목 차

Ⅰ. 시작하기 전에 2
OZ Framework 연동 개요 3

Ⅱ. POST 패러미터 4
기본 DataModule용 POST 패러미터 5
FXDataModule용 POST 패러미터와 Custom 태그 8

Ⅲ. Servlet API 함수 (for OZ Java Server) 22
OZ FX Data를 이용한 FX Data 구현 개요 23
FXDataModule 관련 함수 24
적용 예 31
예제 1 : 데이터 검색 32
예제 2 : 데이터 액션 40

Ⅳ. .Net API 함수 (for OZ .Net Server) 54
FXDataModule 관련 함수 55
적용 예 61
예제 1 : 데이터 검색 61
예제 2 : 데이터 액션 64

Ⅰ. 시작하기 전에

- OZ Framework 연동 개요

OZ Framework 연동 개요

OZR, OZA 등의 폼 파일 뿐만 아니라 데이터도 오즈 서버에서 가져오는 기존 방식과는 달리 OZ Framework을 연동하면 OZR, OZA 등의 폼 파일은 오즈 서버에서 가져오고 데이터는 오즈 서버가 아닌 일반적인 웹 애플리케이션 서버 또는 사용자가 지정한 특정 서버와 연동하여 데이터를 생성하고 생성된 데이터를 가져옵니다.

데이터를 가져올 때는 오즈 서버와 마찬가지로 데이터 셋 또는 데이터 모듈 단위로 생성하고 가져올 수 있으며, 이 때 데이터를 생성하고 가져오는 단위를 패치 단위(Fetch Unit)라고 합니다.
예를 들어 하나의 ODI 파일에 3개의 데이터셋이 있을 경우 패치 단위를 데이터셋으로 설정(fetchunit=dm_per_dataset)하면 해당 데이터셋의 데이터만 가져오고, 패치 단위를 데이터 모듈로 설정(fetchunit=dm_per_datamodule)하면 ODI 파일에 추가되어있는 모든 데이터셋의 데이터를 가져옵니다.

만일 데이터를 특정 URL로부터 데이터를 받아오는 경우에는 ODI명, 데이터셋명 등의 데이터 생성 정보를 POST 패러미터로 전달하여 URL을 호출합니다.
또한 특정 URL을 이용하여 행 삽입, 삭제, 변경 등의 DataAction을 하는 경우 DataAction 내용을 POST 패러미터로 전달하여 URL을 호출합니다. DataAction이 성공한 경우에는 "ok" 또는 "success"로 결과가 리턴됩니다. 설정 가능한 POST 패러미터는 아래 "POST 패러미터" 부분을 참조하시기 바랍니다.

OZ Framework을 이용할 경우에는 ODI의 DB 관련 정보는 사용되지 않기 때문에 사용자 데이터(UDS) 스토어를 이용하거나 오즈 애플리케이션의 경우 FXDataModule을 이용하여 구현할 수 있습니다.

Ⅱ. POST 패러미터

- 기본 DataModule용 POST 패러미터
- FXDataModule용 POST 패러미터와 Custom 태그

클라이언트로 데이터를 전송하거나 클라이언트로부터 행 삽입, 삭제, 변경 정보를 받아 처리할 경우 POST 패러미터를 이용하거나 POST 패러미터에서 정보를 추출하여 처리합니다. Post 패러미터는 ODI 등과 연동되는 기본 DataModule용 POST 패러미터와 FXDataModule에서만 사용되는 FXDataModule용 POST 패러미터로 나뉩니다.

기본 DataModule용 POST 패러미터

서버에 데이터를 조회할 때 서버로 보내지는 POST 패러미터

패러미터 이름	설명
_OZ_ODIFetchType_	패치 타입 DM_BATCH_FETCH 또는 DM_CONCURRENT_FETCH FetchUnit이 DM_PER_DATASET인 경우에 패치 타입은 항상 DM_CONCURRENT_FETCH임.
_OZ_ODIITEM_	ODI 파일명
_OZ_ODICATEGORY_	ODI 카테고리명
_OZ_DATASET_	데이터셋명 FetchUnit이 DM_PER_DATASET인 경우에만 전송됨.
_OZ_DEBUG_	Debug 모드 설정 여부 (true / false) ※ 참고사항 : XML 형태의 SDM을 생성하기 위해서는 반드시 Debug 모드로 설정해야 합니다.
MyParam	ODI 패러미터 ODI 패러미터는 같은 이름으로 전송됨

서버에 DataAction 내용을 보낼 때 서버로 보내지는 POST 패러미터

패러미터 이름	설명
MyParam	ODI 패러미터 ODI 패러미터는 같은 이름으로 전송됨

_OZ_DAC_CNT	DataAction의 개수 <INDEX>와 연관됨
<INDEX>.DATASET	DataAction의 데이터셋명
<INDEX>.TYPE	DataAction의 CUD 타입 Insert, Delete, RowUpdate 중 하나
<INDEX>.EXT	DataAction의 Extra 값
<INDEX>.SRC_CNT	DataAction의 Source 항목 개수 <INDEX2>와 연관됨
<INDEX>.SF_<INDEX2>	DataAction의 <INDEX2>번째 SourceName
<INDEX>.SV_<INDEX2>	DataAction의 <INDEX2>번째 SourceValue
<INDEX>.TRG_CNT	DataAction의 Target 항목 개수 <INDEX2>와 연관됨
<INDEX>.DF_<INDEX2>	DataAction의 <INDEX2>번째 TargetName
<INDEX>.DV_<INDEX2>	DataAction의 <INDEX2>번째 TargetValue

※ 참고사항 : 3개의 ODI 패러미터가 정의되어 있고, DataAction 3개를 Commit한 경우 전송되는 POST 패러미터의 예

```
paramA=somevalue
paramB=somevalue
paramC=somevalue
_OZ_DAC_CNT=3
0.DATASET=dataset1
0.TYPE=Insert
0.EXT=
0.SRC_CNT=2
0.SF_0=FieldName1
0.SV_0=a
0.SF_1=FieldName2
0.SV_1=b
1.DATASET=dataset1
1.TYPE=Delete
1.EXT=
1.TRG_CNT=1
1.DF_0=FieldName1
1.DV_0=a
2.DATASET=dataset1
2.TYPE=RowUpdate
2.EXT=
2.SRC_CNT=2
2.SF_0=FieldName1
```

```
2.SV_0=newvalue
2.SF_1=FieldName2
2.SV_1=oldvalue
2.TRG_CNT=2
2.DF_0=FieldName1
2.DV_0=newvalue
2.DF_1=FieldName2
2.DV_1=oldvalue
```

FXDataModule용 POST 패러미터와 Custom 태그

서버에 **DataAction** 내용을 보낼 때 서버로 보내지는 **POST** 패러미터

패러미터 이름	설명
_OZ_DAC_CNT	DataAction 개수
<DAC_INDEX>.DATASET	데이터셋 이름
<DAC_INDEX>.TYPE	DataAction 타입 (Insert, RowUpdate, Delete)
<DAC_INDEX>.EXT	DataAction의 ext 값
<DAC_INDEX>.SRC_CNT	DataAction의 Source 필드 개수
<DAC_INDEX>.SF_<SRC_INDEX>	DataAction의 Source 필드 이름
<DAC_INDEX>.SFT_<SRC_INDEX>	DataAction의 Source 필드 타입
<DAC_INDEX>.SV_<SRC_INDEX>	DataAction의 Source 필드 값 (값이 없으면 null로 처리됨)
<DAC_INDEX>.TRG_CNT	DataAction의 Target 필드 개수
<DAC_INDEX>.DF_<TRG_INDEX>	DataAction의 Target 필드 이름
<DAC_INDEX>.DFT_<TRG_INDEX>	DataAction의 Target 필드 타입
<DAC_INDEX>.DV_<TRG_INDEX>	DataAction의 Target 필드 값 (값이 없으면 null로 처리됨)

필드 타입별 상수 - 오즈 자바 서버 연동 시

Type	Object	Const	
FX_DT_BIT	Boolean	-7	"BIT"
FX_DT_TINYINT	Integer	-6	"TINYINT"
FX_DT_SMALLINT	Integer	5	"SMALLINT"
FX_DT_INTEGER	Integer	4	"INTEGER"
FX_DT_BIGINT	Long	-5	"BIGINT"

FX_DT_FLOAT	Double	6	"FLOAT"
FX_DT_REAL	Float	7	"REAL"
FX_DT_DOUBLE	Double	8	"DOUBLE"
FX_DT_NUMERIC	String	2	"NUMERIC"
FX_DT_DECIMAL	String	3	"DECIMAL"
FX_DT_CHAR	String	1	"CHAR"
FX_DT_VARCHAR	String	12	"VARCHAR"
FX_DT_LONGVARCHAR	String	-1	"LONGVARCHAR"
FX_DT_DATE	Date	91	"DATE"
FX_DT_TIME	Time	92	"TIME"
FX_DT_TIMESTAMP	Timestamp	93	"TIMESTAMP"
FX_DT_BINARY	byte[]	-2	"BINARY"
FX_DT_VARBINARY	byte[]	-3	"VARBINARY"
FX_DT_LONGVARBINARY	byte[]	-4	"LONGBINARY"
FX_DT_BLOB	byte[]	2004	"BLOB"
FX_DT_CLOB	byte[]	2005	"CLUB"

※ 참고사항 : BINARY, VARBINARY, LONGVARBINARY, BLOB, CLOB 데이터 타입은 BASE64로 저장합니다.

필드 타입별 상수(enum FX_DataTypes) - 오즈 닷넷 서버 연동 시

Type	Object
BIT	Boolean
TINYINT	Integer
SMALLINT	Integer
INTEGER	Integer
BIGINT	Long
FLOAT	Double
REAL	Float

DOUBLE	Double
NUMERIC	String
DECIMAL	String
CHAR	String
VARCHAR	String
LONGVARCHAR	String
DATE	Date
TIME	Time
TIMESTAMP	Timestamp
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
BLOB	byte[]
CLOB	byte[]

※ 참고사항 : BINARY, VARBINARY, LONGVARBINARY, BLOB, CLOB 데이터 타입은 BASE64로 저장합니다.

Custom 태그

■ **DataModule (JSP)**

Definition	데이터 모듈	
Attribute	*isCompressed*	압축 전송 여부
	isSDM	데이터 모듈을 SDM으로 생성할지 XML로 생성할지 여부
	isConcurrent	데이터 모듈을 Concurrent 모드로 전송할지 Batch 모드로 전송할지 여부
Child	Parameter, DataSetMeta, DataSet	

■ **DataModule (ASP.NET)**

Definition	데이터 모듈 ※ 참고사항 : Parameter, DataSetMeta, DataSet 태그를 자식 태그로 가집니다.	
Attribute	*runat*	태그를 서버에서 실행할지 클라이언트에서 실행할지 여부

	※ 참고사항 : 해당 기능에서는 값을 "server"로 설정하였을 경우에만 동작 가능합니다.
Child	Parameter, DataSetMeta, DataSet

■ **Send (ASP.NET)**

Definition	데이터 모듈 전송 옵션	
Attribute	*isCompress*	압축 여부(gzip)
	isSDM	데이터 모듈 형태(true : SDM, false : XML)
	isConcurrent	데이터 모듈을 Concurrent 모드로 전송할지 Batch 모드로 전송할지 여부
Child	Error	

■ **Parameter**

Definition	패러미터	
Attribute	*name*	패러미터 이름
	fieldType	패러미터 타입
Text	패러미터 값	

■ **DataSetMeta**

Definition	데이터셋 메타 정보	
Attribute	*name*	데이터 셋 이름
	masterSetName	마스터 셋 이름
Child	DataFieldMeta	

■ **DataFieldMeta**

Definition	데이터 필드 메타 정보	
Attribute	*fieldName*	필드 이름
	fieldType	필드 타입

■ **DataSet**

Definition	데이터셋	
Attribute	*name*	데이터 셋 이름
Child	Record	

■ **Record**

Definition	데이터셋의 레코드
Child	Column, DataSet

■ **Column**

Definition	데이터셋 레코드의 필드
Attribute	*Name* 필드 이름
Text	필드 값

■ **Error**

Definition	데이터 모듈 생성 실패 메시지
Text	에러 메시지

※ 주의사항

① 태그 라이브러리는 Servlet 2.3 이상 버전부터 지원합니다.

② DataAction은 태그 형태로 지원하지 않습니다.

③ DataModule과 Send 외의 태그는 JSP와 ASP.Net에서 동일하게 사용됩니다.

④ 오즈 닷넷 서버와 연동 시 DataModule 태그에 runat="server"로 반드시 입력되어야 합니다.

⑤ ASP.Net에서 aspx 페이지 작성 시 page 또는 Register 태그 외에 HTML 태그를 사용할 경우 웹 브라우저에서 XML 데이터가 정상적으로 표시되지 않을 수 있습니다.

■ **태그 라이브러리 정의**

FXData를 가져오는 태그 라이브러리는 다음과 같으며 정의된 태그 라이브러리 파일은 ozsdmtagex.tld 파일로 저장하여 JSP에서 FXData를 가져올 때 사용됩니다.
(ASP.Net에서는 별도의 태그 라이브러리 파일없이 aspx 파일에서 작성한 태그를 사용할 수 있습니다.)

- ozsdmtagex.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag
Library 1.1//EN" "http://java.sun.com/j2ee/dtds/web-
jsptaglibrary_1_1.dtd">

<!-- a tag library descriptor -->
```

```
<taglib>

 <tlibversion>1.0</tlibversion>
 <jspversion>1.2</jspversion>
 <shortname>OZ FXDataModule Tags</shortname>
 <uri>http://www.forcs.com/oz/fxsdmapi/taglib</uri>

 <info>
   FxDataModule Web Application Framework Concurrent Tags
 </info>

 <tag>
    <name>DataModule</name>
   <tagclass>oz.fxapi.custom.tagex.TAGEx_DataModule</tagclass>
   <bodycontent>JSP</bodycontent>
   <attribute>
    <name>isCompressed</name>
    <required>false</required>
    <rtexprvalue>false</rtexprvalue>
   </attribute>
   <attribute>
    <name>isSDM</name>
    <required>false</required>
    <rtexprvalue>false</rtexprvalue>
   </attribute>
   <attribute>
    <name>isConcurrent</name>
    <required>false</required>
    <rtexprvalue>false</rtexprvalue>
   </attribute>
 </tag>

 <tag>
   <name>Parameter</name>
   <tagclass>oz.fxapi.custom.tagex.TAGEx_Parameter</tagclass>
   <bodycontent>JSP</bodycontent>
   <attribute>
    <name>name</name>
    <required>true</required>
    <rtexprvalue>false</rtexprvalue>
   </attribute>
   <attribute>
    <name>fieldType</name>
```

```
   <required>false</required>
   <rtexprvalue>false</rtexprvalue>
  </attribute>
</tag>

<tag>
  <name>DataSetMeta</name>
  <tagclass>oz.fxapi.custom.tagex.TAGEx_DataSetMeta</tagclass>
  <bodycontent>JSP</bodycontent>
  <attribute>
   <name>name</name>
   <required>true</required>
   <rtexprvalue>false</rtexprvalue>
  </attribute>
  <attribute>
   <name>masterSetName</name>
   <required>false</required>
   <rtexprvalue>false</rtexprvalue>
  </attribute>
</tag>

<tag>
  <name>DataFieldMeta</name>
  <tagclass>oz.fxapi.custom.tagex.TAGEx_DataFieldMeta</tagclass>
  <bodycontent>JSP</bodycontent>
  <attribute>
   <name>name</name>
   <required>true</required>
   <rtexprvalue>false</rtexprvalue>
  </attribute>
  <attribute>
   <name>fieldType</name>
   <required>false</required>
   <rtexprvalue>false</rtexprvalue>
  </attribute>
</tag>

<tag>
  <name>DataSet</name>
  <tagclass>oz.fxapi.custom.tagex.TAGEx_DataSet</tagclass>
  <bodycontent>JSP</bodycontent>
  <attribute>
   <name>name</name>
   <required>true</required>
```

```
    <rtexprvalue>false</rtexprvalue>
   </attribute>
  </tag>

  <tag>
    <name>Record</name>
    <tagclass>oz.fxapi.custom.tagex.TAGEx_Record</tagclass>
    <bodycontent>JSP</bodycontent>
  </tag>

  <tag>
    <name>Column</name>
    <tagclass>oz.fxapi.custom.tagex.TAGEx_Column</tagclass>
    <bodycontent>JSP</bodycontent>
    <attribute>
     <name>name</name>
     <required>true</required>
     <rtexprvalue>false</rtexprvalue>
    </attribute>
  </tag>

  <tag>
    <name>Error</name>
    <tagclass>oz.fxapi.custom.tagex.TAGEx_Error</tagclass>
    <bodycontent>JSP</bodycontent>
  </tag>

</taglib>
```

Custom 태그 사용 방법

■ JSP 적용 예

본 매뉴얼에서는 Custom Tag Library를 사용사여 JSP에서 FXData를 가져오고 리포트 디자이너에서 FXData를 생성하는 방법에 대해 설명합니다.

➢ JSP 파일 만들기

Custom Tag Library를 사용하여 FXData를 가져오는 태그를 아래와 같이 입력한 후 "JSPSample.jsp"로 저장합니다.

(본 예제에서는 데이터를 concurrent 모드로 압축 전송하고 SDM 형식으로 데이터를 가져오는 방법에 대하여 설명합니다.)

- JSPSample.jsp

```
<%@ taglib uri="http://www.forcs.com/oz/fxsdmapi/taglib"
prefix="oz" %><oz:DataModule isCompressed="true" isSDM="true"
isConcurrent="true">
       <oz:Parameter                              name="param1"
fieldType="INTEGER">1</oz:Parameter>
       <oz:Parameter      name="param2"      >Param      Value
2</oz:Parameter>

       <oz:DataSetMeta name="RegionInfos" >
   <oz:DataFieldMeta name="Region" fieldType="VARCHAR" />
       </oz:DataSetMeta>

       <oz:DataSetMeta                 name="BranchOfficeInfos"
masterSetName="RegionInfos">
              <oz:DataFieldMeta            name="BranchOffice"
fieldType="VARCHAR"/>
       </oz:DataSetMeta>

       <oz:DataSetMeta                          name="CarOrders"
masterSetName="BranchOfficeInfos">
              <oz:DataFieldMeta                   name="CarID"
fieldType="VARCHAR" />
              <oz:DataFieldMeta                 name="CarName"
fieldType="VARCHAR"/>
              <oz:DataFieldMeta                name="OrderDate"
fieldType="DATE" />
              <oz:DataFieldMeta name="Price" />
       </oz:DataSetMeta>

       <oz:DataSet name="RegionInfos">
              <oz:Record>
     <oz:Column name="Region" >KOREA</oz:Column>
                     <oz:DataSet name="BranchOfficeInfos">
                  <oz:Record>
           <oz:Column name="BranchOffice" >Seoul</oz:Column>
        <oz:DataSet name="CarOrders">
                     <oz:Record>
                         <oz:Column                 name="CarID"
>H04</oz:Column>
                         <oz:Column              name="CarName"
>SONATA</oz:Column>
```

```
                            <oz:Column  name="OrderDate"  >2001-
01-12</oz:Column>
                            <oz:Column             name="Price"
>150000</oz:Column>
                        </oz:Record>

                        <oz:Record>
                            <oz:Column                name="CarID"
>K02</oz:Column>
                            <oz:Column              name="CarName"
>CREDOS</oz:Column>
                            <oz:Column  name="OrderDate"  >2003-
05-02</oz:Column>
                            <oz:Column             name="Price"
>250000</oz:Column>
                        </oz:Record>

                        <oz:Record>
                            <oz:Column                name="CarID"
>H01</oz:Column>
                            <oz:Column              name="CarName"
>MORNING</oz:Column>
                            <oz:Column  name="OrderDate"  >2005-
03-25</oz:Column>
                            <oz:Column             name="Price"
>100000</oz:Column>
                        </oz:Record>
                        <oz:Record>
                            <oz:Column                name="CarID"
>K03</oz:Column>
                            <oz:Column              name="CarName"
>CREDOS</oz:Column>
                            <oz:Column  name="OrderDate"  >2003-
01-02</oz:Column>
                            <oz:Column             name="Price"
>30000</oz:Column>
                        </oz:Record>

                    </oz:DataSet>
                </oz:Record>
            </oz:DataSet>
        </oz:Record>
    </oz:DataSet>
```

```
</oz:DataModule>
```

- 저장한 "JSPSample.jsp" 파일을 WAS에 위치시킵니다.(본 예제에서는 아래와 같은 URL에 파일을 위치시킵니다.)

```
http://127.0.0.1:8080/FXData/JSPSample.jsp
```

- 태그 라이브러리가 정의된 ozsdmtagex.tld 파일을 WAS의 WEB-INF 폴더에 위치시킵니다.(본 매뉴얼에서는 톰캣의 아래 경로에 파일을 위치시켰습니다.)

```
C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\ozsdmtagex.tld
```

- crimson.jar, log4j.jar, ozsdmapi15.jar 파일을 WAS의 WEB-INF\lib 폴더에 위치시킵니다.(본 매뉴얼에서는 톰캣의 아래 경로에 파일을 위치시켰습니다.)

```
C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\lib\
```

- web.xml 파일을 일반 텍스트 편집기에서 열어 아래와 같이 태그 라이브러리 파일 관련 내용을 추가합니다.

```
<taglib>
    <taglib-uri>http://www.forcs.com/oz/fxsdmapi/taglib</taglib-uri>
    <taglib-location>/WEB-INF/ozsdmtagex.tld</taglib-location>
</taglib>
```

- FX 데이터, 보고서 파일 만들기

  오즈 리포트 디자이너를 실행하여 데이터 정보 창에서 데이터 트리의 팝업 메뉴에서 [FX 데이터 추가] 메뉴를 클릭합니다. FX 데이터 이름을 입력하고 프레임웍 URL을 "http://127.0.0.1:8080/FXData/JSPSample.jsp" 형태로 입력한 후 기타 옵션을 설정하고 [확인] 버튼을 클릭합니다.

  생성한 FX 데이터를 테이블에 연결시키고 마스터-디테일 셋 구조로 보고서를 디자인한 후 미리보기합니다.

■ ASP.Net 적용 예

본 매뉴얼에서는 User-define Control을 사용하여 aspx에서 FXData를 가져오고 리포트 디자이너에서 FXData를 생성하는 방법에 대해 설명합니다.

➢ aspx 파일 만들기

Custom Tag Library를 사용사여 FXData를 가져오는 태그를 아래와 같이 입력한 후 "ASPSample.aspx"로 저장합니다.

(본 예제에서는 데이터를 concurrent 모드로 압축 전송하고 SDM 형식으로 데이터를 가져오는 방법에 대하여 설명합니다.)

- ASPSample.aspx

```
<%@ Register TagPrefix="oz" Namespace="oz.fxapi.custom.tagex"
Assembly="OZSDMAPI" %>
<oz:DataModule runat="server">

      <oz:Send       isCompressed="true"       isSDM="true"
isConcurrent="true"/>

<oz:Parameter                                          name="NAME1"
fieldType="TYPE">VALUE1</oz:Parameter>
<oz:Parameter                                          name="NAME2"
fieldType="TYPE">VALUE2</oz:Parameter>

      <oz:DataSetMeta name="RegionInfos">
   <oz:DataFieldMeta fieldName="Region" fieldType="VARCHAR" />
      </oz:DataSetMeta>

      <oz:DataSetMeta                name="BranchOfficeInfos"
masterSetName="RegionInfos">
            <oz:DataFieldMeta       fieldName="BranchOffice"
fieldType="VARCHAR"/>
      </oz:DataSetMeta>

      <oz:DataSetMeta                          name="CarOrders"
masterSetName="BranchOfficeInfos">
            <oz:DataFieldMeta              fieldName="CarID"
fieldType="VARCHAR" />
            <oz:DataFieldMeta            fieldName="CarName"
fieldType="VARCHAR"/>
            <oz:DataFieldMeta          fieldName="OrderDate"
fieldType="DATE" />
            <oz:DataFieldMeta fieldName="Price" />
      </oz:DataSetMeta>

      <oz:DataSet name="RegionInfos">
            <oz:Record>
     <oz:Column name="Region">KOREA</oz:Column>
                  <oz:DataSet name="BranchOfficeInfos">
```

```
                <oz:Record>
            <oz:Column name="BranchOffice">Seoul</oz:Column>
        <oz:DataSet name="CarOrders">
                    <oz:Record>
                        <oz:Column
name="CarID">H04</oz:Column>
                        <oz:Column
name="CarName">SONATA</oz:Column>
                            <oz:Column   name="OrderDate">2001-
01-12</oz:Column>
                            <oz:Column
name="Price">150000</oz:Column>
                    </oz:Record>

                    <oz:Record>
                        <oz:Column
name="CarID">K02</oz:Column>
                        <oz:Column
name="CarName">CREDOS</oz:Column>
                            <oz:Column   name="OrderDate">2003-
05-02</oz:Column>
                            <oz:Column
name="Price">250000</oz:Column>
                    </oz:Record>

                    <oz:Record>
                        <oz:Column
name="CarID">H01</oz:Column>
                        <oz:Column
name="CarName">MORNING</oz:Column>
                            <oz:Column   name="OrderDate">2005-
03-25</oz:Column>
                            <oz:Column
name="Price">100000</oz:Column>
                    </oz:Record>

                    <oz:Record>
                        <oz:Column
name="CarID">K03</oz:Column>
                        <oz:Column
name="CarName">CREDOS</oz:Column>
                            <oz:Column   name="OrderDate">2003-
01-02</oz:Column>
                            <oz:Column
```

```
name="Price">30000</oz:Column>
                        </oz:Record>

                      </oz:DataSet>
                    </oz:Record>
                  </oz:DataSet>
                </oz:Record>
        </oz:DataSet>
</oz:DataModule>
```

- 저장한 "ASPSample.aspx" 파일을 닷넷 서버가 설치된 폴더에 위치시킵니다.
  (본 예제에서는 아래와 같은 URL에 파일을 위치시킵니다.)

  http://127.0.0.1/FXData/ASPSample.aspx

- OZSDMAPI.dll 파일을 닷넷 서버가 설치된 bin 폴더에 위치시킵니다.

- FX 데이터, 보고서 파일 만들기
  오즈 리포트 디자이너를 실행하여 데이터 정보 창에서 데이터 트리의 팝업 메뉴에서 [FX 데이터 추가] 메뉴를 클릭합니다. FX 데이터 이름을 입력하고 프레임웍 URL을 "http://127.0.0.1/FXData/ASPSample.aspx" 형태로 입력한 후 기타 옵션을 설정하고 [확인] 버튼을 클릭합니다.
  생성한 FX 데이터를 테이블에 연결시키고 마스터-디테일 셋 구조로 보고서를 디자인한 후 미리보기합니다.

Ⅲ. Servlet API 함수 (for OZ Java Server)

- OZ FX Data를 이용한 FX Data 구현 개요
- FXDataModule 관련 함수
- 적용 예

OZ FX Data를 이용한 FX Data 구현 개요

구현 개요

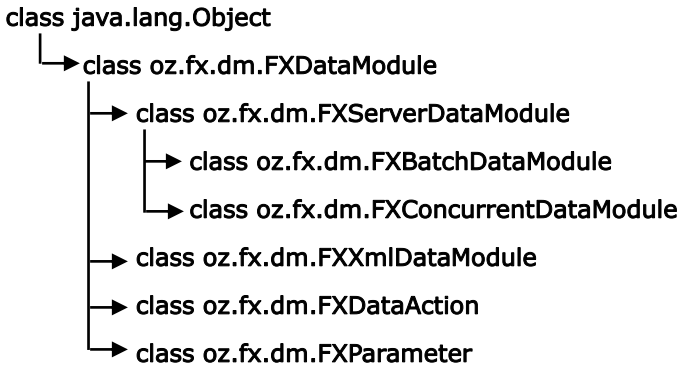
OZ SDM API에서 제공하는 FX Data는 서버 프레임웍 URL로부터 데이터를 가져와 오즈 뷰어에 SDM 또는 XML 포맷으로 전달하여 보고서를 바인딩할 경우에 사용되는 인터페이스입니다.

구현 시 참고사항

OZ FX Data는 별도의 제품을 설치하는 것이 아니라 서블릿의 JSP 파일 등에 API를 구현하여 사용할 수 있습니다.

FXDataModule 관련 함수

OZ FX Data의 oz.fx.dm 패키지 구조는 아래와 같으며 각 객체에 대한 설명과 함수는 다음과 같습니다.



oz.fx.dm.FXDataModuleFactory

■ 개요

데이터 모듈을 생성하는 객체입니다.

■ 함수

- createBatchDataModule

Prototype	`public static FXDataModule createBatchDataModule()`
Definition	전체 데이터에 대한 데이터 모듈 생성 작업을 완료한 후 전송되는 SDM 데이터 모듈을 생성하고, SDM 데이터 모듈 객체를 가져옵니다.

- createBatchXmlDataModule

Prototype	`public static FXDataModule createBatchXmlDataModule()`
Definition	전체 데이터에 대한 데이터 모듈 생성 작업을 완료한 후 전송되는 형태의 XML 데이터 모듈을 생성하고, XML 데이터 모듈 객체를 가져옵니다.

- createConcurrentDataModule

Prototype	`public static FXDataModule createConcurrentDataModule()`
Definition	데이터 모듈 생성 작업과 전송 작업을 동시에 수행하는 형태의 SDM 데이터 모듈을 생성하고, SDM 데이터 모듈 객체를 가져옵니다.

- createConcurrentXmlDataModule

Prototype	`public static FXDataModule createConcurrentXmlDataModule()`
Definition	데이터 모듈 생성 작업과 전송 작업을 동시에 수행하는 형태의 XML 데이터 모듈을 생성하고, XML 데이터 모듈 객체를 가져옵니다.

oz.fx.dm.FXDataModule

■ 개요

데이터 모듈을 제어하는 객체입니다.

■ 함수

- addDataSet

Prototype	`public void addDataSet(String dataSetName)`
	
	`public void addDataSet(String dataSetName, String masterSetName)`

Definition	데이터 셋을 추가합니다. ※ 참고사항 : 디테일 셋을 설정할 경우 마스터 셋 이름을 설정합니다.
Argument	*dataSetName* 데이터 셋 이름
	masterSetName 마스터 셋 이름

- addField

Prototype	`public void addField(String dataSetName, String fieldName, integer fieldType)`
Definition	데이터 셋에 필드를 추가합니다.
Argument	*dataSetName* 데이터 셋 이름
	fieldName 필드 이름

	fieldType 필드 타입 (java.sql.Types 참조)

- completeRow

Prototype	`public void completeRow()`
Definition	하나의 Row에 대한 데이터 값 설정을 완료합니다.

- endSet

Prototype	`public void endSet(String dataSetName)`
Definition	데이터 셋 바인딩을 종료합니다.
Argument	*dataSetName* 데이터 셋 이름

- endBinding

Prototype	`public void endBinding()`
Definition	데이터 모듈 바인딩을 종료합니다.

- finish

Prototype	`public void finish()`
Definition	모든 작업을 종료하며, 데이터 전송 타입이 batch 타입일 경우 데이터 전송 작업이 시작됩니다.

- getDataAction

Prototype	`public FXDataAction[] getDataAction(HttpServletRequest request) throws Exception`
Definition	Request로부터 패러미터를 파싱하여 FXDataAction 객체를 리턴합니다.
Argument	*request* Request 객체

- initialize

Prototype	`public void initialize(java.io.OutputStream out)`
	
	`public void initialize(boolean isCompressed, boolean isEncrypted, java.io.OutputStream out)`
	
	`public void initialize(boolean isCompressed, java.io.OutputStream out, integer bufferSize)`

Definition	데이터 모듈을 기록할 정보를 초기화하며 데이터 압축 여부와 버퍼 크기를 설정할 수 있습니다. FXDataModule의 type=sdm, concurrent=false, compressed=true일 경우 암호화됩니다. ※ 참고사항 : 데이터 바인딩 시작 전(startBinding() 함수 호출 전)에 호출되어야 합니다.	
Argument	*isCompressed*	압축 전송 여부 (기본 값 : false)
	isEncrypted	암호화 여부 (암호화 시 128bits DES 방식 사용)
	out	출력 스트림
	buffersize	버퍼 크기, 단위 : byte (기본 값 : 4096)

- startBinding

Prototype	`public void startBinding()`
Definition	메타 정보가 모두 등록되었을 때 데이터 모듈 바인딩을 시작하며 메타 정보의 무결성을 검증합니다.

- startSet

Prototype	`public void startSet(String dataSetName)`	
Definition	데이터 셋 바인딩을 시작합니다. ※ 참고사항 : startSet(String) 메서드가 호출된 적이 있는 경우 해당 데이터 셋의 디테일 셋 형태로 데이터가 기록됩니다.	
Argument	*dataSetName*	데이터 셋 이름

- setValue

Prototype	`public void setValue(integer oneBasedIndex, Object value)`	
Definition	현재 바인딩 중인 데이터 셋에 값을 설정합니다.	
Argument	*oneBasedIndex*	1 부터 시작되는 인덱스
	value	값

- sendError

Prototype	`public void sendError(String errorMessage)`	
Definition	에러 메시지를 전송합니다.	
Argument	*errorMessage*	에러 메시지

oz.fx.dm.FXDataAction

■ 개요

데이터 액션을 제어하는 객체입니다.

■ 함수

– FXDataAction

Prototype	`public FXDataAction(String name, String actionType)`
	
	`public FXDataAction(String name, String actionType, String ext)`

Definition	DataAcion을 생성합니다.
Argument	*name* 데이터셋 이름
	actionType DataAction 타입 (insert, rowupdate, delete)
	ext ext 값

– getDataSetName

Prototype	`public String getDataSetName()`
Definition	데이터셋 이름을 리턴합니다.

– getActionType

Prototype	`public String getActionType()`
Definition	DataAction 타입을 리턴합니다.
Argument	DataAction 타입 (insert, rowupdate, delete)

– getExt

Prototype	`public String getExt()`
Definition	ext 값을 리턴합니다.

– setExt

Prototype	`public void setExt(String ext)`
Definition	ext 값을 설정합니다.
Argument	*ext* ext 값

- getSourceFields

Prototype	`public FXParameter[] getSourceFields()`
Definition	Source 필드 정보를 리턴합니다.

- setSourceFields

Prototype	`public void setSourceFields(FXParameter[] fields)`
Definition	Source 필드 정보을 설정합니다.
Argument	*fields* Source 필드 정보

- getTargetFields

Prototype	`public FXParameter[] getTargetFields()`
Definition	Target 필드 정보을 리턴합니다.

- setTargetFields

Prototype	`public void setTargetFields(FXParameter[] fields)`
Definition	Target 필드 정보을 설정합니다.
Argument	*fields* Target 필드 정보

oz.fx.dm. FXParameter

■ 개요

데이터 액션 시 패러미터를 제어하는 객체입니다.

■ 함수

- FXDataAction

Prototype	`public FXParameter(String name, int type, Object value)` `public FXParameter(String name, Object value)`
Definition	FXDataModule의 패러미터를 생성합니다.
Argument	*name* 패러미터 이름

type	패러미터 데이터 타입 (sqlType과 동일)
value	패러미터 데이터 값

- getValue

Prototype	`public Object getValue()`
Definition	패러미터 값을 리턴합니다.

- isNull

Prototype	`public boolean isNull()`
Definition	null 값 허용 여부를 리턴합니다.

- setIsNull

Prototype	`public void setIsNull(boolean isNull)`
Definition	null 값 허용 여부를 설정합니다.
Argument	*isNull* null 허용 여부

- getName

Prototype	`public String getName()`
Definition	패러미터 이름을 리턴합니다.

- getType

Prototype	`public int getType()`
Definition	패러미터 데이터 타입을 리턴합니다. (sqlType과 동일)

적용 예

다음은 Servlet API 함수를 사용하는 예에 대해 설명합니다.

예제 1 : 데이터 검색

다음은 Servlet API를 이용하여 FXDataModuler과 연동하는 클래스 파일을 생성하고 오즈 애플리케이션과 오즈 리포트에 생성된 클래스 파일을 연동하여 데이터를 표시하는 방법을 예를 들어 설명합니다.

■ FXDataModule 등록하기

해당 예제의 데이터를 전달하는 방법은 압축 XML 형태이고 Servlet API를 이용하여 데이터를 검색하기 위해서는 먼저 ozsdmapi.jar, crimson.jar 파일이 WAS에 등록하여야 합니다.

- ozsdmapi.jar과 crimson.jar 파일을 WAS에 등록하기
  Servlet API 연동 모듈인 ozsdmapi.jar, crimson.jar 파일을 WAS의 라이브러리에 등록합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\lib 폴더에 ozsdmapi.jar, crimson.jar 파일을 복사하였습니다.

- FXDataModule 연동 클래스 파일 만들기
  FXDataModule과 연동하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 DataModuleSampleServlet.class 파일로 만들었습니다.

```
package sample;

import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oz.fx.dm.*;

public class DataModuleSampleServlet extends HttpServlet {
    // post
    public void doPost(final HttpServletRequest request, final HttpServletResponse response) throws
        ServletException, IOException {
        process(request, response);
    }
```

```
// get
public void doGet(final HttpServletRequest request, final HttpServletResponse response) throws
      ServletException, IOException {
      process(request, response);
}

// real process
private void process(final HttpServletRequest request,
                           final HttpServletResponse response) throws ServletException,
      IOException {

      // request type = > xml, sdm, reqdac, xmldac
      final String type = request.getParameter("type");

      // is compressed??
      final boolean isCompressed = "true".equalsIgnoreCase(request.getParameter(
            "compressed"));

      // type is null
      if (null == type || 0 == type.length()) {
            throw new ServletException("Operation type not specified.");
      }

      // dispatch of type
      try {
            if("sdm".equalsIgnoreCase(type)) {
                  processDataModuleEx(request, response, true, isCompressed);
            }
            else if("xml".equalsIgnoreCase(type)) {
                  processDataModuleEx(request, response, false, isCompressed);
            }
            else {
                  throw new ServletException("Unknown operation code; " + type);
            }
      }
      catch (final Exception e) {
            e.printStackTrace();
            throw new ServletException(e.getMessage());
      }
}

private void processDataModuleEx(final HttpServletRequest httpRequest,
                                       final HttpServletResponse httpResponse,
```

```
                                    final boolean isSDM,
                                    final boolean isCompressed) throws Exception {

final boolean concurrent = "true".equalsIgnoreCase(httpRequest.getParameter(
      "concurrent"));

final String rowCount = httpRequest.getParameter("addRowCount");
int additionalRowCount = 0;

if(null != rowCount && 0 != rowCount.length())
   additionalRowCount = Integer.parseInt(rowCount);


FXDataModule dataModule = null;
if (isSDM) {
   if (concurrent)
      dataModule = FXDataModuleFactory.createConcurrentDataModule();
   else
      dataModule = FXDataModuleFactory.createBatchDataModule();
}
else {
   dataModule = FXDataModuleFactory.createBatchXmlDataModule();
}

dataModule.initialize(isCompressed, encrypt, httpResponse.getOutputStream());

// Access does not support multiple ResultSet
Connection con = null;
Connection con2 = null;

try {
      // connect db
      final String url = "jdbc:odbc:ozcar";
      final java.util.Properties prop = new java.util.Properties();

      final Driver driver = (Driver) Class.forName(
            "sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
      con = driver.connect(url, prop);
      con2 = driver.connect(url, prop);
}
catch (final Exception e) {
      dataModule.sendError(e.getMessage());
      oz.util.OZSQL.close(con);
      oz.util.OZSQL.close(con2);
```

```
        return;
}

// set content type
   httpResponse .setContentType(isSDM || isCompressed ?
                                "application/octet-stream" : "text/xml");

dataModule.addParameter("param1", Types.VARCHAR, "This is para□"meter 1");
dataModule.addParameter("param2", Types.VARCHAR, "This is parameter 2");
dataModule.addParameter("param3", Types.VARCHAR, "This is parameter 3");

// create meta data
final String masterDataSetName = "Orders";
dataModule.addDataSet(masterDataSetName);
dataModule.addField(masterDataSetName, "Region", Types.VARCHAR);

final String detailDataSetName = "OrderDetail";
dataModule.addDataSet(detailDataSetName, masterDataSetName);
dataModule.addField(detailDataSetName, "BranchOffice", Types.VARCHAR);
dataModule.addField(detailDataSetName, "CarID", Types.VARCHAR);
dataModule.addField(detailDataSetName, "Quantity", Types.INTEGER);

final Statement stmt = con.createStatement();
final Statement stmt2 = con2.createStatement();
String id;

ResultSet master = null;

try {
dataModule.startBinding();

// execute master query
master = stmt.executeQuery("Select Region from Orders");

      dataModule.startSet(masterDataSetName);
      while (master.next()) {
            id = master.getString(1);
            dataModule.setValue(1, id);
            dataModule.completeRow();

                  dataModule.startSet(detailDataSetName);
            // execute detail query
            if (!oz.util.OZString.isNullOrEmpty(id)) {
                  final ResultSet detail = stmt2
```

```
                    .executeQuery(
                    "SELECT Orders.BranchOffice, Orders.CarID, Orders.Quantity,
Orders.TotalAmount FROM Orders where Region = '" +
                    id + "'");

                try {
                    while (detail.next()) {
                        for (int i = 1; i <= 3; i++)
                            dataModule.setValue(i, detail.getString(i));

                        dataModule.completeRow();

                        for(int i = 0; i < additionalRowCount; i++) {
                            dataModule.completeRow();
                        }
                    }
                }
                finally {
                    detail.close();
                }
            }
            dataModule.endSet(detailDataSetName);
        }
        dataModule.endSet(masterDataSetName);
    }
    catch (final Exception e) {
        dataModule.sendError(e.getMessage());
        return;
        //throw e;
    }
    finally {
        // close
        master.close();
        stmt.close();
        stmt2.close();
        oz.util.OZSQL.close(con);
        oz.util.OZSQL.close(con2);
    }

    dataModule.endBinding();
    dataModule.finish();
    System.out.println("Success to create data module");
  }
}
```

➢ 클래스 파일 등록하기

DataModuleSampleServlet.class 파일을 WAS에 등록합니다.

- DataModuleSampleServlet.class 파일을 웹 서버의 클래스 폴더에 복사합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\classes 폴더에 sample 폴더를 새로 만든 후 클래스 파일을 복사하였습니다.
- Test.class 파일을 WAS에 매핑합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF의 web.xml 파일을 편집기로 열어 아래와 같이 편집하여 복사한 클래스 파일을 매핑하였습니다.

```
...
<!-- JSPC servlet mappings start -->
...
<servlet>
    <servlet-name>sample.DataModuleSampleServlet</servlet-name>
    <servlet-class>sample.DataModuleSampleServlet</servlet-class>
  </servlet>

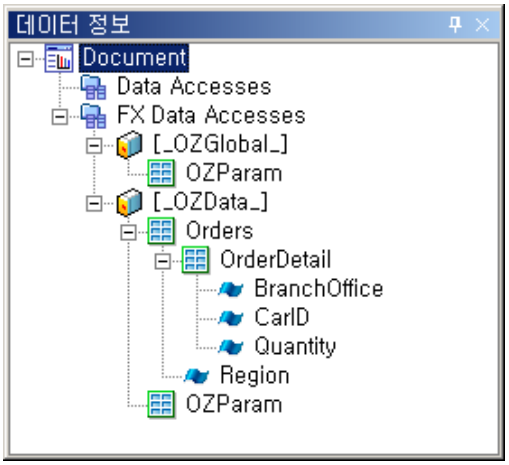
  <servlet-mapping>
          <servlet-name>sample.DataModuleSampleServlet</servlet-name>
          <url-pattern>/sample.DataModuleSampleServlet</url-pattern>
  </servlet-mapping>...
<!-- JSPC servlet mappings end -->
```

■ **오즈 애플리케이션과 연동하기**

다음은 WAS에 등록한 클래스 파일을 이용하여 오즈 애플리케이션 FX Data Accesses의 "_OZData_" 데이터 모듈에서 마스터 디테일 셋을 추가하고 Table에 데이터를 표시하는 방법을 예를 들어 설명하겠습니다.

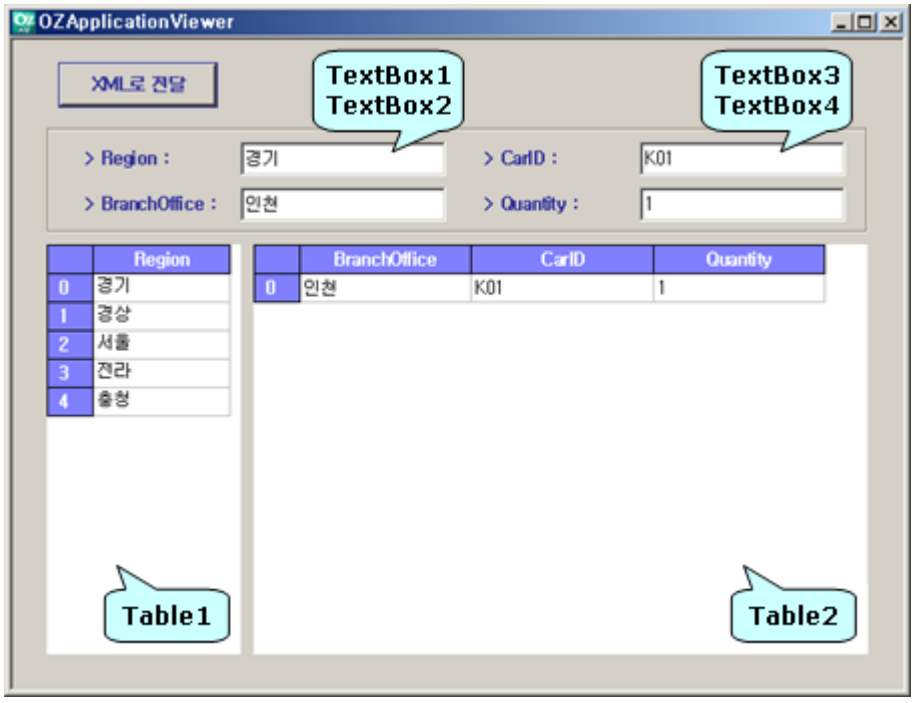
➢ FXDataSet 추가하기

- FX Data Accesses의 "_OZData_"을 마우스 오른쪽 버튼으로 선택하여 나타나는 팝업 메뉴에서 [새 FX DataSet] 메뉴를 클릭하여 아래 그림과 같은 마스터-디테일 구조의 데이터셋을 생성합니다.



➢ 컴포넌트 디자인

Board에 Label, TextBox, Button, Table을 아래 그림과 같이 추가한 후 컴포넌트의 속성 값을 설정합니다.



- TextBox1

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Orders
Field	Region	**-**	-

- TextBox2

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	OrderDetail
Field	BranchOffice	**-**	-

- TextBox3

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	OrderDetail
Field	CarID	**-**	-

- TextBox4

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	OrderDetail
Field	Quantity	**-**	-

- Table1

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Orders
FireRowCursorChange	True	**-**	-

- Table2

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	OrderDetail

➢ 스크립트 입력

정보를 XML로 전달하기 위한 스크립트를 입력합니다.

- [XML로 전달] 버튼의 OnClick 이벤트

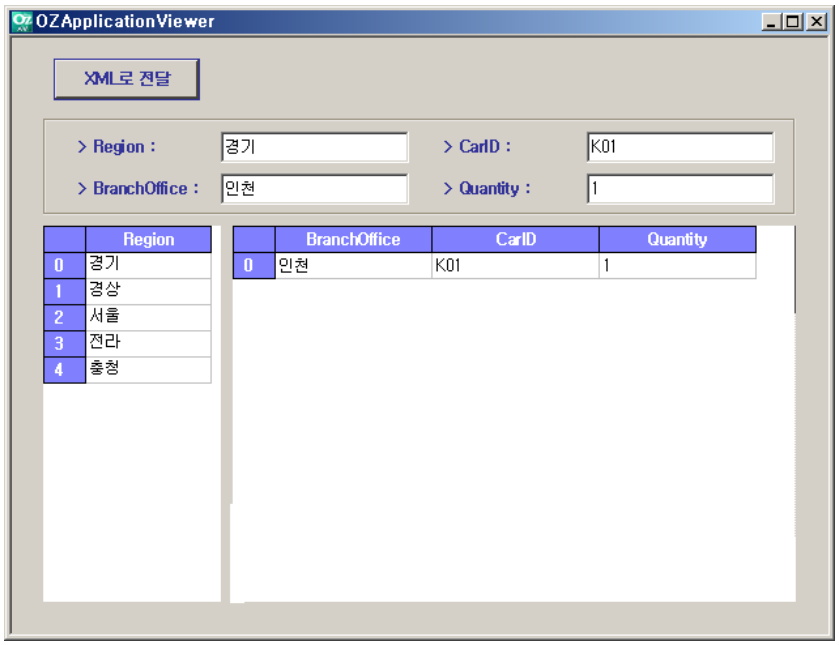
```
//XML 접속 객체 생성
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

//데이터 전송
xmlhttp.Open("POST",          "http://127.0.0.1:8080/sample.ataModuleSampleServlet?type=xml",
false);
xmlhttp.Send("");
```

```
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
        _MessageBox(dm.FXErrorMessage);
}
```

➢ 미리보기

[File] 메뉴의 [Preview]를 클릭하거나 툴바의 미리보기 아이콘( )을 클릭하여 미리 보기합니다.



■ 오즈 리포트와 연동하기

다음은 WAS에 등록한 클래스 파일을 이용하여 마스터 디테일 관계의 FX 데이터를 추가하고 해당 데이터를 테이블에 표시하는 방법에 대하여 설명하겠습니다.

※ 주의사항 : Servlet API를 이용하여 보고서에 데이터를 표시할 경우에는 오즈 리포트 디자이너에서는 미리보기할 수 없으며, 서버 리파지토리에 보고서를 등록한 후 웹 페이지를 통해서만 미리보기할 수 있습니다.

오즈 리포트 디자이너를 실행하여 데이터 정보 창에서 데이터 트리의 팝업 메뉴에서 [FX 데이터 추가] 메뉴를 클릭합니다. FX 데이터 이름을 입력하고 프레임웍 URL을 "http://127.0.0.1:8080/sample.DataModuleSampleServlet?type=xml" 형태로 입력한 후 기타 옵션을 설정하고 [확인] 버튼을 클릭합니다.

생성한 FX 데이터를 테이블에 연결시키고 마스터-디테일 셋 구조로 보고서를 디자인한 후 미리보기합니다.

예제 2 : 데이터 액션

다음은 Servlet API를 이용하여 오즈 애플리케이션 FX Data Accesses의 "_OZData_" 데이터 모듈에서 마스터 디테일 셋을 추가하고 Table에 데이터를 표시하고 표시된 데이터를 삽입, 삭제, 변경하는 예를 들어 설명하겠습니다.

■ FXDataModule 등록하기

해당 예제의 데이터를 전달하는 방법은 압축 XML 형태이고 Servlet API를 이용하여 데이터를 검색하기 위해서는 먼저 ozsdmapi.jar, crimson.jar 파일이 WAS에 등록하여야 합니다.

- ozsdmapi.jar과 crimson.jar 파일을 WAS에 등록하기
  Servlet API 연동 모듈인 ozsdmapi.jar, crimson.jar 파일을 WAS의 라이브러리에 등록합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\lib 폴더에 ozsdmapi.jar, crimson.jar 파일을 복사하였습니다.

- FXDataModule 연동 클래스 파일 만들기
  FXDataModule과 연동하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 DataAction.class 파일로 만들었습니다.

```
package sample;

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

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

import oz.fx.dm.*;
import oz.util.*;

public class DataActionEx extends HttpServlet {
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
      ServletException, IOException {
      process(request, response);
}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws
      ServletException, IOException {
      process(request, response);
}

private void process(HttpServletRequest request,
                           HttpServletResponse response) throws ServletException,
      IOException {
      String type = request.getParameter("type");
      boolean isCompressed = "true".equalsIgnoreCase(request.getParameter("compressed"));

      if (OZString.isNullOrEmpty(type))
            throw new ServletException("Operation type not specified.");

      try {
            if ("sdm".equalsIgnoreCase(type))
                  processSDM(request, response, true, isCompressed);
            else if ("xml".equalsIgnoreCase(type))
                  processSDM(request, response, false, isCompressed);
            else if ("dac".equalsIgnoreCase(type))
                  processDAC(request, response);
            else
                  throw new ServletException("Unknown operation code; " + type);
      }
      catch (Exception e) {
            e.printStackTrace();
            throw new ServletException(e.getMessage());
      }
}

/** Generate data module which has master-detail relation
 * @param request
 * @param response
 * @param isSDM
 * @throws Exception
 */
private void processSDM(HttpServletRequest request,
                              HttpServletResponse response, boolean isSDM,
                              boolean isCompressed) throws Exception {
```

```
    boolean concurrent = "true".equalsIgnoreCase(request.getParameter(
            "concurrent"));

    FXDataModule dataModule = null;
    if (isSDM) {
            if (concurrent)
                    dataModule = FXDataModuleFactory.createConcurrentDataModule();
            else
                    dataModule = FXDataModuleFactory.createBatchDataModule();
    }
    else {
            dataModule = FXDataModuleFactory.createBatchXmlDataModule();
    }

dataModule.initialize(isCompressed, encrypt, response.getOutputStream());

    // Access does not support multiple ResultSet
    Connection con = null;
    Connection con2 = null;

    try {
            String url = "jdbc:odbc:ozcar";
            Properties prop = new Properties();

            Driver driver = (Driver) Class.forName(
                    "sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
            con = driver.connect(url, prop);
            con2 = driver.connect(url, prop);
    }
    catch (Exception e) {
            dataModule.sendError(e.getMessage());
            OZSQL.close(con);
            OZSQL.close(con2);
            throw new ServletException(e.getMessage());
    }

    // set content type
    response.setContentType(isSDM || isCompressed ?
                                    "application/octet-stream" : "text/xml");


    // create meta data
```

```
String masterDataSetName = "IDs";
dataModule.addDataSet(masterDataSetName);
dataModule.addField(masterDataSetName, "CarID", Types.VARCHAR);

String detailDataSetName = "Informations";
dataModule.addDataSet(detailDataSetName, masterDataSetName);
dataModule.addField(detailDataSetName, "Maker", Types.VARCHAR);
dataModule.addField(detailDataSetName, "CarName", Types.VARCHAR);
dataModule.addField(detailDataSetName, "ECarName", Types.VARCHAR);

Statement stmt = con.createStatement();
Statement stmt2 = con2.createStatement();
String id;

dataModule.startBinding();


ResultSet master = stmt.executeQuery("Select CarID from car");
try {

    // force error
    String error = request.getParameter("error");
    if (!OZString.isNullOrEmpty(error)) {
        throw new Exception(error);
    }

    dataModule.startSet(masterDataSetName);
    while (master.next()) {
        id = master.getString(1);
        dataModule.setValue(1, id);
        dataModule.completeRow();

        // execute detail query
        if (!OZString.isNullOrEmpty(id)) {
            ResultSet detail = stmt2.executeQuery(
                "Select Maker, CarName, ECarName from car where CarID = '" +
                id + "'");

            dataModule.startSet(detailDataSetName);
            try {
                while (detail.next()) {
                    for (int i = 1; i <= 3; i++)
                        dataModule.setValue(i, detail.getString(i));
```

```
                        dataModule.completeRow();
                    }
                }
                finally {
                    detail.close();
                }
                dataModule.endSet(detailDataSetName);
            }
        }
        dataModule.endSet(masterDataSetName);
    }
    catch (Exception e) {
        dataModule.sendError(e.getMessage());
        throw e;
    }
    finally {
        // close
        master.close();
        stmt.close();
        stmt2.close();
        OZSQL.close(con);
        OZSQL.close(con2);
    }

    dataModule.endBinding();

    dataModule.finish();
}

private void processDAC(HttpServletRequest request,
                        HttpServletResponse responss) throws Exception {
    Connection con = null;

    try {
        String url = "jdbc:odbc:OZCar";
        Properties prop = new Properties();
        prop.put("user", "");
        prop.put("password", "");

        Driver driver = (Driver) Class.forName(
            "sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        con = driver.connect(url, prop);
    }
    catch (Exception e) {
```

```
            OZSQL.close(con);
            e.printStackTrace();
            throw new ServletException(e.getMessage());
        }

        FXDataModule dm = FXDataModuleFactory.createBatchDataModule();
        FXDataAction[] dacs = dm.getDataAction(request);

        try
        {
             con.setAutoCommit(false);
            for (int i = 0; i < dacs.length; i++) {
                FXDataAction dac = dacs[i];
                String type = dac.getActionType();
                if ("insert".equalsIgnoreCase(type)) {
                    // do insert
                    insert(dac, con);
                }
                else if ("delete".equalsIgnoreCase(type)) {
                    // do delete
                    delete(dac, con);
                }
                else if ("rowupdate".equalsIgnoreCase(type)) {
                    // to update
                    update(dac, con);
                }
                else {
                    throw new Exception("Illegal data action type; " + type);
                }
            }
            con.commit();
        }catch(Exception ex) {
            con.rollback();
            ex.printStackTrace();
            throw new ServletException(ex.getMessage());
        }finally {
            OZSQL.close(con);
        }
    }

    private String m_insert, m_update, m_delete;
    private static final String TABLENAME = "car";
    private PreparedStatement m_insertStmt, m_deleteStmt, m_updateStmt;
```

```
private void insert(FXDataAction dac, Connection con) throws Exception {
    FXParameter[] srcFields = dac.getSourceFields();

    if (OZString.isNullOrEmpty(m_insert)) {
        // 맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.
        m_insert = "INSERT INTO " + TABLENAME + " (";
        for (int i = 0; i < srcFields.length; i++) {
            if (i == srcFields.length - 1) {
                m_insert += srcFields[i].getName();
            }
            else {
                m_insert += srcFields[i].getName() + ",";
            }
        }
        m_insert += ") VALUES ( ";

        for (int i = 0; i < srcFields.length; i++) {
            if (i == srcFields.length - 1) {
                m_insert += "?";
            }
            else {
                m_insert += "? ,";
            }
        }

        m_insert += ")";
        m_insertStmt = con.prepareStatement(m_insert);
    }

    for (int i = 0; i < srcFields.length; i++) {
        m_insertStmt.setString(i + 1, encode(srcFields[i].getValue().toString()));
    }

    m_insertStmt.execute();
    if (m_insertStmt != null) m_insertStmt.close();

}

private void update(FXDataAction dac, Connection con) throws Exception {
    FXParameter[] srcFields = dac.getSourceFields();
    FXParameter[] destFields = dac.getTargetFields();

    if (OZString.isNullOrEmpty(m_update)) {
        // 맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.
```

```
            m_update = "UPDATE " + TABLENAME + " set ";

            for (int i = 0; i < srcFields.length; i++) {
                if (i == srcFields.length - 1) {
                    m_update += srcFields[i].getName() + " = ? ";
                }
                else {
                    m_update += srcFields[i].getName() + " = ?, ";
                }
            }
            m_update += " WHERE ";

            for (int i = 0; i < destFields.length; i++) {
                if (i == destFields.length - 1) {
                    m_update += destFields[i].getName() + " = ? ";
                }
                else {
                    m_update += destFields[i].getName() + " = ? AND ";
                }
            }

            m_updateStmt = con.prepareStatement(m_update);
        }

        int i = 0;
        for (i = 0; i < srcFields.length; i++) {
            m_updateStmt.setString(i + 1, encode(srcFields[i].getValue().toString()));
        }

        for (int k = 0; k < destFields.length; k++) {
            m_updateStmt.setString(i + 1, encode(destFields[k].getValue().toString()));
            i++;
        }
        m_updateStmt.execute();

        if (m_updateStmt != null) m_updateStmt.close();
    }

    private void delete(FXDataAction dac, Connection con) throws Exception {
        FXParameter[] targetFields = dac.getTargetFields();

        if (OZString.isNullOrEmpty(m_delete)) {
            m_delete = "DELETE FROM " + TABLENAME + " WHERE ";
```

```
                for (int i = 0; i < targetFields.length; i++) {
                    if (i == targetFields.length - 1) {
                        m_delete += targetFields[i].getName() + " = ? ";
                    }
                    else {
                        m_delete += targetFields[i].getName() + " = ? AND ";
                    }
                }

                m_deleteStmt = con.prepareStatement(m_delete);
            }

            int i = 0;
            for (i = 0; i < targetFields.length; i++) {
                m_deleteStmt.setString(i + 1, encode(targetFields[i].getValue().toString()));
            }

            m_deleteStmt.execute();

            if (m_deleteStmt != null) m_deleteStmt.close();

        }

        private String encode(String value) {
            try {
                return new String(value.getBytes("8859_1"), "KSC5601");
            }
            catch (Exception ex) {
                return value;
            }
        }
}
```

➢ 클래스 파일 등록하기

DataAction.class 파일을 WAS에 등록합니다.

- DataAction.class 파일을 웹 서버의 클래스 폴더에 복사합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\classes 폴더에 sample 폴더를 새로 만든 후 클래스 파일을 복사하였습니다.
- DataAction.class 파일을 WAS에 매핑합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF의 web.xml 파일을 편집기로 열어 아래와 같이 편집하여 복사한 클래스 파일을 매핑하였습니다.

```
...
<!-- JSPC servlet mappings start -->
...
<servlet>
      <servlet-name>sample.DataAction</servlet-name>
      <servlet-class>sample.DataAction</servlet-class>
   </servlet>

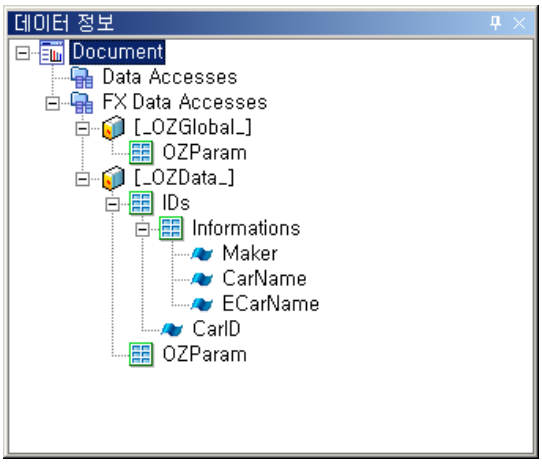
   <servlet-mapping>
            <servlet-name>sample.DataAction</servlet-name>
            <url-pattern>/sample.DataAction</url-pattern>
   </servlet-mapping>...
<!-- JSPC servlet mappings end -->
```

■ **오즈 애플리케이션과 연동하기**

다음은 WAS에 등록한 클래스 파일을 이용하여 오즈 애플리케이션 FX Data Accesses의 "_OZData_" 데이터 모듈에서 마스터 디테일 셋을 추가하고 Table에 데이터를 표시하는 방법을 예를 들어 설명하겠습니다.

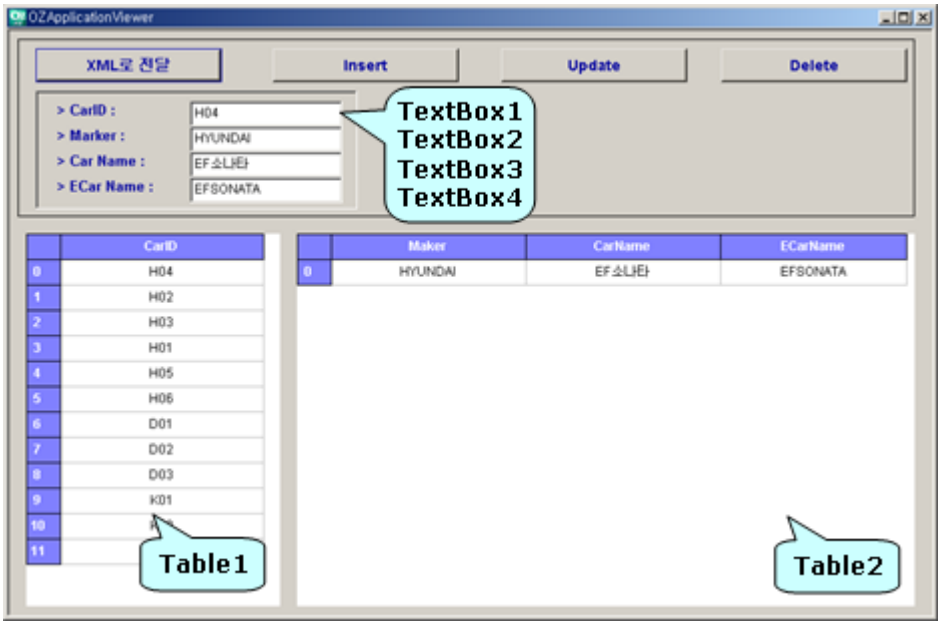
➢ FXDataSet 추가하기

- FX Data Accesses의 "_OZData_"을 마우스 오른쪽 버튼으로 선택하여 나타나는 팝업 메뉴에서 [새 FX DataSet] 메뉴를 클릭하여 아래 그림과 같은 마스터-디테일 구조의 데이터셋을 생성합니다.



➢ 컴포넌트 디자인

Board에 Panel, Label, TextBox, Button, Table을 아래 그림과 같이 추가한 후 컴포넌트의 속성 값을 설정합니다.



- TextBox1

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	IDs
Field	CarID	**-**	-

- TextBox2

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Informations
Field	Maker	**-**	-

- TextBox3

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Informations
Field	CarName	**-**	-

- TextBox4

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Informations
Field	ECarName	**-**	-

- Table1

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	IDs
FireRowCursorChange	True	**-**	-

- Table2

Property	Value	Property	Value
ODIKey	_OZData_	**DataSet**	Informations

➢ 스크립트 입력

정보를 XML로 전달하고 DataAction을 위한 스크립트를 입력합니다.

- [XML로 전달] 버튼의 OnClick 이벤트

```
//XML 접속 객체 생성
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

//데이터 전송
xmlhttp.Open("POST", "http://127.0.0.1:8080/sample.DataAction?type=xml", false);
xmlhttp.Send("");
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
        _MessageBox(dm.FXErrorMessage);
}
```

- [Insert] 버튼의 OnClick 이벤트

```
//XML 접속 객체 생성
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

//데이터 전송
xmlhttp.Open("POST",
http://127.0.0.1:8080/sample.DataAction?type=dac&_OZ_DAC_CNT=1&0.TYPE=insert&0.SRC_CNT=4&0.S
F_0=CarID&0.SFT_0=VARCHAR&0.SV_0=K04&0.SF_1=Maker&0.SFT_1=VARCHAR&0.SV_1=기아자동차&0.SF_2=CarName&0.SFT_2=VARCHAR&0.SV_2=세피아&0.SF_3=ECarName&0.SFT_3=VARCHAR&0.SV_3=SEPHIA, false);
xmlhttp.Send("");

//데이터 전송 결과 확인
var dm = _GetFXDataModule();
```

```
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
        _MessageBox(dm.FXErrorMessage);
}
```

- [Update] 버튼의 OnClick 이벤트

```
//XML 접속 객체 생성
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

//데이터 전송
xmlhttp.Open("POST", "http://127.0.0.1:8080/sample.DataAction?type
=dac&_OZ_DAC_CNT=1&0.TYPE=rowupdate&0.SRC_CNT=1
&0.SF_0=ECarName&0.SFT_0=VARCHAR&0.SV_0=SEPHIA2&0.TRG_CNT=1&0.DF_0=CarID&0.D
FT_0=VARCHAR&0.DV_0=K04", false);
xmlhttp.Send("");

//데이터 전송 결과 확인
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
        _MessageBox(dm.FXErrorMessage);
}
```

- [Delete] 버튼의 OnClick 이벤트

```
//XML 접속 객체 생성
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

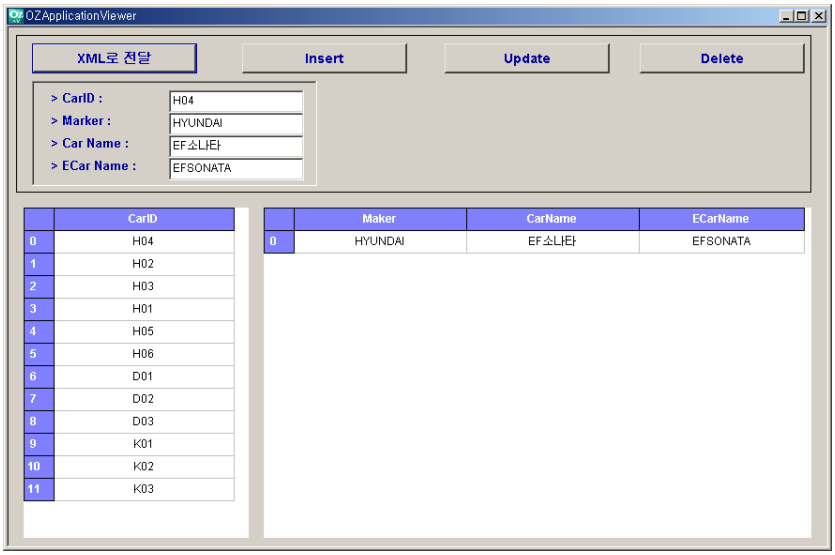
//데이터 전송
xmlhttp.Open("POST",
http://127.0.0.1:8080/sample.DataAction?type=dac&_OZ_DAC_CNT=1&0.TYPE=delete&0.TRG_C
NT=1&0.D
F_0=CarID&0.DFT_0=VARCHAR&0.DV_0=K04, false);
xmlhttp.Send("");

//데이터 전송 결과 확인
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
        _MessageBox(dm.FXErrorMessage);
}
```

➢ 미리보기

정보를 XML로 전달하고 DataAction을 위한 스크립트를 입력합니다.

- [File] 메뉴의 [Preview]를 클릭하거나 툴바의 미리보기 아이콘( )을 클릭하여 미리보기합니다.



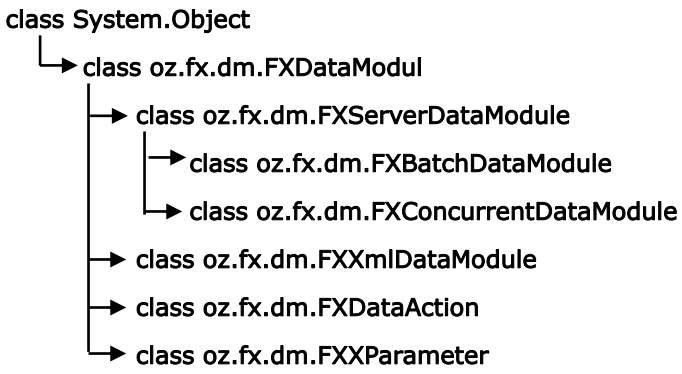
- 텍스트 박스에 추가할 내용을 입력하고 [Insert] 버튼을 클릭한 후 [XML 전달] 버튼을 클릭하면 추가된 내용을 확인할 수 있습니다.
- Update와 Delete 액션도 Insert와 동일한 방법으로 실행하여 확인합니다.

Ⅳ. .Net API 함수 (for OZ .Net Server)

- FXDataModule 관련 함수
- 적용 예

FXDataModule 관련 함수

OZ FX Data의 oz.fx.dm 패키지 구조는 아래와 같으며 각 객체에 대한 설명과 함수는 다음과 같습니다.



oz.fx.dm.FXDataModuleFactory

■ 개요

데이터 모듈을 생성하는 객체입니다.

■ 함수

- CreateBatchDataModule

Prototype	`public static FXDataModule CreateBatchDataModule()`
Definition	전체 데이터에 대한 데이터 모듈 생성 작업을 완료한 후 전송되는 SDM 데이터 모듈을 생성하고, SDM 데이터 모듈 객체를 가져옵니다.

- CreateBatchXmlDataModule

Prototype	`public static FXDataModule CreateBatchXmlDataModule()`
Definition	전체 데이터에 대한 데이터 모듈 생성 작업을 완료한 후 전송되는 형태의 XML 데이터 모듈을 생성하고, XML 데이터 모듈 객체를 가져옵니다.

- CreateConcurrentDataModule

Prototype	public static FXDataModule CreateConcurrentDataModule()
Definition	데이터 모듈 생성 작업과 전송 작업을 동시에 수행하는 형태의 SDM 데이터 모듈을 생성하고, SDM 데이터 모듈 객체를 가져옵니다.

- CreateConcurrentXmlDataModule

Prototype	public static FXDataModule CreateConcurrentXmlDataModule()
Definition	데이터 모듈 생성 작업과 전송 작업을 동시에 수행하는 형태의 XML 데이터 모듈을 생성하고, XML 데이터 모듈 객체를 가져옵니다.

oz.fx.dm.FXDataModule

■ 개요

데이터 모듈을 제어하는 객체입니다.

■ 함수

- AddDataSet

Prototype	public void AddDataSet(String dataSetName) public void AddDataSet(String dataSetName, String masterSetName)	
Definition	디테일 데이터 셋을 추가합니다. ※ 참고사항 : 디테일 셋을 설정할 경우 마스터 셋 이름을 설정합니다.	
Argument	*dataSetName*	데이터 셋 이름
	masterSetName	마스터 셋 이름

- AddField

Prototype	public void AddField(String dataSetName, String fieldName, integer fieldType)	
Definition	데이터 셋에 필드를 추가합니다.	
Argument	*dataSetName*	데이터 셋 이름
	fieldName	필드 이름
	fieldType	필드 타입 (java.sql.Types 참조)

- CompleteRow

Prototype	`public void CompleteRow()`
Definition	하나의 Row에 대한 데이터 값 설정을 완료합니다.

- EndSet

Prototype	`public void EndSet(String dataSetName)`
Definition	데이터 셋 바인딩을 종료합니다.
Argument	*dataSetName* 데이터 셋 이름

- EndBinding

Prototype	`public void EndBinding()`
Definition	데이터 모듈 바인딩을 종료합니다.

- Finish

Prototype	`public void Finish()`
Definition	모든 작업을 종료하며, 데이터 전송 타입이 batch 타입일 경우 데이터 전송 작업이 시작됩니다.

- Initialize

Prototype	`public void Initialize(java.io.OutputStream out)`
	
	`public void Initialize(boolean isCompressed, bool isEncrypted, java.io.OutputStream out)`
	
	`public void Initialize(boolean isCompressed, java.io.OutputStream out, integer bufferSize)`

Definition	데이터 모듈을 기록할 정보를 초기화하며 데이터 압축 여부와 버퍼 크기를 설정할 수 있습니다. FXDataModule의 type=sdm, concurrent=false, compressed=true일 경우 암호화됩니다. ※ 참고사항 : 데이터 바인딩 시작 전(startBinding() 함수 호출 전)에 호출되어야 합니다.
Argument	*isCompressed* 압축 여부
	isEncrypted 암호화 여부 (암호화 시 128bits DES 방식 사용)
	out 출력 스트림
	bufferSize 버퍼 크기, 단위 : byte (기본 값 : 4096)

- GetDataAction

Prototype	`public FXDataAction[] GetDataAction(HttpRequest request)`
Definition	Request로부터 패러미터를 파싱하여 FXDataAction 객체를 리턴합니다.
Argument	*request* Request 객체

- StartBinding

Prototype	`public void StartBinding()`
Definition	메타 정보가 모두 등록되었을 때 데이터 모듈 바인딩을 시작하며 메타 정보의 무결성을 검증합니다.

- StartSet

Prototype	`public void StartSet(String dataSetName)`
Definition	데이터 셋 바인딩을 시작합니다. ※ 참고사항 : startSet(String) 메서드가 호출된 적이 있는 경우 해당 데이터 셋의 디테일 셋 형태로 데이터가 기록됩니다.
Argument	*dataSetName* 데이터 셋 이름

- SetValue

Prototype	`public void setValue(integer ColumndIndex, Object value)`	
Definition	현재 바인딩 중인 데이터 셋에 값을 설정합니다.	
Argument	*ColumndIndex*	열 인덱스 ※ 참고사항 : 해당 인덱스는 0 부터 시작되는 인덱스입니다.
	value	값

- SendError

Prototype	`public void SendError(String errorMessage)`
Definition	에러 메시지를 전송합니다.
Argument	*errorMessage* 에러 메시지

oz.fx.dm.FXDataAction

■ 개요

데이터 액션을 제어하는 객체입니다.

■ 함수

– FXDataAction

Prototype	`public FXDataAction(string name, string actionType)`
	
	`public FXDataAction(string name, string actionType, string ext)`

Definition	DataAcion을 생성합니다.
Argument	*name* 데이터셋 이름
	actionType DataAction 타입 (insert, rowupdate, delete)
	ext ext 값

– DataSetName

Prototype	`public string DataSetName{get;}`
Definition	데이터셋 이름을 리턴합니다.

– ActionType

Prototype	`public string ActionType{get;}`
Definition	DataAction 타입을 리턴합니다.
Argument	DataAction 타입 (insert, rowupdate, delete)

– ExtraArgument

Prototype	`public string ExtraArgument{get; set;}`
Definition	ext 값을 리턴하거나 설정합니다.

– SourceFields

Prototype	`public FXParameter[] SourceFields{get; set;}`
Definition	Source 필드 정보를 리턴하거나 설정합니다.

– TargetFields

Prototype	`public FXParameter[] TargetFields{get; set;}`
Definition	Target 필드 정보를 리턴하거나 설정합니다.

oz.fx.dm. FXParameter

■ 개요

데이터 액션 시 패러미터를 제어하는 객체입니다.

■ 함수

- FXDataAction

Prototype	`public FXParameter(string name, Object value)` `public FXParameter(string name, int type, Object value)`
Definition	FXDataModule의 패러미터를 생성합니다.
Argument	*name* 패러미터 이름 *type* 패러미터 데이터 타입 (sqlType과 동일) *value* 패러미터 데이터 값

- Value

Prototype	`public object Value{get;}`
Definition	패러미터 값을 리턴합니다.

- isNull

Prototype	`public bool isNull{get; set;}`
Definition	null 값의 허용 여부를 리턴하거나 설정합니다.

- Name

Prototype	`public string Name{get;}`
Definition	패러미터 이름을 리턴합니다.

- Type

Prototype	`public int Type{get;}`
Definition	패러미터 데이터 타입을 리턴합니다. (sqlType과 동일)

적용 예

다음은 ASP.NET API 함수를 사용하는 예에 대해 설명합니다.
오즈 애플리케이션 또는 오즈 리포트에 생성된 어셈블리 파일을 연동하는 방법은 Ⅲ. Servlet API 함수 (for OZ Java Server)의 적용 예를 참조하시기 바랍니다.

예제 1 : 데이터 검색

다음은 FXDataModuler과 연동하여 데이터를 검색하는 어셈플리 파일을 생성하는 방법에 대해 설명합니다.

본 예제에서는 아래와 같은 어셈블리 소스 파일을 이용하여 DataModuleSample.aspx 파일로 만들었습니다.

```
<%@import Namespace="System.Data" %>
<%@import Namespace="System.Data.Odbc" %>
<%@import Namespace="oz.fx.dm" %>
<Script language="c#" runat="server">

  public void Page_Load(Object sender, EventArgs e)
  {
    //FX Data의 프레임웍 URL에 설정한 옵션에 따라 데이터 모듈의 데이터 Fetch 타입 및 데이터 타입을 생성
    bool concurrent = 0 == string.Compare("true", Request.Params["concurrent"], true);
    bool isSDM = 0 != string.Compare("true", Request.Params["xml"], true);
    bool isCompressed = 0 == string.Compare("true", Request.Params["compress"], true);

    string rowCount = Request.Params["addRowCount"];

    int addRowCount = 0;

    if(null != rowCount && 0 != rowCount.Length)
        addRowCount = int.Parse(rowCount);

    Response.ContentType = isSDM || isCompressed ? "application/octet-stream" : "text/xml";

    FXDataModule dataModule = null;

    //SDM 타입의 데이터 모듈 객체 생성
    if (isSDM){
```

```
        if (concurrent)
            dataModule = FXDataModuleFactory.CreateConcurrentDataModule();
        else
            dataModule = FXDataModuleFactory.CreateBatchDataModule();
    }
    else{
        if(concurrent)
            dataModule = FXDataModuleFactory.CreateConcurrentXmlDataModule();
        else
            dataModule = FXDataModuleFactory.CreateBatchXmlDataModule();
    }

    try{
       //데이터 모듈 정보 초기화
       dataModule.Initialize(isCompressed, isEncrypt, Response.OutputStream);
        bindDataModule(dataModule, addRowCount);
    }
    catch(System.IO.IOException)
    {
        // pass
    }
    catch(Exception ex)
    {
        dataModule.SendError(ex.Message);
        return;
    }

    dataModule.Finish();
}



private void bindDataModule(FXDataModule dataModule, int additionalRowCount)
{
    using(OdbcConnection masterConnection = new OdbcConnection("Dsn=ozdemo"))
    using(OdbcConnection detailConnection = new OdbcConnection("Dsn=ozdemo"))
    using(IDbCommand masterCommand = masterConnection.CreateCommand())
    using(IDbCommand detailCommand = detailConnection.CreateCommand())
    {
        masterConnection.Open();
        detailConnection.Open();

        dataModule.AddParameter("param1", SqlTypes.VARCHAR, "This is parameter 1");
        dataModule.AddParameter("param2", SqlTypes.VARCHAR, "This is parameter 2");
```

```
            dataModule.AddParameter("param3", SqlTypes.VARCHAR, "This is parameter 3");
            dataModule.AddParameter("param4□"<>", SqlTypes.VARCHAR, null);

            //메타 데이터 셋 및 데이터 필드 생성
            string masterDataSetName = "K_Car";
            dataModule.AddDataSet(masterDataSetName);
            dataModule.AddField(masterDataSetName, "CarID", SqlTypes.VARCHAR);

            string detailDataSetName = "K_CarOrders";
            dataModule.AddDataSet(detailDataSetName, masterDataSetName);
            dataModule.AddField(detailDataSetName, "OrderDate", SqlTypes.VARCHAR);
            dataModule.AddField(detailDataSetName, "Region", SqlTypes.VARCHAR);
            dataModule.AddField(detailDataSetName, "BranchOffice", SqlTypes.VARCHAR);

            string id;

            if("before_start_binding".Equals(Request.Params["error"]))
                throw new Exception("This is intended exception. " + Request.Params["message"]);

            dataModule.StartBinding();
             //마스터 셋 쿼리문 실행
            masterCommand.CommandText = "Select CarID from K_Car";

            using(IDataReader master = masterCommand.ExecuteReader())
            {
              //마스터 셋 바인딩 시작
                dataModule.StartSet(masterDataSetName);
                while (master.Read())
                {
                    id = master.GetString(0);
                     //마스터 셋에 값 설정
                    dataModule.SetValue(0, id);
                    dataModule.CompleteRow();

                    //디테일 셋 바인딩 시작
                    dataModule.StartSet(detailDataSetName);
                     //디테일 셋 쿼리문 실행
                    if (null != id && 0 != id.Length)
                    {
                        detailCommand.CommandText = "Select OrderDate, Region, BranchOffice from
K_CarOrders where CarID = '" +
                            id + "'";
                        using(IDataReader detail = detailCommand.ExecuteReader())
                        {
```

```
                        while (detail.Read())
                        {
                            for (int i = 0; i < 3; i++)
                                //디테일 셋에 값 설정
                                dataModule.SetValue(i, detail.GetString(i));

                            dataModule.CompleteRow();

                            for(int i = 0; i < additionalRowCount; i++)
                            {
                                dataModule.CompleteRow();
                            }
                        }
                    }
                    //디테일 셋 바인딩 완료
                    dataModule.EndSet(detailDataSetName);
                }

                if("during_binding".Equals(Request.Params["error"]))
                    throw new Exception("This is intended exception. " + Request.Params["message"]);

                //마스터 셋 바인딩 완료
                dataModule.EndSet(masterDataSetName);
            }
        }
         //데이터 모듈 바인딩 완료
        dataModule.EndBinding();
    }

</Script>
```

예제 2 : 데이터 액션

다음은 FXDataModuler과 연동하여 오즈 애플리케이션의 폼에 데이터를 표시하고 표시된 데이터를 삽입, 삭제, 변경하는 어셈블리 파일을 생성하는 방법에 대해 설명합니다.

본 예제에서는 아래와 같은 어셈블리 소스 파일을 이용하여 DataActionSample.aspx 파일로 만들었습니다.

```
<%@ Page language="C#" runat="server" Debug="true"%>
```

```
<%@ Import namespace="System" %>
<%@ Import namespace="System.Collections" %>
<%@ Import namespace="System.ComponentModel" %>
<%@ Import namespace="System.Data" %>
<%@ Import namespace="System.Data.Odbc" %>
<%@ Import namespace="System.Drawing" %>
<%@ Import namespace="System.Web" %>
<%@ Import namespace="System.Web.SessionState" %>
<%@ Import namespace="System.Web.UI" %>
<%@ Import namespace="System.Web.UI.WebControls" %>
<%@ Import namespace="System.Web.UI.HtmlControls" %>
<%@ Import namespace="oz.fx" %>
<%@ Import namespace="oz.fx.dm" %>
<%@ Import namespace="log4net" %>

<script language="C#" runat="server">

                        string TABLENAME = "car";
    private                         static                          ILog                            logger
=LogManager.GetLogger(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);
                        private void Page_Load(object sender, System.EventArgs e)
                        {
                                    string type = Request["type"];
            if(type==null || type.Length==0) type = "xml";
            logger.Debug("Page_Load()");
            // is compressed??
            bool isCompressed = EqualsIgnoreCase("true", Request["compressed"]);
            bool concurrent = EqualsIgnoreCase("true", Request["concurrent"]);
            // type is null
            if (IsNullOrEmpty(type))
            {
                throw new Exception("Operation type not specified.");
            }

            // dispatch of type
            try
            {
                bool isSDM = EqualsIgnoreCase("sdm", type);
                if(isSDM)
                    processSDM(Request,Response,true,isCompressed);
                else if(EqualsIgnoreCase("xml",type))
                    processSDM(Request,Response,false,isCompressed);
                else if(EqualsIgnoreCase("dac",type))
                    processDAC(Request,Response);
```

```
                else
                    throw new Exception("Unknown operation code; " + type);
            }
            catch(Exception ex)
            {
                throw ex;
            }

                }

                private void processSDM(HttpRequest request,HttpResponse response,bool
isSDM,bool
            isCompressed)
        {
            logger.Debug("processSDM()");
            OdbcConnection con=null;
            OdbcConnection con2=null;
            try
            {
                bool concurrent = EqualsIgnoreCase("true", request["concurrent"]);
                FXDataModule dataModule = null;
                if (isSDM)
                {
                    if (concurrent)
                        dataModule = FXDataModuleFactory.CreateConcurrentDataModule();
                    else
                        dataModule = FXDataModuleFactory.CreateBatchDataModule();
                }
                else
                {
                    if(concurrent)
                        dataModule = FXDataModuleFactory.CreateConcurrentXmlDataModule();
                    else
                        dataModule = FXDataModuleFactory.CreateBatchXmlDataModule();
                }
                dataModule.Initialize(isCompressed, isEncrypt, response.OutputStream);
                // Access does not support multiple ResultSet
                // set content type
                response.ContentType = (isSDM || isCompressed) ? "application/octet-stream" :
"text/xml";
                //System.Data.Odbc.OdbcConnection
                string masterDataSetName = "IDs";
                dataModule.AddDataSet(masterDataSetName);
                dataModule.AddField(masterDataSetName, "CarID", SqlTypes.VARCHAR);
```

```
String detailDataSetName = "Informations";
dataModule.AddDataSet(detailDataSetName, masterDataSetName);
dataModule.AddField(detailDataSetName, "Maker", SqlTypes.VARCHAR);
dataModule.AddField(detailDataSetName, "CarName", SqlTypes.VARCHAR);
dataModule.AddField(detailDataSetName, "ECarName", SqlTypes.VARCHAR);
//start binding
dataModule.StartBinding();
// force error
string error = Request["error"];
if (!IsNullOrEmpty(error))
{
    throw new Exception(error);
}
con= new OdbcConnection("dsn=ozcar;UID=;PWD=;");
con.Open();
con2= new OdbcConnection("dsn=ozcar;UID=;PWD=;");
con2.Open();
OdbcCommand cmd = new OdbcCommand();
cmd.Connection = con;
cmd.CommandType = CommandType.Text;
cmd.CommandText = "Select CarID from car";
OdbcDataReader dr = cmd.ExecuteReader();
OdbcCommand cmd2 = new OdbcCommand();
cmd2.Connection = con2;
cmd2.CommandType = CommandType.Text;
dataModule.StartSet(masterDataSetName);
string CarID = string.Empty;
//execute master 반복이 되어야 한다.
while(dr.Read())
{
    CarID = dr.GetString(0);
    dataModule.SetValue(0,CarID);
    dataModule.CompleteRow();
    cmd2.CommandText="Select Maker, CarName, ECarName from car where CarID = '"
+ CarID + "'";
    OdbcDataReader detailDR = cmd2.ExecuteReader();
    dataModule.StartSet(detailDataSetName);

    while(detailDR.Read())
    {
        for (int i = 0; i < detailDR.FieldCount; i++)
        {
            dataModule.SetValue(i,detailDR.GetValue(i));
        }
```

```
                    dataModule.CompleteRow();
                }
                dataModule.EndSet(detailDataSetName);
                detailDR.Close();
            }
            dataModule.EndSet(masterDataSetName);
            dr.Close();
            dataModule.EndBinding();
            dr.Close();
            dataModule.Finish();
        }
        catch(Exception e)
        {
            throw e;
        }
        finally
        {
            if(con!=null)con.Close();
            if(con2!=null)con2.Close();
        }
    }

    private void processDAC(HttpRequest request,HttpResponse response)
    {
        logger.Debug("processSDM()");
        OdbcConnection con=null;
        try
        {
            con= new OdbcConnection("dsn=ozcar;UID=;PWD=;");
            con.Open();
            //con.BeginTransaction();
        }
        catch (Exception e)
        {
            con.Close();
            throw e;
        }
        FXDataModule dm = FXDataModuleFactory.CreateBatchDataModule();
        FXDataAction[] dacs = dm.GetDataAction(request);
        try
        {
            for (int i = 0; i < dacs.Length; i++)
            {
                FXDataAction dac = dacs[i];
```

```
                String type = dac.ActionType;
                if(EqualsIgnoreCase("insert",type))
                {
                    // do insert
                    insert(dac, con);
                }
                else if(EqualsIgnoreCase("delete",type))
                {
                    // do delete
                    delete(dac, con);
                }
                else if(EqualsIgnoreCase("rowupdate",type))
                {
                    // to update
                    update(dac, con);
                }
                else
                {
                    throw new Exception("Illegal data action type; " + type);
                }
            }
            //con.commit();
        }
        catch(Exception ex)
        {
            //con.rollback();
            string msg = ex.Message + "□n" + ex.StackTrace;
            logger.Error(System.DateTime.Now + msg);
            throw ex;
        }
        finally
        {
            con.Close();
        }
    }

    private void insert(FXDataAction dac,OdbcConnection con)
    {
        OdbcCommand cmd = new OdbcCommand();
        cmd.Connection = con;
        cmd.CommandType = CommandType.Text;
        string m_insert = string.Empty;
        FXParameter[] srcFields = dac.SourceFields;
        if(IsNullOrEmpty(m_insert))
```

```
        {
            // 맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.
            m_insert = "INSERT INTO " + TABLENAME + " (";
            for (int i = 0; i < srcFields.Length; i++)
            {
                if (i == srcFields.Length - 1)
                {
                    m_insert += srcFields[i].Name;
                }
                else
                {
                    m_insert += srcFields[i].Name + ",";
                }
            }
            m_insert += ") VALUES ( ";
            for (int i = 0; i < srcFields.Length; i++)
            {
                if (i == srcFields.Length - 1)
                {
                    m_insert += "?";
                }
                else
                {
                    m_insert += "? ,";
                }
            }
            m_insert += ")";
            cmd.CommandText = m_insert;
        }
        for (int i = 0; i < srcFields.Length; i++)
        {
            cmd.Parameters.Add("@field"+i,OdbcType.VarChar);
            cmd.Parameters[i].Value=srcFields[i].Value.ToString();
        }
        logger.Debug(m_insert);
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }

    private void delete(FXDataAction dac,OdbcConnection con)
    {
        OdbcCommand cmd = new OdbcCommand();
        cmd.Connection = con;
        cmd.CommandType = CommandType.Text;
```

```
        FXParameter[] targetFields = dac.TargetFields;
        string m_delete = string.Empty;
        if(IsNullOrEmpty(m_delete))
        {
            m_delete = "DELETE FROM " + TABLENAME + " WHERE ";
            for (int i = 0; i < targetFields.Length; i++)
            {
                if (i == targetFields.Length - 1)
                {
                    m_delete += targetFields[i].Name + " = ? ";
                }
                else
                {
                    m_delete += targetFields[i].Name + " = ? AND ";
                }
            }
            cmd.CommandText = m_delete;
        }
        for (int i = 0; i < targetFields.Length; i++)
        {
            cmd.Parameters.Add("@field"+i,OdbcType.VarChar);
            cmd.Parameters[i].Value=targetFields[i].Value.ToString();
        }
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }

    private void update(FXDataAction dac,OdbcConnection con)
    {
        logger.Debug("update()");
        OdbcCommand cmd = new OdbcCommand();
        cmd.Connection = con;
        cmd.CommandType = CommandType.Text;
        FXParameter[] srcFields = dac.SourceFields;
        FXParameter[] destFields = dac.TargetFields;
        string m_update = string.Empty;
        if(IsNullOrEmpty(m_update))
        {
            // 맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.
            m_update = "UPDATE " + TABLENAME + " set ";
            for (int i = 0; i < srcFields.Length; i++)
            {
                if (i == srcFields.Length - 1)
                {
```

```
                    //m_update += srcFields[i].Name + " = ? ";
                    m_update += srcFields[i].Name + " = '"+srcFields[i].Value+"' ";
                }
                else
                {
                    //m_update += srcFields[i].Name + " = ?, ";
                    m_update += srcFields[i].Name + " = '"+srcFields[i].Value+"', ";
                }
            }
            m_update += " WHERE ";
            for (int i = 0; i < destFields.Length; i++)
            {
                if (i == destFields.Length - 1)
                {
                    //m_update += destFields[i].Name + " = ? ";
                    m_update += destFields[i].Name + " = '"+destFields[i].Value+"' ";
                }
                else
                {
                    m_update += destFields[i].Name + " = '"+destFields[i].Value+"' AND ";
                }
            }
            cmd.CommandText = m_update;
            logger.Debug(m_update);
        }
        /*
        int ii=0;
        for (ii = 0; ii < srcFields.Length; ii++)
        {
        cmd.Parameters.Add("@field"+ii,OdbcType.VarChar);
        cmd.Parameters[ii].Value=srcFields[ii].Value.ToString();
        }
        for (int k = 0; k < destFields.Length; k++)
        {
        cmd.Parameters.Add("@field"+ii,OdbcType.VarChar);
        cmd.Parameters[ii].Value=destFields[k].Value.ToString();
        ii++;
        }*/
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }

    private string encode(string src)
    {
```

```
            System.Text.Encoding e = System.Text.Encoding.GetEncoding(949);
            //return ToByteArray(src);
            //Encoding e = Encoding.GetEncoding(encoding);
            return new string(System.Text.Encoding.Default.GetChars(e.GetBytes(src)));
            //return e.GetBytes(src);
        }
        private bool EqualsIgnoreCase(string str1, string str2)
        {
            if(IsNullOrEmpty(str1) || IsNullOrEmpty(str2))
                return false;
            return 0 == string.Compare(str1, str2, true);
        }
        private bool IsNullOrEmpty(string s)
        {
            return (null == s) || (0 == s.Length);
        }

</script>
```