "\n");
       for(int i=0; i<dst_fields.length; i++) {
          info.append("DF"+i+"["+dst_fields[i]+"] :" + dst_values[i] + "\n");
       }
       info.append("Ext :" + ext + "\n\n");
       Enumeration sets = param.keys();
       String key = null;
```

```
      while(sets.hasMoreElements()) {
         key = (String)sets.nextElement();
         OZMap map = (OZMap)param.get(key);
         Iterator iterK = map.keys();
         Iterator iterV = map.elements();
         while(iterK.hasNext() && iterV.hasNext()) {
            info.append("[" + key + "] "+iterK.next()+":" +iterV.next()+"\n");
         }
      }
      info.append("------------------------------------------------------------
\n");
      System.out.println(info.toString());

      try {
         //delete Action이 있을 경우 리턴할 문자열
          return "delete:"+stmt.executeUpdate(cmd);
      }
      catch(java.sql.SQLException sqlexception) {
         if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
         }
         throw new OZUserDataStoreException(sqlexception.getMessage());
      }
   }

   public String updateRow(String cmd,
                  String[] dst_fields, Object[] dst_values,
                  String[] src_fields, Object[] src_values,
                  String ext, Hashtable param)
      throws OZUserDataStoreException
   {
      StringBuffer info = new StringBuffer();
      info.append("TEST_DAC_UDS  :  updateRow  ---------------------------------
\n");
      info.append("CMD :" + cmd + "\n");
      for(int i=0; i<src_fields.length; i++) {
         info.append("SF"+i+"["+src_fields[i]+"] :" + src_values[i] + "\n");
      }
      for(int i=0; i<dst_fields.length; i++) {
         info.append("DF"+i+"["+dst_fields[i]+"] :" + dst_values[i] + "\n");
      }
      info.append("Ext :" + ext + "\n\n");
      Enumeration sets = param.keys();
      String key = null;
```

```
      while(sets.hasMoreElements()) {
         key = (String)sets.nextElement();
         OZMap map = (OZMap)param.get(key);
         Iterator iterK = map.keys();
         Iterator iterV = map.elements();
         while(iterK.hasNext() && iterV.hasNext()) {
            info.append("[" + key + "] "+iterK.next()+":" +iterV.next()+"\n");
         }
      }
      info.append("------------------------------------------------------------
\n");
      System.out.println(info.toString());

      try {
         //update Action이 있을 경우 리턴할 문자열
         return "update:"+stmt.executeUpdate(cmd);
      }
      catch(java.sql.SQLException sqlexception) {
         if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
         }
         throw new OZUserDataStoreException(sqlexception.getMessage());
      }
   }

   public boolean isInsertRowAvailable() {
      return true;
   }

   public String insertCommand(String cmd,
                        String[] src_fields, Object[] src_values,
                        String ext)
      throws OZUserDataStoreException
   {
      StringBuffer result = new StringBuffer("INSERT INTO " + ext + "(");

      for( int i = 0; i < src_fields.length; i++) {
         if (i == (src_fields.length-1)) {
            result.append(src_fields[i]);
         }
         else {
            result.append(src_fields[i]).append(", ");
         }
      }
```

```
        result.append(") VALUES ( ");

        for( int i = 0; i < src_values.length; i++) {
            if (i == (src_values.length-1)) {
                result.append(src_values[i]);
            }
            else {
                result.append(src_values[i]).append(", ");
            }
        }

        return result.append(")").toString();
    }

    public boolean isDeleteRowAvailable() {
        return true;
    }

    public String deleteCommand(String cmd,
                            String[] dst_fields, Object[] dst_values,
                            String ext)
        throws OZUserDataStoreException
    {
        StringBuffer result = new StringBuffer("DELETE FROM " + ext + " WHERE ");

        for( int i = 0; i < dst_fields.length; i++) {
            if (i == 0) {
                result.append(dst_fields[i]).append(" = ").append(dst_values[i]);
            }
            else {
                result.append("     AND     ").append(dst_fields[i]).append("     =
").append(dst_values[i]);
            }
        }

        return result.toString();
    }

    public boolean isUpdateRowAvailable() {
        return true;
    }

    public String updateCommand(String cmd,
```

```
                          String[] dst_fields, Object[] dst_values,
                          String[] src_fields, Object[] src_values,
                          String ext)
      throws OZUserDataStoreException
   {
      StringBuffer result = new StringBuffer("UPDATE " + ext + " SET ");

      for( int i = 0; i < src_fields.length; i++) {
         if (i == (src_fields.length-1)) {
            result.append(src_fields[i]).append(" = ").append(src_values[i]);
         }
         else {
            result.append(src_fields[i]).append("                          =
").append(src_values[i]).append(", ");
         }
      }

      result.append(" WHERE ");

      for( int i = 0; i < dst_fields.length; i++) {
         if (i == 0) {
            result.append(dst_fields[i]).append(" = ").append(dst_values[i]);
         }
         else {
            result.append("     AND     ").append(dst_fields[i]).append("     =
").append(dst_values[i]);
         }
      }

      return result.toString();
   }

   /**
    * commit all data actions
    *
    * called after all data action transactions are successfully completed.
    *
    * @since 3.0
    */
   public String commit()
      throws OZUserDataStoreException
   {
      if(DefaultUserDataStore.log) {
         log("################ commit ###################");
```

```
    }

    try {
        conn.commit();
        return "commit";
    }
    catch(java.sql.SQLException sqlexception) {
        if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
        }
        throw new OZUserDataStoreException(sqlexception.getMessage());
    }
}

public boolean isCommitAvailable()
    throws OZUserDataStoreException
{
    return true;
}

/**
 *
 * rollback all data actions
 *
 * called if exception occured while doing data actions.
 *
 * @since 3.0
 */
public void rollback()
    throws OZUserDataStoreException
{
    if(DefaultUserDataStore.log) {
        log("############### rollback ##################");
    }

    try {
        conn.rollback();
    }
    catch(java.sql.SQLException sqlexception) {
        if(DefaultUserDataStore.log) {
            sqlexception.printStackTrace(DefaultUserDataStore.ps);
        }
        throw new OZUserDataStoreException(sqlexception.getMessage());
    }
```

```
    }

    public boolean isRollBackAvailable()
        throws OZUserDataStoreException
    {
        return true;
    }

}
```

■ 적용 예

오즈 애플리케이션의 테이블을 이용하여 데이터를 삽입, 삭제, 변경하는 예제를 살펴보겠습니다.

➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "DataAction"으로, 클래스 파일 경로는 "UDSDataActionMARef.class"으로 설정하여 사용자 데이터 스토어를 추가하고, "MASTER"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 아래와 같이 입력하고

```
select distinct country
from ECustomer
```

쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



"DataAction"에 "DETAIL"로 데이터 셋을 추가한 후 "마스터셋" 속성을 "MASTER"로 설정합니다. "DETAIL" 데이터 셋의 쿼리문 입력 창에 아래와 같이 입력하고

```
select customer_id, fname, lname, member_card, gender, phone1, city
from ECustomer
where ECustomer.country = '#MASTER.country#'
```

쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



DETAIL 데이터 셋에 아래와 같이 행 삽입, 삭제, 변경 쿼리문을 입력합니다.

- 행 삽입 쿼리

```
INSERT INTO ECustomer (
  #@ARG_SF1#,  #@ARG_SF2#,
  #@ARG_SF3#,  #@ARG_SF4#,
  #@ARG_SF5#,  #@ARG_SF6#,
  #@ARG_SF7#,  country
)
VALUES (
  #@ARG_SV1#,  '#@ARG_SV2#',
  '#@ARG_SV3#',  '#@ARG_SV4#',
  '#@ARG_SV5#',  '#@ARG_SV6#',
  '#@ARG_SV7#',  '#MASTER.country#'
)
```

- 행 삭제 쿼리

```
DELETE FROM ECustomer
WHERE #@ARG_DF1# = #@ARG_DV1# and country = '#MASTER.country#'
```

- 행 변경 쿼리

```
UPDATE ECustomer SET
  #@ARG_SF2# = '#@ARG_SV2#',  #@ARG_SF3# = '#@ARG_SV3#',
  #@ARG_SF4# = '#@ARG_SV4#',  #@ARG_SF5# = '#@ARG_SV5#',
  #@ARG_SF6# = '#@ARG_SV6#',  #@ARG_SF7# = '#@ARG_SV7#'
WHERE
  #@ARG_DF1# = #@ARG_DV1# and  country = '#MASTER.country#'
```

ODI 파일을 "class6.odi" 파일로 저장합니다.

- 폼 디자인하기

  애플리케이션 디자이너를 실행하여 프로젝트를 새로 만든 후 "class6.odi" 파일을 추가합니다.

  보드에 콤보박스를 추가하고 콤보박스의 "ODIKey" 속성을 "class6"으로, "DataSet" 속성을 "MASTER"로, "Field" 속성을 "country"로, "FireRowCursorChange" 속성을 "True"로 설정합니다.

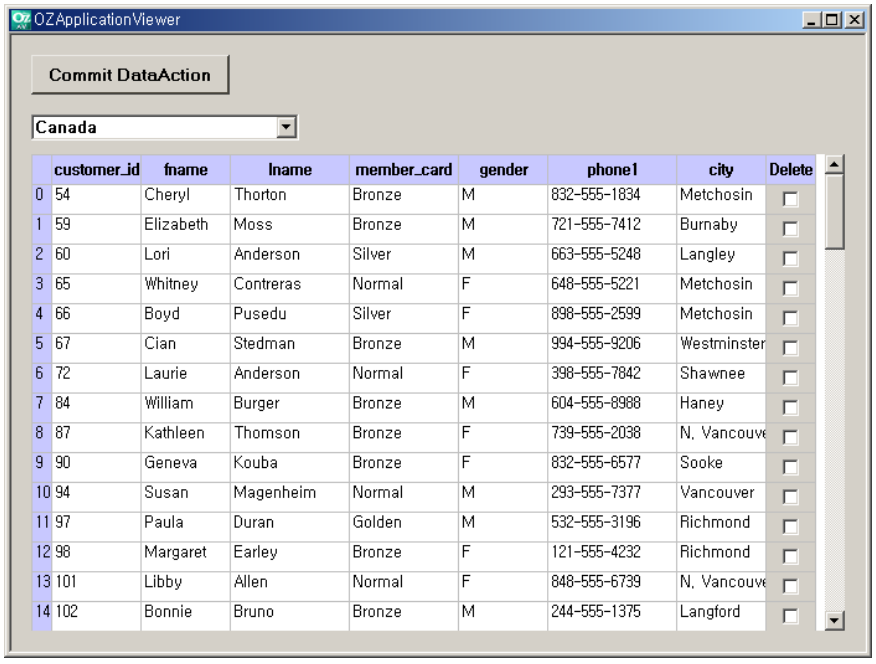
  보드에 테이블을 추가하고 테이블의 "ODIKey" 속성 값을 "class6"으로, "DataSet" 속성을 "DETAIL"로, "AllowInsert", "AllowDelete", "AllowUpdate" 속성을 "True"로, "CellSelectionMode" 속성을 "Single"로 설정합니다.
  테이블 마법사를 실행하여 "DETAIL" 데이터 셋의 필드와 "#OZDeleteFlag#"를 추가하고 추가된 모든 항목을 선택하여 "ColumnEditable" 속성을 "True"로 설정합니다.

  보드에 버튼을 추가하고 버튼의 OnClick 이벤트에 아래와 같이 스크립트를 입력합니다.

```
var Result = Table1.CommitQueuedActions();
_MessageBox(Result);
Table1.GetDataSet().RefreshDataSet();
```

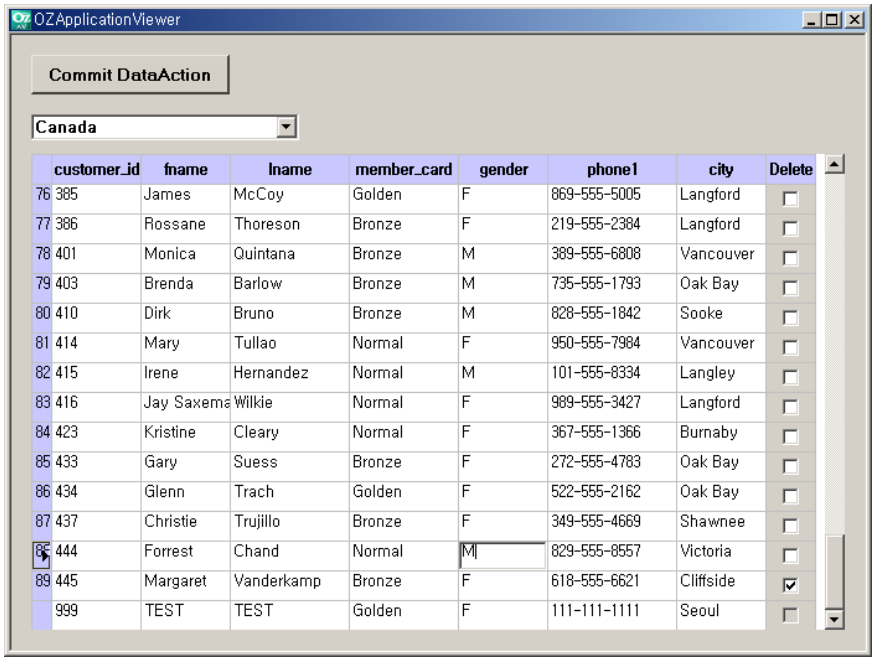
- DataAction 실행하기

  툴바의 미리보기 아이콘을 클릭하여 폼을 미리보기합니다.



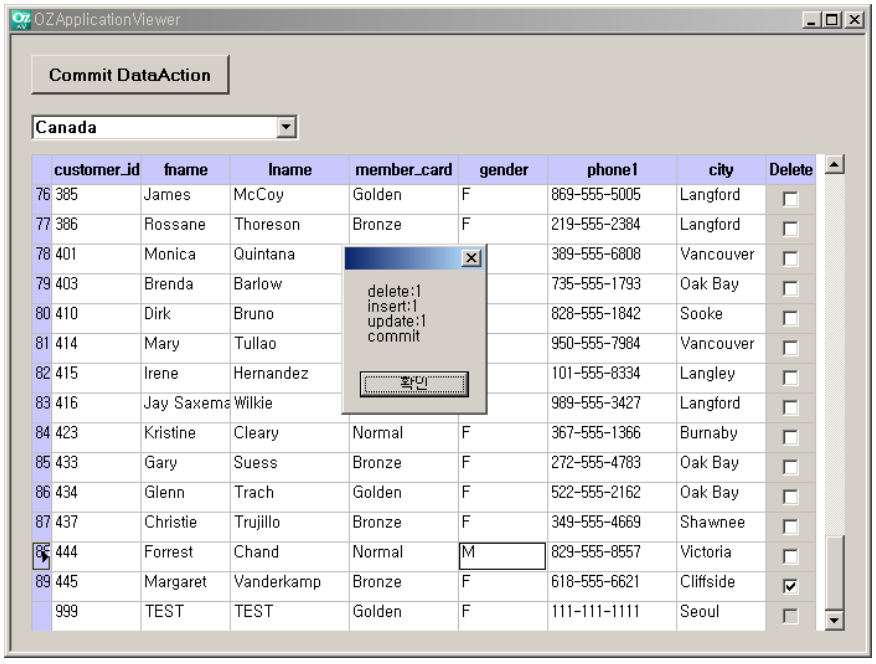
	customer_id	fname	lname	member_card	gender	phone1	city	Delete
0	54	Cheryl	Thorton	Bronze	M	832-555-1834	Metchosin	
1	59	Elizabeth	Moss	Bronze	M	721-555-7412	Burnaby	
2	60	Lori	Anderson	Silver	M	663-555-5248	Langley	
3	65	Whitney	Contreras	Normal	F	648-555-5221	Metchosin	
4	66	Boyd	Pusedu	Silver	F	898-555-2599	Metchosin	
5	67	Cian	Stedman	Bronze	M	994-555-9206	Westminster	
6	72	Laurie	Anderson	Normal	F	398-555-7842	Shawnee	
7	84	William	Burger	Bronze	M	604-555-8988	Haney	
8	87	Kathleen	Thomson	Bronze	F	739-555-2038	N. Vancouve	
9	90	Geneva	Kouba	Bronze	F	832-555-6577	Sooke	
10	94	Susan	Magenheim	Normal	M	293-555-7377	Vancouver	
11	97	Paula	Duran	Golden	M	532-555-3196	Richmond	
12	98	Margaret	Earley	Bronze	F	121-555-4232	Richmond	
13	101	Libby	Allen	Normal	F	848-555-6739	N. Vancouve	
14	102	Bonnie	Bruno	Bronze	M	244-555-1375	Langford	

마지막에 새로 추가된 행에 데이터를 입력하여 추가합니다. Delete 체크박스를 체크하여 삭제할 행을 선택합니다. 데이터를 변경할 셀을 더블 클릭하여 값을 변경합니다.



	customer_id	fname	lname	member_card	gender	phone1	city	Delete
76	385	James	McCoy	Golden	F	869-555-5005	Langford	
77	386	Rossane	Thoreson	Bronze	F	219-555-2384	Langford	
78	401	Monica	Quintana	Bronze	M	389-555-6808	Vancouver	
79	403	Brenda	Barlow	Bronze	M	735-555-1793	Oak Bay	
80	410	Dirk	Bruno	Bronze	M	828-555-1842	Sooke	
81	414	Mary	Tullao	Normal	F	950-555-7984	Vancouver	
82	415	Irene	Hernandez	Normal	M	101-555-8334	Langley	
83	416	Jay Saxema	Wilkie	Normal	F	989-555-3427	Langford	
84	423	Kristine	Cleary	Normal	F	367-555-1366	Burnaby	
85	433	Gary	Suess	Bronze	F	272-555-4783	Oak Bay	
86	434	Glenn	Trach	Golden	F	522-555-2162	Oak Bay	
87	437	Christie	Trujillo	Bronze	F	349-555-4669	Shawnee	
88	444	Forrest	Chand	Normal	M	829-555-8557	Victoria	
89	445	Margaret	Vanderkamp	Bronze	F	618-555-6621	Cliffside	☑
	999	TEST	TEST	Golden	F	111-111-1111	Seoul	

[Commit DataAction] 버튼을 클릭하면 DataAction이 실행된 후 실행된 DataAction 개수를 나타내는 메시지 박스가 표시됩니다.


OZApplicationViewer
Commit DataAction
Canada
customer_id fname lname member_card gender phone1 city Delete
76 385 James McCoy Golden F 869-555-5005 Langford
77 386 Rossane Thoreson Bronze F 219-555-2384 Langford
78 401 Monica Quintana 389-555-6808 Vancouver
79 403 Brenda Barlow 735-555-1793 Oak Bay
80 410 Dirk Bruno 828-555-1842 Sooke
81 414 Mary Tullao 950-555-7984 Vancouver
82 415 Irene Hernandez 101-555-8334 Langley
83 416 Jay Saxema Wilkie 989-555-3427 Langford
84 423 Kristine Cleary Normal F 367-555-1366 Burnaby
85 433 Gary Suess Bronze F 272-555-4783 Oak Bay
86 434 Glenn Trach Golden F 522-555-2162 Oak Bay
87 437 Christie Trujillo Bronze F 349-555-4669 Shawnee
88 444 Forrest Chand Normal M 829-555-8557 Victoria
89 445 Margaret Vanderkamp Bronze F 618-555-6621 Cliffside
999 TEST TEST Golden F 111-111-1111 Seoul
delete:1
insert:1
update:1
commit
확인

Ⅳ. 패러미터 넘겨주는 인터페이스 사용 방법

UDS에서 데이터 모듈 요청시에 패러미터를 필요로 할 경우가 있습니다. 이때 패러미터를 처리하기 위해서는 oz.uds.OZUDSParameterRef 인터페이스를 구현해야 합니다.

oz.uds.OZUDSParameterRef 인터페이스의 getParamters(), setParameters() 두 개의 메소드로 패러미터를 사용할 수 있으며, 해당 패러미터 타입은 java.util.Hashtable로 주고 받습니다.

■ **oz.uds.OZUDSParameterRef 인터페이스**

```
package oz.uds;

import java.util.Hashtable;

/**
* Description: UDS 를 통해 DataModule 요청시
* Parameter를 UDS 에서 접근 할 수 있는 interface
*/

public interface OZUDSParameterRef {
    /**
     * Hashtable 형태로 설정된 파라미터를 리턴한다.
     * @return parameter hashtable
     */
    public Hashtable getParameters();
    /**
     * 파라미터를 세팅해준다.
     * @param parameters Hashtable
     */
    public void setParameters(Hashtable parameters);
}
```

- getParameters

Prototype	public Hashtable getParameters()
Definition	Hashtable 형태의 값으로 설정된 패러미터를 리턴합니다.

- setParameters

Prototype	public void setParameters(Hashtable parameters)
Definition	Hashtable 값으로 패러미터 값을 받아 패러미터 값을 설정합니다.
Argument	*parameters* 설정할 Hashtable 패러미터 값

■ OZUDSParameterRef를 구현한 UDSParamStore Example - UDSParamStore.java

```
package uds;

import java.sql.ResultSet;
import java.util.Enumeration;
import java.util.Hashtable;
import oz.uds.*;
import oz.uds.basic.*;

public class UDSParamStore extends DefaultUserDataStore implements
OZUDSParameterRef {

   private Hashtable m_parameters;

   static {
     log = true;
   }

   public UDSParamStore()
   {
      m_parameters = new Hashtable();
   }

   public synchronized void init() {}

   public ResultSet getResultSet(String argument)
      throws OZUserDataStoreException
   {
      try {
         m_parameters.put("command", argument);
         return new UDSParamResultSet(m_parameters);
      }
      catch(Exception ex){
         ex.printStackTrace();
         throw new OZUserDataStoreException(ex.getMessage());
      }
   }

   public void freeResultSet(ResultSet resultset) {

   }

   public void close()
   {
```

```
        m_parameters.clear();
    }

    public Hashtable getParameters()
    {
        return m_parameters;
    }

    public void setParameters(Hashtable parameters)
    {
        m_parameters = new Hashtable();
        String key = "";
        Enumeration keys = parameters.keys();
        while(keys.hasMoreElements()) {
            key = (String) keys.nextElement();
            m_parameters.put(key, parameters.get(key));
        }
    }

    public synchronized Object getRawData(String command)
        throws OZUserDataStoreException {
      return null;
    }

}
```

- **DefaultResultSet를 구현한 UDSParamResultSet Example - UDSParamResultSet.java**

```
package uds;

import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.*;
import oz.uds.ResultSetMetaSkeleton;
import oz.uds.rs.DefaultResultSet;

public class UDSParamResultSet extends DefaultResultSet
{
    private Vector field1;
    private Vector field2;
    private int rowNo;

    public UDSParamResultSet(Hashtable param)
    {
        field1 = new Vector();
```

```
        field2 = new Vector();
        String f1 = "";
        Object f2 = new Object();
        for(Enumeration    enum    =    param.keys();    enum.hasMoreElements();
field2.add(f2 != null ? f2 : ""))
        {
            f1 = (String)enum.nextElement();
            f2 = param.get(f1);
            System.out.println("[KEY:" + f1 + ", VALUE:" + f2 + "]");
            field1.add(f1);
        }

        rowNo = -1;
    }

    public boolean next()
    {
        return ++rowNo < field1.size();
    }

    public ResultSetMetaData getMetaData()
        throws SQLException
    {
        return new ResultSetMetaSkeleton(new String[] {
            "KEY", "VALUE"
        }, new String[] {
            "VARCHAR", "VARCHAR"
        });
    }

    public String getString(int col)
    {
        switch(col)
        {
        case 1: // '\001'
            return (String)field1.elementAt(rowNo);

        case 2: // '\002'
            return (String)field2.elementAt(rowNo);
        }
        return "NO DATA";
    }

    public String getString(String field)
```

```
    {
        if(field == null)
            return "NO DATA";
        if(field.equalsIgnoreCase("KEY"))
            return getString(1);
        if(field.equalsIgnoreCase("VALUE"))
            return getString(2);
        else
            return "NO DATA";
    }

}
```

■ 적용 예

패러미터 값을 보고서에 출력하는 예제를 살펴보겠습니다.

➢ ODI 만들기

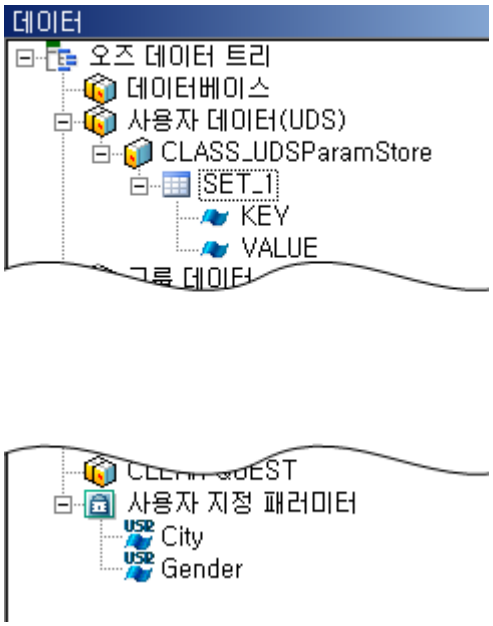
쿼리 디자이너를 실행하여 "City"와 "Gender"라는 이름으로 두 개의 패러미터를 추가하고, 패러미터의 값을 각각 "Canada"와 "F"로 설정합니다.

스토어 이름을 "CLASS_UDSParamStore"로, 클래스 파일 경로를 "UDSParamStore.class" 사용자 데이터 스토어를 추가하고, "SET_1"로 데이터 셋을 추가합니다.

데이터 셋의 쿼리문 입력 창에 아래와 같이 입력하고

```
select * from ECustomer where city = "#OZParam.City#" and gender = "#OZParam.Gender#"
```

쿼리문 실행 아이콘()을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class7.odi" 파일로 저장합니다.

- 보고서 디자인하기

  리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class7.odi" 파일을 추가합니다.

  보고서에 데이터 밴드를 추가하고 데이터 밴드의 "ODI 이름" 속성을 "class7"로, "데이터셋" 속성을 "SET_1"로 설정합니다.

  데이터 트리 창에서 "SET_1"의 "KEY"와 "VALUE" 필드를 선택한 후 데이터 밴드로 드래그&드롭하여 추가합니다.

- 미리보기

  툴바의 미리보기 아이콘을 클릭하여 보고서를 미리보기합니다.

Ⅴ. HttpRequest 접근을 위한 인터페이스 사용 방법

WAS의 서블릿 객체와 Request객체로 통해 UDS를 사용할 경우에는 서블릿 HttpRequest 객체를 처리하기 위해서 oz.uds.OZUDSServletRef 인터페이스를 구현해야 합니다.

oz.uds.OZUDSServletRef 인터페이스는 oz.uds.OZUserDataStore 클래스의 init() 메소드를 호출하기 전에 setServlet() 함수와 setHttpRequest() 함수를 전달하는 역할을 합니다.

■ oz.uds.OZUDSServletRef 인터페이스

```
package oz.uds;

import javax.servlet.*;
import javax.servlet.http.*;

public interface OZUDSServletRef
{
   public void setServlet(HttpServlet http_servlet);
   public void setHttpRequest(HttpServletRequest http_request);
}
```

- setServlet

Prototype	public void setServlet(HttpServlet http_servlet)
Definition	UDS에서 사용할 HttpServlet 값을 설정합니다.
Argument	*http_servlet* 설정할 http_servlet 값

- setHttpRequest

Prototype	public void setHttpRequest(HttpServletRequest http_request)
Definition	UDS에서 사용할 서블릿의 HttpRequest 값을 설정합니다.
Argument	*http_request* 설정할 http_request 값

■ OZUDSServletRef 를 구현한 UDSSample Example - OZUDSSample.java

```
package oz.uds;

import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Hashtable;
import java.util.*;
```

```
public class OZUDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
    HttpServlet http_servlet = null;
    HttpServletRequest http_request = null;

    // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;
    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;
    }

    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
    // methods from OZUserDataStore
    public void init() throws OZUserDataStoreException
    {
        // user my_servlet & my_request
    }

    public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
        // return a data
        Vector field1 = new Vector();
        Vector field2 = new Vector();

        // request의 값을 받아 record로 add하는 예제이다.
        if (http_request != null) {
            Enumeration enum = http_request.getParameterNames();
            while (enum.hasMoreElements()) {
                String temp = (String) enum.nextElement();
                System.out.println("name=" + temp + "   value=" +
                                   http_request.getParameter(temp));
                field1.add(temp);
                field2.add(http_request.getParameter(temp));
            }
```

```
        }else {
            System.out.println("http_request is null.");
        }
        return new OZUDSSampleResultSet(field1, field2);
    }

    public void freeResultSet(ResultSet rst){
        // release resource for specific ResultSet
    }

    public void close(){
        // release all resource
    }
}
```

■ Servelt을 위한 OZUDSSampleResultSet Example - OZUDSSampleResultSet.java

```
package oz.uds;

import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.*;
import oz.uds.ResultSetMetaSkeleton;
import oz.uds.rs.DefaultResultSet;

import javax.servlet.*;
import javax.servlet.http.*;

public class OZUDSSampleResultSet extends DefaultResultSet
{
    private Vector field1;
    private Vector field2;
    private int rowNo;

    public OZUDSSampleResultSet(Vector value, Vector value2)
    {
        field1 = value;
        field2 = value2;
        rowNo = -1;//field1.size()-2;
    }

    public boolean next()
    {
        return ++rowNo < field1.size();
    }
```

```
    public ResultSetMetaData getMetaData()
        throws SQLException
    {
        return new ResultSetMetaSkeleton(new String[] {
            "KEY", "VALUE"
        }, new String[] {
            "VARCHAR", "VARCHAR"
        });
    }

    public String getString(int col)
    {
        switch(col)
        {
        case 1: // '\001'
            return (String)field1.elementAt(rowNo);

        case 2: // '\002'
            return (String)field2.elementAt(rowNo);
        }
        return "NO DATA";
    }

    public String getString(String field)
    {
        if(field == null)
            return "NO DATA";
        if(field.equalsIgnoreCase("KEY"))
            return getString(1);
        if(field.equalsIgnoreCase("VALUE"))
            return getString(2);
        else
            return "NO DATA";
    }

}
```

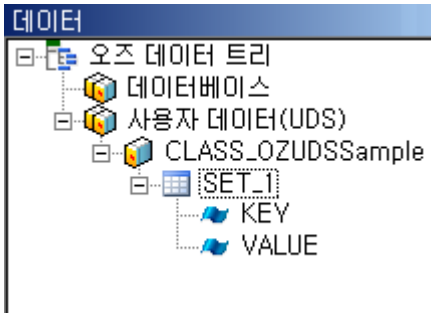
■ 적용 예

서블릿 HttpRequest 객체를 사용하는 UDS의 데이터를 보고서에 출력하는 예제를 살펴보겠습니다.

➢ ODI 만들기

쿼리 디자이너를 실행하여 스토어 이름은 "CLASS_OZUDSSample"로, 클래스 파일 경로는 "OZUDSSample.class"로 설정하여 사용자 데이터 스토어를 추가하고, "SET_1"로 데이터 셋을 추가합니다.

쿼리문 입력 창에 실행문을 입력하고 쿼리문 실행 아이콘( )을 클릭하여 필드 리스트를 생성합니다.



ODI 파일을 "class8.odi"로 저장합니다.

- 보고서 디자인하기

  리포트 디자이너를 실행하여 보고서를 새로 만든 후 "class8.odi" 파일을 추가합니다.

  보고서에 데이터 밴드를 추가하고후 데이터 밴드의 "ODI 이름" 속성을 "class8"로, "데이터셋" 속성을 "SET_1"로 설정합니다.

  데이터 트리 창에서 "SET_1"의 "KEY"와 "VALUE" 필드를 선택한 후 데이터 밴드로 드래그&드롭하여 추가합니다.

  디자인한 보고서를 "class8.ozr"로 저장한 후 ODI와 OZR을 서버 리파지토리에 업로드 합니다.

- 웹에서 보고서 미리보기

  작성된 보고서를 호출하는 jsp 페이지를 작성합니다.

  HTTP Request의 Key와 Value는 보고서를 호출하는 웹페이지의 뷰어 패러미터 중 "connection.servlet" 패러미터에 "?key1=value1&key2=value2&..."이런 식으로 입력하여야 합니다.

  예를 들어 보고서를 호출하는 jsp 페이지를 아래처럼 작성한 후

```
...
<param name="connection.servlet"
value="http://127.0.0.1:8080/oz60/server?k1=a1&k2=a2&k3=a3">
...
```

  웹 브라우저에서 jsp 페이지를 실행하여 보고서를 바인딩합니다.

Ⅵ. ResultSet 사용 방법

JDBC String/Integer 타입에 대한 매핑 정보

JDBC String TYPE	JDBC Intger Type
BIT	-7
TINYINT	-6
SMALLINT	5
INTEGER	4
BIGINT	-5
FLOAT	6
REAL	7
DOUBLE	8
NUMERIC	2
NUMBER	2
DECIMAL	3
CHAR	1
VARCHAR	12
LONGVARCHAR	-1
DATE	91
TIME	92
TIMESTAMP	93
BINARY	-2
VARBINARY	-3
LONGVARBINARY	-4
JAVA_OBJECT	2000
DISTINCT	2001
STRUCT	2002
ARRAY	2003
BLOB	2004
CLOB	2005
REF	2006
그 밖의 String 타입	12

oz.uds.rs.DefaultResultSet

기본적인 값을 제공하는 추상 클래스입니다.

DefaultResultSet을 사용할 경우 다음의 추상 멤버를 반드시 구현해야 합니다.

```
abstract public boolean next() throws SQLException;
abstract public String getString(String columnName) throws SQLException;
abstract public ResultSetMetaData getMetaData() throws SQLException;
```

oz.uds.rs.ArrResultSet

배열 형태의 데이터를 처리하는 클래스입니다.

■ Constructor Summary

- **public ArrResultSet(String[] meta, String[][] data)**
- **public ArrResultSet(String[] meta, String[] type, String[][] data)**
- **public ArrResultSet(String[] meta, int[] type, String[][] data)**

■ Constructor Detail

Prototype	//데이터 타입이 **varchar**인 경우
	public ArrResultSet(String[] meta, String[][] data)
	
	//데이터 타입이 **String**인 경우
	public ArrResultSet(String[] meta, String[] type, String[][] data)
	
	//데이터 타입이 **integer**인 경우
	public ArrResultSet(String[] meta, int[] type, String[][] data)	

Argument	*meta*	필드 이름
	type	데이터 타입
	data	2차 배열 형태의 데이터

■ Sample : ArrResultSet.java

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
```

```
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
   HttpServlet http_servlet = null;
   HttpServletRequest http_request = null;

   public void init() throws OZUserDataStoreException
   {
   }

   public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
      try {

         return getArrResultSet();
      }
      catch(Exception ex){
         ex.printStackTrace();
         throw new OZUserDataStoreException(ex.getMessage());
      }
   }

   private ResultSet getArrResultSet() throws OZUserDataStoreException {
      String[] fieldName = {"FIELD1","FILED2"};
      String[] fieldType = {"VARCHAR", "INTEGER"};

      String[][] data = {
         {"data11", "data12"},
         {"data21", "data22"},
         {"data31", "data32"}
      };

      return new oz.uds.rs.ArrResultSet(fieldName, fieldType, data);
   }

   public void close(){
      // release all resource
```

```
    }

    // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;

    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;

        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
            System.out.println("Session is null...");
        }

    }

    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```

oz.uds.rs.CSVResultSet

CSV 형태의 데이터를 처리하는 클래스입니다.

■ Constructor Summary

- public CSVResultSet(InputStream in, Properties props) throws Exception

■ Constructor Detail

Prototype	`public CSVResultSet(InputStream in, Properties props) throws Exception`

Argument		
	in	FileInputStream 또는 HttpInputStream 형태의 데이터
	props	CSV ResultSet을 만들기 위한 설정 값

※ 참고사항 : CSVResultSet에서 설정 가능한 CSV 옵션

CSV 옵션	키
필드 구분자	KEY_FIELD_DELIM
라인 구분자	KEY_LINE_DELIM
필드 타입 포함 여부	KEY_HAS_FIELDTYPE
널 마크	KEY_NULLMARK
인코딩 셋	KEY_ENCODING_CHARSET
에러 체크 여부	KEY_CHECK_ERROR

※ 참고사항 : CSVResultSet에서 설정 가능한 CSV 옵션 중 필드 구분자로 설정 가능한 값

필드 구분자	값	예약어
Tab	**□t**	FIELD_DELIMETER_TAB
Space		FIELD_DELIMETER_SPACE
Comma	**,**	FIELD_DELIMETER_COMMA
Colon	**:**	FIELD_DELIMETER_COLON
Semicolon	**;**	FIELD_DELIMETER_SEMICOLON
Null	**□0**	FIELD_DELIMETER_NULL

※ 참고사항 : props 설정 방법

```
//CSV 옵션
Properties props = new Properties();
props.put(KEY_FIELD_DELIM, ",");
props.put(KEY_LINE_DELIM, "\n");
props.put(KEY_HAS_FIELDTYPE, "true");
props.put(KEY_NULLMARK, "null");
props.put(KEY_ENCODING_CHARSET, "");
props.put(KEY_CHECK_ERROR, "true");

//필드 구분자 - 직접 설정
props.put(KEY_FIELD_DELIM, ",");

//필드 구분자 - 예약어로 설정
props.put(KEY_FIELD_DELIM, CSVResultSet.FIELD_DELIMETER_COMMA);
```

Sample : CSVResultSet.java

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
    HttpServlet http_servlet = null;
    HttpServletRequest http_request = null;

    public void init() throws OZUserDataStoreException
    {
    }

    public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
       try {
          return getCSVResultSet();
       }
       catch(Exception ex){
          ex.printStackTrace();
          throw new OZUserDataStoreException(ex.getMessage());
       }
    }

    private ResultSet getCSVResultSet() throws OZUserDataStoreException{
       try {

          String address = "http://localhost/data.csv";
          URL url = new URL(address);
          URLConnection conn = url.openConnection();

          conn.setDoOutput(true);
          conn.setDoInput(true);
          ( (HttpURLConnection) conn).setRequestMethod("POST");
```

```
            String params = url.getQuery();
            if (params != null) {
                OutputStream os = conn.getOutputStream();
                OutputStreamWriter ow = new OutputStreamWriter(os);
                ow.write(params); // caution: it is safe only with URLEncode =
true
            }

            Properties props = new Properties();
            props.put(CSVResultSet.KEY_FIELD_DELIM, "|");
            props.put(CSVResultSet.KEY_LINE_DELIM, "\n");

            props.put(CSVResultSet.KEY_HAS_FIELDTYPE, "true");
            props.put(CSVResultSet.KEY_NULLMARK, "null");
            props.put(CSVResultSet.KEY_ENCODING_CHARSET, "");

            return new oz.uds.rs.CSVResultSet(conn.getInputStream(), props);
        }
        catch (Exception ex) {
            ex.printStackTrace();
            throw new OZUserDataStoreException(ex.getMessage());
        }
    }

    public void close(){
        // release all resource
    }

    // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;

    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;

        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
```

```
            System.out.println("Session is null...");
        }
    }
    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```

oz.uds.rs.DymResultSet

동적인 형태의 데이터를 처리하는 클래스입니다.

■ Constructor Summary

- public DymResultSet(ResultSet rSet) throws SQLException

■ Constructor Detail

Prototype	public DymResultSet(ResultSet rSet) throws SQLException	
Argument	*rSet*	데이터가 동적으로 변경되는 ResultSet을 패러미터로 설정

■ Sample : DymResultSet.java

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
    HttpServlet http_servlet = null;
    HttpServletRequest http_request = null;
```

```
   public void init() throws OZUserDataStoreException
   {
   }

   public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
       try {
          return getDymResultSet();
       }
       catch(Exception ex){
          ex.printStackTrace();
          throw new OZUserDataStoreException(ex.getMessage());
       }
   }


   private ResultSet getDymResultSet() throws OZUserDataStoreException,
SQLException {

   //Param등의 타입에 따라 field가 2개인지 3개인지 동적으로 변하는 ResultSet인 경우 사용
       int type = 1;
       if (type == 1) {
          String[] fieldName = {"FIELD1","FILED2"};
          String[] fieldType = {"VARCHAR", "INTEGER"};

          String[][] data = {
             {"data11", "data12"},
             {"data21", "data22"},
             {"data31", "data32"}
          };
          return new oz.uds.rs.DymResultSet(new
oz.uds.rs.ArrResultSet(fieldName, fieldType, data));
       }else {
          String[] fieldName = {"FIELD1","FILED2", "FIELD3"};
          String[] fieldType = {"VARCHAR", "INTEGER", "VARCHAR"};

          String[][] data = {
             {"data11", "data12", "data13"},
             {"data21", "data22", "data23"},
             {"data31", "data32", "data33"}
          };
          return new oz.uds.rs.DymResultSet(new
oz.uds.rs.ArrResultSet(fieldName, fieldType, data));
       }
```

```
    }

    public void freeResultSet(ResultSet rst){
        // release resource for specific ResultSet
    }

    public void close(){
        // release all resource
    }


    // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;

    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;

        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
            System.out.println("Session is null...");
        }
    }
    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```

oz.uds.rs.VectorResultSet

Vector[] 형태의 데이터를 처리하는 클래스입니다.

Constructor Summary

- public VectorResultSet(Vector meta, Vector[] data) throws SQLException
- public VectorResultSet(Vector meta, Vector type, Vector[] data) throws SQLException

Constructor Detail

Prototype	`//데이터 타입이 varchar인 경우`
	`public VectorResultSet(Vector meta, Vector[] data) throws SQLException`
	
	`//데이터 타입이 vector인 경우`
	`public VectorResultSet(Vector meta, Vector type, Vector[] data) throws SQLException`	

Argument	*meta*	필드 이름
	type	데이터 타입
	data	vector 형태의 데이터

Sample : VectorResultSet.java

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
    HttpServlet http_servlet = null;
    HttpServletRequest http_request = null;

    public void init() throws OZUserDataStoreException
    {
    }

    public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
```

```
      try {
         return getVectorResultSet();
      }
      catch(Exception ex){
         ex.printStackTrace();
         throw new OZUserDataStoreException(ex.getMessage());
      }
   }

  private ResultSet getVectorResultSet() throws OZUserDataStoreException,
SQLException {
      Vector fieldName = new Vector();
       fieldName.add("field1");
       fieldName.add("field2");

       Vector fieldType = new Vector();
       fieldType.add("VARCHAR");
       fieldType.add("INTEGER");


       Vector[] data = new Vector[2];
       data[0] = new Vector();
       data[0].add("1");
       data[0].add("2");
       data[1] = new Vector();
       data[1].add("3");
       data[1].add("4");

      return new oz.uds.rs.VectorResultSet(fieldName, fieldType, data);
   }

   public void close(){
      // release all resource
   }

      // methods from OZUDSServletRef
   public void setServlet(HttpServlet http_servlet){
      System.out.println("http_servlet=="+http_servlet.toString());
      this.http_servlet = http_servlet;

   }
   public void setHttpRequest(HttpServletRequest http_request){
      System.out.println("http_request=="+http_request.toString());
      this.http_request = http_request;
```

```
        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
            System.out.println("Session is null...");
        }
    }

    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```

oz.uds.rs.Vector2ResultSet

Vector에서 Vector를 Wrapping한 형태의 데이터를 처리하는 클래스입니다.

■ Constructor Summary

- public Vector2ResultSet(Vector meta, Vector data) throws SQLException
- public Vector2ResultSet(Vector meta, Vector type, Vector data) throws SQLException

■ Constructor Detail

Prototype	//데이터 타입이 **varchar**인 경우
	public Vector2ResultSet(Vector meta, Vector data) throws SQLException
	
	//데이터 타입이 **vector**인 경우
	public Vector2ResultSet(Vector meta, Vector type, Vector data) throws SQLException	

Definition	vector 형태로 데이터를 설정합니다.	
Argument	*meta*	필드 이름
	type	데이터 타입
	data	vector 형태의 데이터

■ Sample : Vector2ResultSet.java

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
   HttpServlet http_servlet = null;
   HttpServletRequest http_request = null;

   public void init() throws OZUserDataStoreException
   {
   }

   public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
      try {
         return getVector2ResultSet();
      }
      catch(Exception ex){
         ex.printStackTrace();
         throw new OZUserDataStoreException(ex.getMessage());
      }
   }

    private ResultSet getVector2ResultSet() throws OZUserDataStoreException,
SQLException {
      Vector fieldName = new Vector();
       fieldName.add("field1");
       fieldName.add("field2");

      Vector fieldType = new Vector();
       fieldType.add("VARCHAR");
       fieldType.add("INTEGER");

       Vector data0 = new Vector();
```

```
        data0.add("1");
        data0.add("2");

        Vector data1 = new Vector();
        data1.add("3");
        data1.add("4");

        Vector data = new Vector();
        data.add(data0);
        data.add(data1);

        return new oz.uds.rs.Vector2ResultSet(fieldName, fieldType, data);
    }

    public void close(){
        // release all resource
    }

        // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;

    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;

        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
            System.out.println("Session is null...");
        }
    }

    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```

oz.uds.rs.NullResultSet

null 값을 리턴해주는 클래스입니다.

- **Sample : NullResultSet.java**

```
package oz.uds.rs.sample;

import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;

import oz.uds.*;
import oz.uds.rs.*;

public class UDSSample implements OZUserDataStore, OZUDSServletRef,
OZUDSParameterRef
{
   HttpServlet http_servlet = null;
   HttpServletRequest http_request = null;

   public void init() throws OZUserDataStoreException
   {
   }

   public ResultSet getResultSet(String argument) throws
OZUserDataStoreException{
      try {
        return getNullResultSet();
      }
      catch(Exception ex){
         ex.printStackTrace();
         throw new OZUserDataStoreException(ex.getMessage());
      }
   }

   private ResultSet getNullResult() throws OZUserDataStoreException,
SQLException {
      return new oz.uds.rs.NullResultSet();
   }

   public void close(){
```

```
        // release all resource
    }

        // methods from OZUDSServletRef
    public void setServlet(HttpServlet http_servlet){
        System.out.println("http_servlet=="+http_servlet.toString());
        this.http_servlet = http_servlet;

    }
    public void setHttpRequest(HttpServletRequest http_request){
        System.out.println("http_request=="+http_request.toString());
        this.http_request = http_request;

        HttpSession hs = http_request.getSession();

        if (hs != null) {
            System.out.println("Session id = "+hs.getId());
            System.out.println("Session test = "+hs.getAttribute("aaa"));
        }else{
            System.out.println("Session is null...");
        }
    }

    public void setParameters(Hashtable args) {
    }
    public Hashtable getParameters() {
        return null;
    }
}
```