

OZ Framework Manual

OZ Framework	2
POST	3
DataModule POST	3
FXDataModule POST Custom	5
Servlet API (for OZ Java Server)	12
DataModuleFactory	12
DataModule	13
FXDataModule	19
.....	26
Servlet API (for OZ .Net Server)	27
FXDataModule	27
Servlet API	33
1 : Application -	33
2 : Application - DataAction	48
3 : Report -	59
4 : XML SDM	64
5 : FXDataModule	76

OZ Java Framework

2 *FORCS Co., LTD*

POST

```
graph LR
    POST[POST] --> ODI[ODI]
    ODI --> FXDataModule[FXDataModule]
    POST --> DataModule[DataModule]
    DataModule --> POST
    DataModule --> FXDataModule
    FXDataModule --> POST
    FXDataModule --> DataModule
```

DataModule POST

■ POST

_OZ_ODIFetchType_	DM_BATCH_FETCH DM_CONCURRENT_FETCH FetchUnit DM_PER_DATASET DM_CONCURRENT_FETCH .
_OZ_ODIITEM_	ODI
_OZ_ODICATEGORY_	ODI
_OZ_DATASET_	FetchUnit DM_PER_DATASET .
_OZ_DEBUG_	Debug (true / false) : XML SDM Debug .
MyParam	ODI ODI

- **DataAction** POST

- **DataAction** POST

MyParam	ODI ODI

_OZ_DAC_CNT	DataAction <INDEX>
<INDEX>.DATASET	DataAction
<INDEX>.TYPE	DataAction CUD Insert, Delete, RowUpdate
<INDEX>.EXT	DataAction Extra
<INDEX>.SRC_CNT	DataAction Source <INDEX2>
<INDEX>.SF_<INDEX2>	DataAction <INDEX2> SourceName
<INDEX>.SV_<INDEX2>	DataAction <INDEX2> SourceValue
<INDEX>.TRG_CNT	DataAction Target <INDEX2>
<INDEX>.DF_<INDEX2>	DataAction <INDEX2> TargetName
<INDEX>.DV_<INDEX2>	DataAction <INDEX2> TargetValue

: 3 ODI 가 , DataAction 3 Commit
POST

```

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

```

FXDataModule POST Custom

■ DataAction

POST

_OZ_DAC_CNT_	DataAction
<DAC_INDEX>.DATASET	
<DAC_INDEX>.TYPE	DataAction (Insert, RowUpdate, Delete)
<DAC_INDEX>.EXT	DataAction ext
<DAC_INDEX>.SRC_CNT	DataAction Source
<DAC_INDEX>.SF_<SRC_INDEX>	DataAction Source
<DAC_INDEX>.SFT_<SRC_INDEX>	DataAction Source
<DAC_INDEX>.SV_<SRC_INDEX>	DataAction Source (null)
<DAC_INDEX>.TRG_CNT	DataAction Target
<DAC_INDEX>.DF_<TRG_INDEX>	DataAction Target
<DAC_INDEX>.DFT_<TRG_INDEX>	DataAction Target
<DAC_INDEX>.DV_<TRG_INDEX>	DataAction Target (null)

■ -

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"

: BINARY, VARBINARY, LONGVARBINARY, BLOB, CLOB
BASE64 .

■ (enum FX_DataTypes) -

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[]

: BINARY, VARBINARY, LONGVARBINARY, BLOB, CLOB
BASE64 .

■ Custom

- DataModule ()

Definition	
Child	Parameter, DataSetMeta, DataSet

- DataModule ()

Definition		
Attribute	<i>runat</i>	"server"
Child	Parameter, DataSetMeta, DataSet	

- Parameter

Definition	
Attribute	<i>name</i>
	<i>fieldType</i>

	Text	
-	DataSetMeta	
	Definition	
	Attribute	<i>name</i>
		<i>masterSetName</i>
	Child	DataFieldMeta
-	DataFieldMeta	
	Definition	
	Attribute	<i>fieldName</i>
		<i>fieldType</i>
-	DataSet	
	Definition	
	Attribute	<i>name</i>
	Child	Record
-	Record	
	Definition	
	Child	Column, DataSet
-	Column	
	Definition	
	Attribute	<i>Name</i>
	Text	
-	Send	
	Definition	
	Attribute	<i>isCompress</i> (gzip)
		<i>isSDM</i> (true : SDM, false : XML)
	Child	Error

- Error

Definition

Text

Servlet 2.3

DataAction

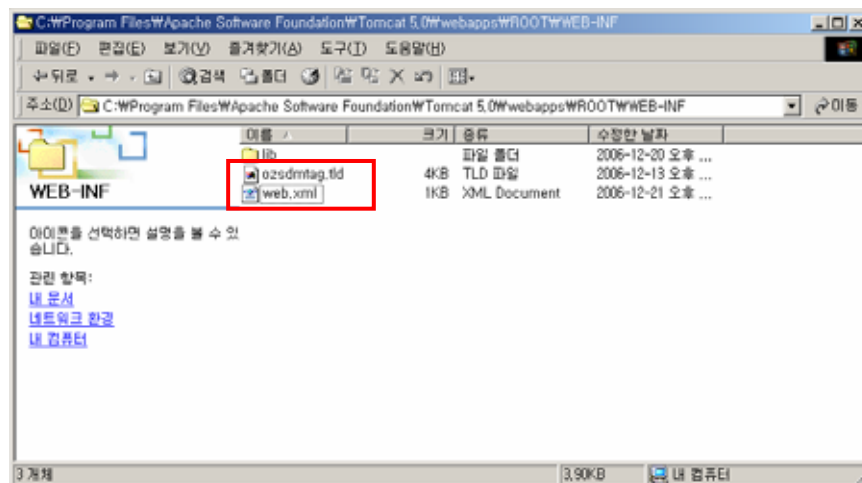
DataModule

runat="server"

■ Custom

- JSP

WAS web.xml Custom location
ozsdmtag.tld



web.xml

```
...

<!-- JSPC servlet mappings start -->
...
<taglib>
  <taglib-uri>http://www.forcs.com/oz/fxsdmap1/taglib</taglib-uri>
  <taglib-location>/WEB-INF/ozsdmtag.tld</taglib-location>
</taglib>
...
<servlet-mapping>
```

```

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

```

```

        jsp      HTML
        . URL    web.xml
        URL
        .

```

```

<%@ taglib uri="http://www.forcs.com/oz/fxsdmapi/taglib" prefix="oz" %>

```

- ASP.NET

OZSDMAPI 가

aspx HTML

```

<%@ Register TagPrefix="oz" Namespace="oz.fxapi.custom.tag"
Assembly="OZSDMAPI" %>

```

```

<oz:DataModule runat="server">

<oz:DataSetMeta name="IDs" >
    <oz:DataFieldMeta fieldName="CarID" fieldType="VARCHAR" />
</oz:DataSetMeta>
<oz:DataSetMeta name="Informations" masterSetName="IDs" >
    <oz:DataFieldMeta fieldName="Maker" fieldType="VARCHAR" />
    <oz:DataFieldMeta fieldName="CarName" fieldType="CLOB" />
    <oz:DataFieldMeta fieldName="ECarName" fieldType="VARCHAR" />
</oz:DataSetMeta>
<oz:Parameter name="param1" fieldType="INTEGER">35</oz:Parameter>
<oz:DataSet name="IDs">
    <oz:Record>
        <oz:Column name="CarID" >08</oz:Column>
        <oz:DataSet name="Informations" >
            <oz:Record>
                <oz:Column name="Maker" >HYUNDAI</oz:Column>
                <oz:Column name="CarName" >EF      </oz:Column>
                <oz:Column name="ECarName" >EFSONATA</oz:Column>
            </oz:Record>
        </oz:DataSet>
    </oz:Record>
    <oz:Record>
        <oz:Column name="CarID" >05</oz:Column>
        <oz:DataSet name="Informations" >

```

```
<oz:Record>
  <oz:Column name="Maker" >HYUNDAI</oz:Column>
  <oz:Column name="CarName" >          </oz:Column>
  <oz:Column name="ECarName" >DYNASTY</oz:Column>
</oz:Record>
</oz:DataSet>
</oz:Record>
<oz:Record>
  <oz:Column name="CarID" >07</oz:Column>
  <oz:DataSet name="Informations" >
    <oz:Record>
      <oz:Column name="Maker" >HYUNDAI</oz:Column>
      <oz:Column name="CarName" >          </oz:Column>
      <oz:Column name="ECarName" >GRANDEUR</oz:Column>
    </oz:Record>
  </oz:DataSet>
</oz:Record>
<oz:Record>
  <oz:Column name="CarID" >04</oz:Column>
  <oz:DataSet name="Informations" >
    <oz:Record>
      <oz:Column name="Maker" >HYUNDAI</oz:Column>
      <oz:Column name="CarName" >          </oz:Column>
      <oz:Column name="ECarName" >EQUUS</oz:Column>
    </oz:Record>
  </oz:DataSet>
</oz:Record>
<oz:Record>
  <oz:Column name="CarID" >09</oz:Column>
  <oz:DataSet name="Informations" >
    <oz:Record>
      <oz:Column name="Maker" >HYUNDAI</oz:Column>
      <oz:Column name="CarName" >          </oz:Column>
      <oz:Column name="ECarName" >VERNA</oz:Column>
    </oz:Record>
  </oz:DataSet>
</oz:Record>
</oz:DataSet>

<oz:Send isCompressed="false" isSDM="true">
<oz:Error>Not implemented</oz:Error>
</oz:Send>

</oz:DataModule>
```

Servlet API (for OZ Java Server)

DataModuleFactory

■ DataModuleFactory

- getDataModule

Prototype	public static DataModule getDataModule(String FetchType) throws OZSDMException
Definition	가 "DM_CONCURRENT_FETCH" getDataModule CONCURRENT 가
Argument	FetchType DM_CONCURRENT_FETCH / DM_BATCH_FETCH / DM_PER_DATASET

DM_CONCURRENT_FETCH :
DM_BATCH_FETCH :
DM_PER_DATASET :

DataModule

■ DataModule

- init

Prototype	public void init(OutputStream out) throws IOException
Definition	Stream .
Argument	<i>out</i> Response OutputStream

- startBinding

Prototype	public void startBinding() throws IOException, SQLException
Definition	. .

- endBinding

Prototype	public void endBinding() throws IOException, SQLException
Definition	. .

- startSet

Prototype	public void startSet(String DatasetName) throws IOException
Definition	. .
Argument	<i>DatasetName</i>

- endSet

Prototype	public void endSet(String DatasetName) throws IOException
Definition	. .
Argument	<i>DatasetName</i>

- addParameter

Prototype	public void addParameter(String ParamName, int FieldType, Object Value)
------------------	---

Definition	가	
	, getDataModule()	
	DM_CONCURRENT_FETCH	DM_BATCH_FETCH
Argument	ParamName	
	FieldType	
	Value	

- addSetInfo

Prototype	public void addSetInfo(String DatasetName, String MasterDatasetName, String[] FieldNames, int[] FieldTypes) throws IllegalArgumentException		
	Definition	, DM_CONCURRENT_FETCH DM_BATCH_FETCH 가	
		DatasetName	
Argument		MasterDatasetName	
	FieldNames		
	FieldTypes		

- addSetInfo

Prototype	public void addSetInfo(String DatasetName, String[] FieldNames, int[] FieldTypes) throws IllegalArgumentException		
	Definition	, DM_PER_DATASET 가	
		DatasetName	
Argument		FieldNames	
	FieldTypes		

- addRow

Prototype	public void addRow(String DatasetName, HashMap hMap) throws IOException, SQLException	
-----------	---	--

Definition	ResultSet	Row	가	.
	DatasetName			
		HashMap	가	Row
		)		
		...		
		rs = stmt.executeQuery(query);		
		while(rs.next()) {		
		HashMap hMap = new HashMap();		
Argument	hMap	hMap.put(_FN[0], rs.getString(_FN[0]));		
		hMap.put(_FN[1], rs.getString(_FN[1]));		
		hMap.put(_FN[2], rs.getString(_FN[2]));		
		hMap.put(_FN[3], rs.getString(_FN[3]));		
		hMap.put(_FN[4], rs.getString(_FN[4]));		
		hMap.put(_FN[5], rs.getString(_FN[5]));		
		module.addRow("SET_1", hMap);		
		}		
		...		

- addRow

Prototype	public void addRow(String DatasetName, List list) throws IOException, SQLException			
Definition	ResultSet	Row	가	.
	<i>DatasetName</i>			
Argument	<i>list</i>	ArrayList	가	Row

- addRow

Prototype	public void addRow(String DatasetName, String[] arr) throws IOException, SQLException			
Definition	ResultSet	Row	가	.
	<i>DatasetName</i>			
Argument	<i>arr</i>	String[]	가	Row

- makeSDM_SET

Prototype	public void makeSDM_SET(String DatasetName, ResultSet rs, OutputStream out) throws OZSDMException, IOException		
Definition	DM_PER_DATASET	SDM	.
	, Batch		.
	<i>DatasetName</i>		
Argument	<i>rs</i>	ResultSet	
	<i>out</i>	Response	OutputStream

- ParseXML

Prototype	public string ParseXML(XMLID)		
Definition	XML	SDM	.
Argument	<i>XMLID</i>	XML ID	
Return	<i>Success</i>		

- sendBindErrorMessage

Prototype	public void sendBindErrorMessage(String msg) throws IOException		
Definition	:	가	.
Argument	<i>msg</i>		

- sendErrorMessage

Prototype	public void sendErrorMessage(String msg, OutputStream out) throws IOException		
Definition	:	가	.
Argument	<i>msg</i>		
	<i>out</i>	Response	OutputStream

- setDebug

Prototype	public void setDebug(boolean isDebug, HttpServletResponse response) throws IOException
Definition	Debug : XML SDM Debug : init , "setDebug method can't call after init method calling and call only one time" 가
Argument	Debug <i>isDebug</i> - true : Debug - false : Debug () <i>response</i>

■ Servlet API

- 가

	가	→ <i>getDataModule();</i>
		→ <i>init();</i>
ODI	가	→ <i>addParameter();</i>
		→ <i>addSetInfo();</i>
		→ <i>startBinding();</i>
		→ <i>startSet();</i>
*	가	→ <i>addRow();</i>
		→ <i>endSet();</i>
		→ <i>endBinding();</i>

:

- - 가

	가	→ <code>getDataModule();</code>
		→ <code>init();</code>
ODI	가	→ <code>addParameter();</code>
		→ <code>addSetInfo();</code>
		→ <code>startBinding();</code>
		→ <code>startSet();</code>
*		
	가	→ <code>addRow();</code>
*		→ <code>startSet();</code>
*		
*	가	→ <code>addRow();</code>
*		→ <code>endSet();</code>
		→ <code>endSet();</code>
		→ <code>endBinding();</code>

:

,

.

- 가

	가	→ <code>getDataModule();</code>
		→ <code>init();</code>
		→ <code>addSetInfo();</code>
		→ <code>startBinding();</code>
		→ <code>startSet();</code>
*	가	→ <code>addRow();</code>
		→ <code>endSet();</code>

→ **endBinding();**

:

.

FXDataModule

■ FX_DataModule

- FX_DataModule

Prototype	public FX_DataModule()
------------------	------------------------

Argument	FXDataModule .
-----------------	----------------

- addParameter

Prototype	public void addParameter(FX_Parameter param)
------------------	--

Definition	.
-------------------	---

Argument	<i>Param</i>
-----------------	--------------

- addParameters

Prototype	public void addParameters(FX_Parameter[] param)
------------------	---

Definition	.
-------------------	---

Argument	<i>Params</i>
-----------------	---------------

- addDataSetMeta

Prototype	public void addDataSetMeta(FX_DataSetMeta meta)
------------------	---

Definition	.
-------------------	---

Argument	<i>Meta</i>
-----------------	-------------

- write

Prototype	public void write(OutputStream out, boolean isSDM, boolean isCompress)
------------------	--

Definition	DataModule		SDM	XML	
	:	ContentType	XML	가	Response (Ex :
	"text/xml")				
	out	OutStream			
Argument	isSDM	SDM		(false	XML)
	isCompress		: true	GZip	Unzip
Return					
- sendErrorMessage					
Prototype	public void sendErrorMessage(OutputStream out, boolean isSDM, boolean isCompress, String error)				
Definition	DataModule		SDM	XML	,
	out	OutStream			
	isSDM	SDM		(false	XML)
Argument	isCompress		: true	GZip	Unzip
Return	error				
- getDataAction					
Prototype	public FX_DataAction getDataAction(HttpServletRequest request)				
Definition	Request	FX_DataAction			
Argument	request	Request			
	error				
Return	DataAction				

■ FX_DataSetMeta

- FX_DataSetMeta

Prototype	public FX_DataSetMeta(String name)
Definition	FXDataModule .
Argument	<i>name</i>

- FX_DataSetMeta

Prototype	public FX_DataSetMeta(String name, String mSetName)
Definition	FXDataModule .
Argument	<i>name</i> <i>mSetName</i>

- setMasterSetName

Prototype	public void setMasterSetName(String mSetName)
Definition	.
Argument	<i>mSetName</i>

- addDataFieldMeta

Prototype	public void addDataFieldMeta(FX_DataFieldMeta field)
Definition	가 .
Argument	<i>field</i> 가

- addDataFieldMeta

Prototype	public void addDataFieldMeta(FX_DataFieldMeta[] fields)
Definition	가 .
Argument	<i>fields</i> 가

■ FX_DataFieldMeta

- FX_DataFieldMeta

Prototype	public FX_DataFieldMeta(String name, int type)
Definition	.
Argument	<i>name</i> <i>type</i>

- FX_DataFieldMeta

Prototype	public FX_DataFieldMeta(String name)
Definition	VARCHAR .
Argument	<i>name</i>

- setType

Prototype	public void setType(int type)
Definition	.
Argument	<i>type</i>

■ FX_DataSet

- FX_DataSet

Prototype	public FX_DataSet(String name)
Definition	FXDataModule .
Argument	<i>name</i>

- addRecord

Prototype	public void addRecord(FX_Record record)
Definition	, 가 .
Argument	<i>record</i>

■ FX_Record

- FX_Record

Prototype	public FX_Record()
Definition	, .

- addColumn

Prototype	public void addColumn(String name, Object value)
Definition	가 .
Argument	<i>name</i> <i>value</i>

- addDetailDataSet

Prototype	public void addDetailDataSet(FX_DataSet detail)
Definition	가 .
Argument	<i>detail</i>

■ FX_DataAction

- FX_DataAction

Prototype	public FX_DataAction(String name, String actionType)
Definition	DataAcion . <i>name</i>
Argument	<i>actionType</i> DataAction

- FX_DataAction

Prototype	public FX_DataAction(String name, String actionType, String ext)
Definition	DataAcion . <i>name</i>
Argument	<i>actionType</i> DataAction <i>ext</i> ext

- getDataSetName

Prototype	public String getDataSetName()
Definition	.
Return	

- getActionType

Prototype	public String getActionType()
Definition	DataAction .
Return	DataAction (Insert, RowUpdate, Delete)

- getExt

Prototype	public String getExt()
Definition	ext .
Return	ext

- setExt

Prototype	public String setExt(String ext)
Definition	ext .
Argument	<i>ext</i> ext

- getSourceFields

Prototype	public FX_Parameter[] getSourceFields()
Definition	Source .
Return	Source

- setSourceFields

Prototype	public void setSourceFields(FX_Parameter[] fields)
Definition	Source .
Argument	<i>fields</i> Source

- getTargetFields

Prototype	public FX_Parameter[] getTargetFields()
Definition	Target .
Return	Target

- setTargetFields

Prototype	public void setTargetFields(FX_Parameter[] fields)
Definition	Target .
Argument	<i>fields</i> Target

■ FX_Parameter

- FX_Parameter

Prototype	public FX_Parameter(String name, int type, String value)
Definition	FXDataModule .
	<i>name</i>
Argument	<i>type</i>
	<i>value</i>

- FX_Parameter

Prototype	public FX_Parameter(String name, String value)
Definition	FXDataModule VARCHAR .
Argument	<i>name</i> <i>value</i>

- getName

Prototype	public String getName()
Definition	.
Return	

- getType

Prototype	public int getType()
Definition	.
Return	

- setType

Prototype	public void setType(int type)
Definition	.
Definition	<i>type</i>

- getValue

Prototype	public String getType()
Definition	.
Return	

- isNull

Prototype	public boolean isNull()
Definition	null .
Return	null

Servlet API (for OZ .Net Server)

: Servlet API Framework 1.1.4322 가

FXDataModule

■ FX_DataModule

- FX_DataModule

Prototype	public FX_DataModule()
------------------	------------------------

Argument	FXDataModule .
-----------------	----------------

- addParameter

Prototype	public void addParameter(FX_Parameter param)
------------------	--

Definition	.
-------------------	---

Argument	<i>Param</i>
-----------------	--------------

- addParameters

Prototype	public void addParameters(FX_Parameter[] param)
------------------	---

Definition	.
-------------------	---

Argument	<i>Params</i>
-----------------	---------------

- addDataSetMeta

Prototype	Public void addDataSetMeta(FX_DataSetMeta meta)
------------------	---

Definition	.
-------------------	---

Argument	<i>Meta</i>
-----------------	-------------

- write

Prototype	public void write(OutputStream output, boolean isSDM, boolean isCompress)
------------------	---

Definition	<code>DataModule</code>	<code>SDM</code>	<code>XML</code>	<code>.</code>
	<code>:</code>	<code>XML</code>	<code>Response</code>	<code>(Ex : "</code>
	<code>ContentType</code>	<code>가</code>	<code>.</code>	<code>)</code>
	<code>"text/xml")</code>			
Argument	<code>output</code>	<code>OutputStream</code>		
	<code>isSDM</code>	<code>SDM</code>	<code>(false</code>	<code>XML</code>
			<code>)</code>	
	<code>isCompress</code>	<code>: true</code>	<code>GZip</code>	<code>.</code>
			<code>Unzip</code>	
Return				

- `sendErrorMessage`

Prototype	<code>public void sendErrorMessage(OutputStream output, boolean isSDM, boolean isCompress, String error)</code>			
Definition	<code>DataModule</code>	<code>SDM</code>	<code>XML</code>	<code>,</code>
	<code>.</code>			
	<code>output</code>	<code>OutputStream</code>		
	<code>isSDM</code>	<code>SDM</code>	<code>(false</code>	<code>XML</code>
			<code>)</code>	
Argument	<code>isCompress</code>	<code>: true</code>	<code>GZip</code>	<code>.</code>
			<code>Unzip</code>	
	<code>error</code>			
Return				

- `getDataAction`

Prototype	<code>public FX_DataAction getDataAction(HttpServletRequest request)</code>			
Definition	<code>Request</code>	<code>FX_DataAction</code>	<code>.</code>	
	<code>request</code>	<code>Request</code>		
Argument	<code>error</code>			
Return	<code>DataAction</code>			

■ `FX_DataSetMeta`

- `FX_DataSetMeta`

Prototype	<code>public FX_DataSetMeta(String name)</code>			
------------------	---	--	--	--

Definition	FXDataModule
-------------------	--------------

Argument	<i>name</i>
-----------------	-------------

- FX_DataSetMeta

Prototype	public FX_DataSetMeta(String name, String mSetName)
------------------	---

Definition	FXDataModule
-------------------	--------------

	<i>name</i>
--	-------------

Argument	<i>mSetName</i>
-----------------	-----------------

- MasterSetName

Prototype	public string MasterSetName{get; set;}
------------------	--

Definition	
-------------------	--

- addDataFieldMeta

Prototype	public void addDataFieldMeta(FX_DataFieldMeta[] field)
------------------	--

Definition	가
-------------------	---

Argument	<i>field</i> 가
-----------------	----------------

■ FX_DataFieldMeta

- FX_DataFieldMeta

Prototype	public FX_DataFieldMeta(string name, int type)
------------------	--

Definition	
-------------------	--

	<i>name</i>
--	-------------

Argument	<i>type</i>
-----------------	-------------

- FX_DataFieldMeta

Prototype	public FX_DataFieldMeta(String name)
------------------	--------------------------------------

Definition	VARCHAR
-------------------	---------

Argument	<i>name</i>
-----------------	-------------

- Type

Prototype	public FX_DataTypes Type{get; set;}
------------------	-------------------------------------

Definition	
-------------------	--

■ FX_DataSet

- FX_DataSet

Prototype	public FX_DataSet(string name)
------------------	--------------------------------

Definition	FXDataModule
-------------------	--------------

Argument	<i>name</i>
-----------------	-------------

- addRecord

Prototype	public void AddRecord(FX_Record record)
------------------	---

Definition	record을 FX_DataSet에 추가합니다.
-------------------	----------------------------

Argument	<i>record</i>
-----------------	---------------

■ FX_Record

- FX_Record

Prototype	public FX_Record()
------------------	--------------------

Definition	FX_DataSet의 한 레코드입니다.
-------------------	-----------------------

- addColumn

Prototype	public void addColumn(string name, Object value)
------------------	--

Definition	FX_DataSet에 새로운 열을 추가합니다.
-------------------	---------------------------

Argument	<i>name</i>
-----------------	-------------

Argument	<i>value</i>
-----------------	--------------

- addDetailDataSet

Prototype	public void addDetailDataSet(FX_DataSet detail)
------------------	---

Definition	FX_DataSet에 세부 데이터를 추가합니다.
-------------------	----------------------------

Argument	<i>detail</i>
-----------------	---------------

■ FX_DataAction

- FX_DataAction

Prototype	public FX_DataAction(string name, string actionType)
------------------	--

Definition	DataAction 클래스를 상속받습니다.
-------------------	-------------------------

Argument	<i>name</i>
-----------------	-------------

	<i>actionType</i>	DataAction
--	-------------------	------------

- FX_DataAction

Prototype	public FX_DataAction(string name, string actionType, string ext)	
Definition	DataAccion	.
	<i>name</i>	
Argument	<i>actionType</i>	DataAction
	<i>ext</i>	ext

- DataSetName

Prototype	public string DataSetName{get;}	
Definition		.

- ActionType

Prototype	public string ActionType{get;}	
Definition	DataAction	. (Insert, RowUpdate, Delete)

- ExtraArgument

Prototype	public string ExtraArgument{get; set;}	
Definition	ext	.

- SourceFields

Prototype	public FX_Parameter[] SourceFields{get; set;}	
Definition	Source	.

- TargetFields

Prototype	public FX_Parameter[] TargetFields{get; set;}	
Definition	Target	.

- FX_Parameter

- FX_Parameter

Prototype	public FX_Parameter(string name, int type, string value)	
------------------	--	--

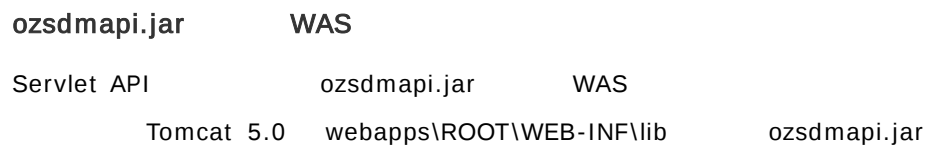
	Definition	FXDataModule	.
		<i>name</i>	
	Argument	<i>type</i>	
		<i>value</i>	
-	FX_Parameter		
	Prototype	public FX_Parameter(string name, string value)	
	Definition	FXDataModule	VARCHAR .
		<i>name</i>	
	Argument	<i>value</i>	
-	Name		
	Prototype	public string Name{get;}	
	Definition	.	
-	Type		
	Prototype	public FX_DataTypes Type{get; set;}	
	Definition	.	
-	value		
	Prototype	public string value{get;}	
	Definition	.	
-	IsNull		
	Prototype	public string getType{get; set;}	
	Definition	null .	

Servlet API

1 : Application -



Step 1



Step 2

```

package sample;

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

import java.util.*;

import oz.framework.api.DataModule;
import oz.uds.rs.ListMapResultSet;
import oz.sdm.DataModuleFactory;
import oz.uds.rs.ListMapResultSet;

/**
 * <p>Title: OZ SDM API</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: </p>
 * @author Forcs
 * @version 1.0
 */

public class DataModuleSampleServlet extends HttpServlet
{
    java
    DataModuleSampleServlet.class
  
```

```
private static final String _KEY_ODI_FETCH_TYPE = "_OZ_ODIFetchType_";
private static final String _KEY_ODIITEM       = "_OZ_ODIITEM_";
private static final String _KEY_ODICATEGORY   = "_OZ_ODICATEGORY_";
private static final String _KEY_OZ_DATASET_   = "_OZ_DATASET_";

private Connection m_conn = null;

public void init(ServletConfig config)
    throws ServletException
{
    super.init(config);

    System.out.println("init...");
}

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
{
    // init connection
    try {
        String _URL = "jdbc:odbc:ozdemokr30";
        Properties prop = new Properties();
        prop.put("user", "");
        prop.put("password", "");

        Driver driver = (Driver)
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        m_conn = driver.connect(_URL, prop);
        System.out.println("connection ok...");
    }
    catch (Exception e) {
        System.out.println("connection error...");
        DataModule.sendErrorMessage("connection error...", response.getOutputStream());

        if (m_conn != null) {
            try {
                m_conn.close();
            }
            catch (Exception e1) {
            }
        }

        throw new ServletException(e.getMessage());
    }

    try {
        System.out.println("start-----");
        Enumeration enum = request.getParameterNames();
        while (enum.hasMoreElements()) {
            String temp = (String) enum.nextElement();
            System.out.println("name=" + temp + " value=" +
                getEncode(request.getParameter(temp)));
        }
    }
```

```

// -----
// fetch parameters
// "DM_BATCH_FETCH|DM_CONCURRENT_FETCH"
String odiFetchType = getEncode(request.getParameter(_KEY_ODI_FETCH_TYPE));
String item = getEncode(request.getParameter(_KEY_ODI_ITEM));
String category = getEncode(request.getParameter(_KEY_ODI_CATEGORY));
String dataset = getEncode(request.getParameter(_KEY_OZ_DATASET));
// -----

if (odiFetchType == null) odiFetchType = "DM_CONCURRENT_FETCH";

if ( (dataset == null) || (dataset.length() == 0) ) {
    // ODI FetchUnit   DM_PER_DATAMODULE
    if (item.equals("JASMIN_SAMPLE3.odi")) {
        // Master-Detail   Sample
        Master_Detail_Style(request, response, odiFetchType);
    }
    else if (item.equals("JASMIN_SAMPLE2.odi") ||
             (item.equals("JASMIN_SAMPLE4.odi"))) {
        // SET   Sample
        default_Style(request, response, odiFetchType);
    }
    else {
        System.out.println("unsupported odi error...");
        DataModule.sendErrorMessage("unsupported odi error...",
                                    response.getOutputStream());
        throw new ServletException("Unknown Item :" + item);
    }
}
else {
    // ODI FetchUnit   DM_PER_DATASET
    //
    //   odiFetchType   DM_PER_DATASET
    DM_CONCURRENT_FETCH
    //
    //   (batch
    //   )
    if (odiFetchType.equalsIgnoreCase("DM_CONCURRENT_FETCH")) {
        // set   #1
        setDefaultStyle(request, response, "DM_PER_DATASET");
        // set   #2
        //
        //   setResultsetQueryStyle(request, response, "DM_CONCURRENT_FETCH");
    }
    else {
        System.out.println("unsupported error...");
        DataModule.sendErrorMessage("unsupported error...",
                                    response.getOutputStream());
        throw new ServletException("unsupported");
    }
}
}
catch(Exception ex){
    ex.printStackTrace();
    //
    //   sendErrorMessage
    //   throw   client?
    DataModule.sendErrorMessage(ex.getMessage(),
                                response.getOutputStream());
    throw new ServletException(ex.getMessage());
}finally{

    if (m_conn != null) {
        try {
            m_conn.close();
        }
        catch (Exception e) {
        }
    }
}

```

```
        }
        System.out.println("end-----");
    }

}

private void Master_Detail_Style(HttpServletRequest request, HttpServletResponse response,
String fetchType)
    throws ServletException, IOException
{
    Statement stmt1 = null;
    Statement stmt4 = null;
    ResultSet rs1 = null;
    ResultSet rs4 = null;

    DataModule module = null;

    // jasmin/JASMIN_SAMPLE1.odi
    // SET_1
    String[] _FN1 = {"FirstName" };
    int[] _FT1 = {java.sql.Types.VARCHAR };
    // SET_4
    String[] _FN4 = {"ContactId", "FirstName", "LastName", "Phone", "Fax", "Email",
"OrgUnitId", "UserName" };
    int[] _FT4 = {4, 12, 12, 12, 12, 12, 5, 12 };
    try {
        module = DataModuleFactory.getDataModule(fetchType);

        module.init(response.getOutputStream());

        // set SET Info.
        module.addSetInfo("SET_1", "", _FN1, _FT1);
        module.addSetInfo("SET_4", "SET_1", _FN4, _FT4);
    }
    catch(Exception ex) {
        throw new ServletException(ex.getMessage());
    }
    try {
        module.startBinding();

        String query1 = "select distinct FirstName from contact";

        module.startSet("SET_1");
        stmt1 = m_conn.createStatement();
        rs1 = stmt1.executeQuery(query1);
        while(rs1.next()) {
            String f1 = rs1.getString(_FN1[0]);
            HashMap map1 = new HashMap();
            map1.put(_FN1[0], f1);

            module.addRow("SET_1", map1);

            String query4 = "select * from contact where FirstName='" + f1 + "' order by
contactid";

            module.startSet("SET_4");
            stmt4 = m_conn.createStatement();
            rs4 = stmt4.executeQuery(query4);
            while(rs4.next()) {
                HashMap map4 = new HashMap();
                map4.put(_FN4[0], new Integer(rs4.getInt(_FN4[0])));
                map4.put(_FN4[1], rs4.getString(_FN4[1]));
                map4.put(_FN4[2], rs4.getString(_FN4[2]));
                map4.put(_FN4[3], rs4.getString(_FN4[3]));
                map4.put(_FN4[4], rs4.getString(_FN4[4]));
```

```
        map4.put(_FN4[5], rs4.getString(_FN4[5]));
        map4.put(_FN4[6], new Integer(rs4.getInt(_FN4[6])));
        map4.put(_FN4[7], rs4.getString(_FN4[7]));

        module.addRow("SET_4", map4);
    }
    rs4.close(); rs4 = null;
    stmt4.close(); stmt4 = null;

    module.endSet("SET_4");
}

rs1.close(); rs1 = null;
stmt1.close(); stmt1 = null;

module.endSet("SET_1");

module.endBinding();
}
catch(Exception e) {
    e.printStackTrace();
    // sendBindErrorMessage
    if(module != null) {
        module.sendBindErrorMessage(e.toString());
    }
}
finally {
    if(rs1 != null) {
        try { rs1.close(); } catch(Exception e) {}
    }
    if(stmt1 != null) {
        try { stmt1.close(); } catch(Exception e) {}
    }
    if(rs4 != null) {
        try { rs4.close(); } catch(Exception e) {}
    }
    if(stmt4 != null) {
        try { stmt4.close(); } catch(Exception e) {}
    }
}
}

private void default_Style(HttpServletRequest request, HttpServletResponse response, String
fetchType)
throws ServletException, IOException
{
    System.out.println("select time==" + System.currentTimeMillis());
    Statement stmt = null;
    ResultSet rs = null;

    DataModule module = null;

    // jasmin/JASMIN_SAMPLE2.odi
    // SET_1
    String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName",
        "CarImageFile"};
    int[] _FT = {
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR
    };

    try {
        module = DataModuleFactory.getDataModule(fetchType);
```

```
module.init(response.getOutputStream());
//      module.init(output);

// set Parameter Info.
module.addParameter("PARAM1", java.sql.Types.VARCHAR,
                    getEncode(request.getParameter("PARAM1")));
module.addParameter("PARAM2", java.sql.Types.VARCHAR,
                    getEncode(request.getParameter("PARAM2")));

// set SET Info.
module.addSetInfo("SET_1", "", _FN, _FT);
} catch (Exception ex) {
    throw new ServletException(ex.getMessage());
}

try {
    module.startBinding();

    String query = "select * from car";

    module.startSet("SET_1");
    stmt = m_conn.createStatement();
    rs = stmt.executeQuery(query);
    while(rs.next()) {
        HashMap map = new HashMap();
        String a = rs.getString(_FN[0]);
        String b = rs.getString(_FN[1]);
        String c = rs.getString(_FN[2]);
        String d = rs.getString(_FN[3]);
        String e = rs.getString(_FN[4]);
        String f = rs.getString(_FN[5]);

        map.put(_FN[0], a);
        map.put(_FN[1], b);
        map.put(_FN[2], c);
        map.put(_FN[3], d);
        map.put(_FN[4], e);
        map.put(_FN[5], f);

        module.addRow("SET_1", map);
        System.out.print("select FN0="+_FN[0]+" name="+a);
        System.out.print("    FN1="+_FN[1]+" name="+b);
        System.out.print("    FN2="+_FN[2]+" name="+c);
        System.out.print("    FN3="+_FN[3]+" name="+d);
        System.out.print("    FN4="+_FN[4]+" name="+e);
        System.out.print("    FN5="+_FN[5]+" name="+f);
        System.out.println("");
    }
    module.endSet("SET_1");
    module.endBinding();

    rs.close(); rs = null;
    stmt.close(); stmt = null;
}
catch (Exception e) {
    e.printStackTrace();
    //      sendBindErrorMessage
    module.sendBindErrorMessage(e.toString());
}
finally {
    OutputStream out = response.getOutputStream();
```

```
        if(rs != null) {
            try { rs.close(); } catch(Exception e) {}
        }
        if(stmt != null) {
            try { stmt.close(); } catch(Exception e) {}
        }
    }
}

private void setDefaultStyle(HttpServletRequest request, HttpServletResponse response, String
fetchType)
    throws ServletException, IOException
{
    System.out.println("setDefaultStyle fetchType =="+fetchType);
    Statement stmt = null;
    ResultSet rs = null;

    DataModule module = null;

    // jasmin/JASMIN_SAMPLE4.odi
    // SET_1
    String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName",
        "CarImageFile"};
    int[] _FT = {
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR
    };

    try {
        module = DataModuleFactory.getDataModule(fetchType);
        module.init(response.getOutputStream());

        // set SET Info.
        module.addSetInfo("SET_1", _FN, _FT);
    }catch(Exception ex){
        throw new ServletException(ex.getMessage());
    }

    try {
        module.startBinding();

        String query = "select * from car";

        module.startSet("SET_1");
        stmt = m_conn.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()) {
            HashMap map = new HashMap();
            String a = rs.getString(_FN[0]);
            String b = rs.getString(_FN[1]);
            String c = rs.getString(_FN[2]);
            String d = rs.getString(_FN[3]);
            String e = rs.getString(_FN[4]);
            String f = rs.getString(_FN[5]);

            map.put(_FN[0], a);
            map.put(_FN[1], b);
            map.put(_FN[2], c);
            map.put(_FN[3], d);
            map.put(_FN[4], e);
            map.put(_FN[5], f);
        }
    }
}
```

```
        module.addRow("SET_1", map);
        System.out.print("select FN0="+_FN[0]+" name="+a);
        System.out.print("    FN1="+_FN[1]+" name="+b);
        System.out.print("    FN2="+_FN[2]+" name="+c);
        System.out.print("    FN3="+_FN[3]+" name="+d);
        System.out.print("    FN4="+_FN[4]+" name="+e);
        System.out.print("    FN5="+_FN[5]+" name="+f);
        System.out.println("");

    }
    module.endSet("SET_1");
    module.endBinding();

    rs.close(); rs = null;
    stmt.close(); stmt = null;

}
catch(Exception e) {
    e.printStackTrace();
    //                sendBindErrorMessage
    module.sendBindErrorMessage(e.toString());
}
finally {
    OutputStream out = response.getOutputStream();

    if(rs != null) {
        try { rs.close(); } catch(Exception e) {}
    }
    if(stmt != null) {
        try { stmt.close(); } catch(Exception e) {}
    }
}
}

private void setResultSetQueryStyle(HttpServletRequest request, HttpServletResponse response,
String fetchType)
    throws ServletException, IOException
{
    System.out.println("setResultSetQueryStyle...");
    Statement stmt = null;
    ResultSet rs = null;

    DataModule module = null;
    try {
        // jasmin/JASMIN_SAMPLE4.odi
        // SET_1
        String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName",
"CarImageFile" };
        int[] _FT = {java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR,java.sql.Types.VARCHAR};

        module = DataModuleFactory.getDataModule(fetchType);

        String query = "select * from car";

        stmt = m_conn.createStatement();
        rs = stmt.executeQuery(query);

        module.makeSDM_SET("SET_1",rs,response.getOutputStream());

        rs.close(); rs = null;
        stmt.close(); stmt = null;
    }
}
```

```
}
catch(Exception e) {
    e.printStackTrace();
    // set          sendBindErrorMessage
    //          makeSDM_SET          7}
    throw new ServletException(e.getMessage());
}
finally {
    if(rs != null) {
        try { rs.close(); } catch(Exception e) {}
    }
    if(stmt != null) {
        try { stmt.close(); } catch(Exception e) {}
    }
}
}

public void destroy()
{
    super.destroy();
}

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

Step 3

DataModuleSampleServlet.class WAS

➤ DataModuleSampleServlet.class

Tomcat 5.0 webapps\ROOT\WEB-INF\classes
sample

➤ DataModuleSampleServlet.class WAS

Tomcat 5.0 webapps\ROOT\WEB-INF web.xml

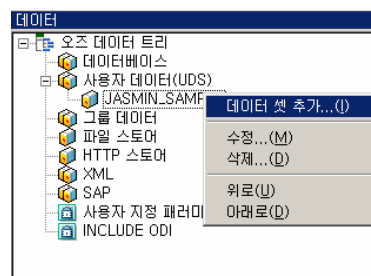
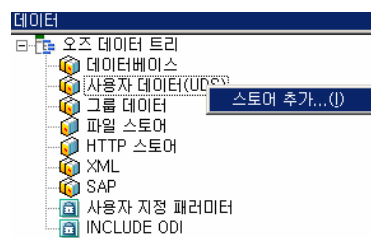
```
...

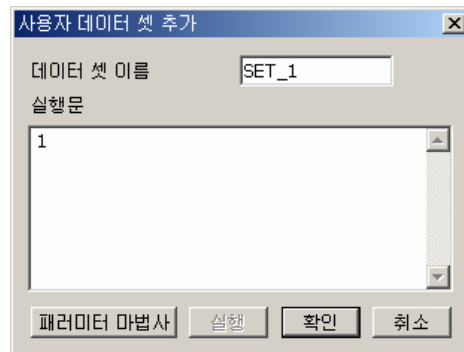
<!-- JSPC servlet mappings start -->
...
<servlet>
    <servlet-name>sample.DataModuleSampleServlet</servlet-name>
    <servlet-class>sample.DataModuleSampleServlet</servlet-class>
</servlet>
```

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

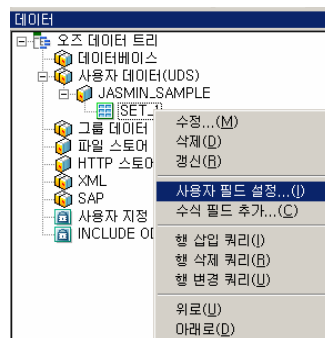
Step 4 ODI

Servlet API (UDS) 가 . 가 POST " _OZ_ODIITEM_" 가 . 가 (UDS) 가

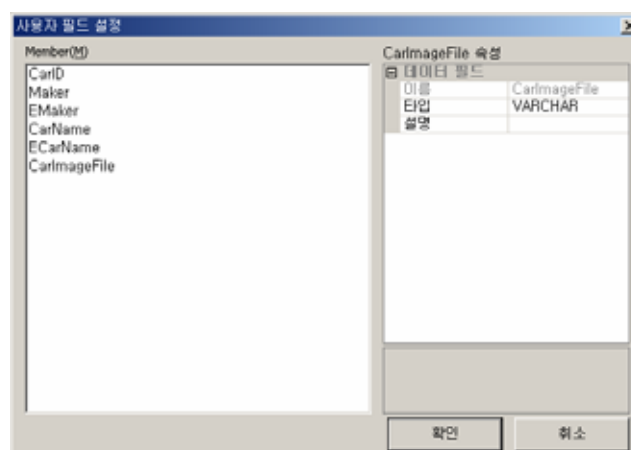




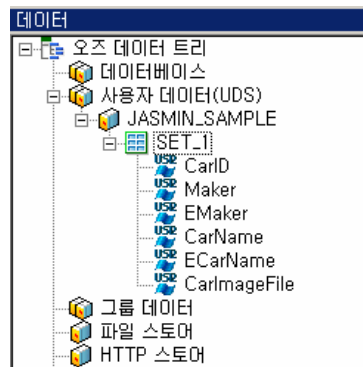
➤ 가 . 가
[]



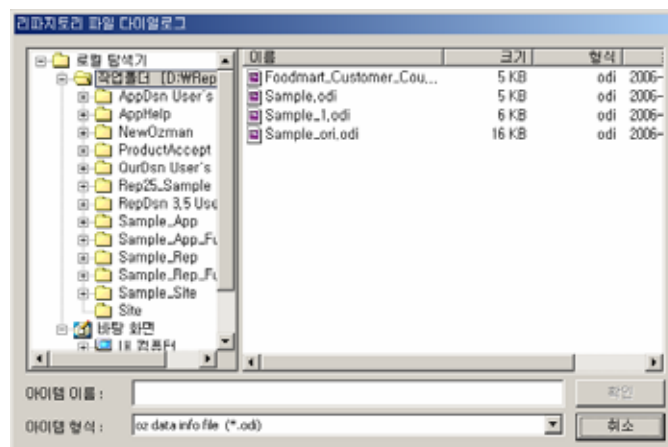
➤ 가 []



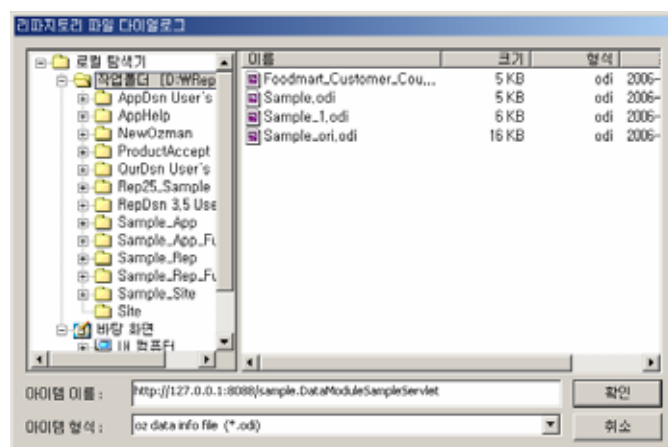
가 가 .



가



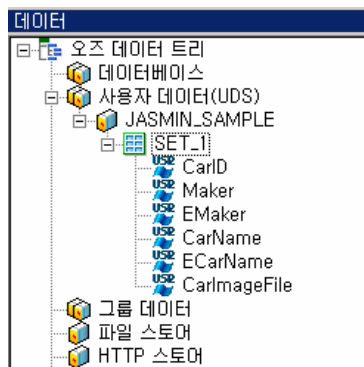
URL []



➤ "_OZ_ODIITEM_=ODI" []



➤ 가 가



➤ ODI "JASMIN_SAMPLE2.odi"

Step 5 OZF

: Servlet API 가
URL DataModule
RegisterUserDataModule JavaScript
. JavaScript
, Document.GlobalFunction OZF
OZF 가
OZF 가

➤ OZF

"MyFrameworkURLUDS.ozf"

```
MyFrameworkURLUDS.prototype.GetFrameworkURL =
    MyFrameworkURLUDS_GetFrameworkURL;
MyFrameworkURLUDS.prototype.GetCUDFrameworkURL =
```

```

        MyFrameworkURLUDS_GetCUDAFrameworkURL;
MyFrameworkURLUDS.prototype.GetFrameworkPostParam =
        MyFrameworkURLUDS_GetFrameworkPostParam;
MyFrameworkURLUDS.prototype.GetCUDAFrameworkPostParam =
        MyFrameworkURLUDS_GetCUDAFrameworkPostParam;

function MyFrameworkURLUDS(_url, _cud_url, _url_param, _cud_url_param){
    this.url = _url;
    this.cud_url = _cud_url;
    this.url_param = _url_param;
    if(this.url_param == null){
        this.url_param = "";
    }
    this.cud_url_param = _cud_url_param;
    if(this.cud_url_param == null){
        this.cud_url_param = "";
    }
}

function MyFrameworkURLUDS_GetFrameworkURL(dataset_name){
    return this.url;
}

function MyFrameworkURLUDS_GetCUDAFrameworkURL(dataset_name){
    return this.cud_url;
}

function MyFrameworkURLUDS_GetFrameworkPostParam(dataset_name){
    return "default&"+this.url_param;
}

function MyFrameworkURLUDS_GetCUDAFrameworkPostParam(dataset_name){
    return "default&"+this.cud_url_param;
}

```

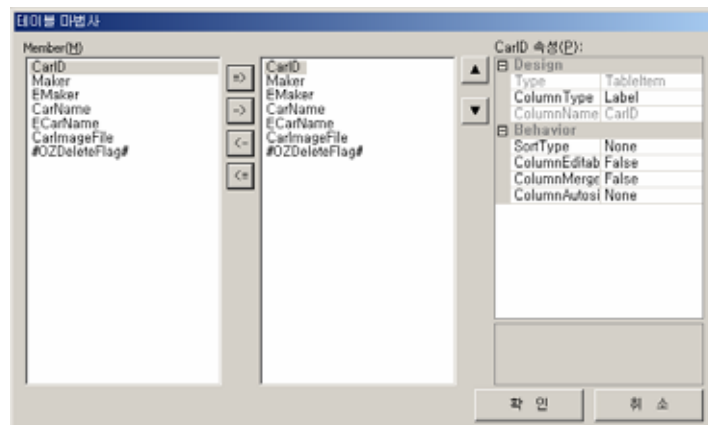
```

        GetFrameworkURL    GetCUDAFrameworkURL    가
        .
        GetFrameworkURL    가
        URL
        GetCUDAFrameworkURL    DataAction
        URL
        URL

```

URL

- "JASMIN_SAMPLE2.odi" "MyFrameworkURLUDS.ozf" 가
- Board Table 가 Table ODIKey "JASMIN_SAMPLE2" , DataSet "SET_1"



- Table OnInitialize
- ```
var uds = new
MyFrameworkURLUDS("http://127.0.0.1:8088/sample.DataModuleSampleServlet","http://127.0
.0.1:8088/sample.DataModuleSampleServlet");
var datamanager = _GetDataManager();
var datamodule = datamanager.GetDataModule("JASMIN_SAMPLE2");
datamodule.RegisterUserDataModule(uds);
```

**Step 6**

WAS



|    | CarID | Maker | EMaker  | CarName | ECarName  | CarImageFile                   |
|----|-------|-------|---------|---------|-----------|--------------------------------|
| 1  | H04   | 현대자동차 | Hyundai | EF소나타   | EFSONATA  | http://127.0.0.1/img/EF소나타.gif |
| 2  | H02   | 현대자동차 | Hyundai | 다이네스티   | DYNASTY   | http://127.0.0.1/img/다이네스티.gif |
| 3  | H03   | 현대자동차 | Hyundai | 그랜저     | GRANDEUR  | http://127.0.0.1/img/그랜저.gif   |
| 4  | H01   | 현대자동차 | Hyundai | 에쿠우스    | EQUUS     | http://127.0.0.1/img/에쿠우스.gif  |
| 5  | H05   | 현대자동차 | Hyundai | 베르나     | VERNA     | http://127.0.0.1/img/베르나.gif   |
| 6  | H06   | 현대자동차 | Hyundai | 아토스     | ATOZ      | http://127.0.0.1/img/아토스.gif   |
| 7  | D01   | 대우자동차 | DaeWoo  | 라노스     | RANOS     | http://127.0.0.1/img/라노스.gif   |
| 8  | D02   | 대우자동차 | DaeWoo  | 누비라     | NUBIRA    | http://127.0.0.1/img/누비라.gif   |
| 9  | D03   | 대우자동차 | DaeWoo  | 매그너스    | MAGNUS    | http://127.0.0.1/img/매그너스.gif  |
| 10 | K01   | 기아자동차 | KIA     | 세피아 I   | SEPHIA    | http://127.0.0.1/img/세피아.jpg   |
| 11 | K03   | 기아자동차 | KIA     | 세피아 II  | SEPHIA II | http://127.0.0.1/img/세피아2.jpg  |

## 2 : Application - DataAction

Servlet API Table , ,

### Step 1

ozsdmap.jar WAS

Servlet API ozsdmap.jar WAS  
Tomcat 5.0 webapps\ROOT\WEB-INF\lib ozsdmap.jar

### Step 2

DataAction

DataAction java

DataActionSampleServlet.class

```
package sample;

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

import java.util.*;

/**
 * <p>Title: OZ SDM API</p>
 */
```

```
* <p>Description: </p>
* <p>Copyright: Copyright (c) 2005</p>
* <p>Company: </p>
* @author Forcs
* @version 1.0
*/

public class DataActionSampleServlet extends HttpServlet
{
 private static final String _KEY_ODI_FETCH_TYPE = "_OZ_ODIFetchType_";
 private static final String _KEY_ODIITEM = "_OZ_ODIITEM_";
 private static final String _KEY_ODICATEGORY = "_OZ_ODICATEGORY_";
 private static final String _KEY_OZ_DATASET = "_OZ_DATASET_";

 private static final String _KEY_OZ_DAC_CNT = "_OZ_DAC_CNT";

 private static String TABLENAME = "car";

 private String dac_insert_query = "";
 private String dac_update_query = "";
 private String dac_delete_query = "";

 private PreparedStatement p_insert_stmt = null;
 private PreparedStatement p_update_stmt = null;
 private PreparedStatement p_delete_stmt = null;

 private Connection m_conn = null;

 public void init(ServletConfig config)
 throws ServletException
 {
 super.init(config);
 }

 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
 {
 try {
 String _URL = "jdbc:odbc:ozdemokr30";
 Properties prop = new Properties();
 prop.put("user", "");
 prop.put("password", "");

 Driver driver =
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
 m_conn = driver.connect(_URL, prop);
 }
 catch (Exception ex) {
 response.getOutputStream().write(ex.getMessage().getBytes());
 throw new ServletException(ex.getMessage());
 }
 }
}
```

```
try {
 dac_insert_query = "";
 dac_update_query = "";
 dac_delete_query = "";

 Enumeration enum = request.getParameterNames();
 while (enum.hasMoreElements()) {
 String temp = (String) enum.nextElement();
 System.out.println("name=" + temp + " value=" +
 getEncode(request.getParameter(temp)));
 }

 // -----
 // fetch parameters
 // "DM_BATCH_FETCH|DM_CONCURRENT_FETCH"
 String odiFetchType = getEncode(request.getParameter(
 _KEY_ODI_FETCH_TYPE));
 String item = getEncode(request.getParameter(_KEY_ODI_ITEM));
 String category = getEncode(request.getParameter(_KEY_ODI_CATEGORY));
 String dataset = getEncode(request.getParameter(_KEY_OZ_DATASET_));
 // -----
 System.out.println("-----");
 if (odiFetchType == null) odiFetchType = "DM_CONCURRENT_FETCH";

 m_conn.setAutoCommit(false);

 // Action
 int dac_cnt = Integer.parseInt(request.getParameter(_KEY_OZ_DAC_CNT));

 for (int i = 0; i < dac_cnt; i++) {
 String type = request.getParameter(i + ".TYPE");
 if (type.equalsIgnoreCase("insert")) {
 insert(request, response, i);
 }
 else if (type.equalsIgnoreCase("rowupdate")) {
 update(request, response, i);
 }
 else if (type.equalsIgnoreCase("delete")) {
 delete(request, response, i);
 }
 System.out.println(type+" time==" + System.currentTimeMillis());
 }
 m_conn.commit();
 response.getOutputStream().write(new String("OK").getBytes());
} catch (Exception ex) {
 ex.printStackTrace();
 try { m_conn.rollback(); } catch (Exception e) {}
 response.getOutputStream().write(ex.getMessage().getBytes());
 throw new ServletException(ex.getMessage());
} finally {
 if (p_insert_stmt != null) {
 try { p_insert_stmt.close(); } catch (Exception e) {}
 }
 if (p_update_stmt != null) {
 try { p_update_stmt.close(); } catch (Exception e) {}
 }
 if (p_delete_stmt != null) {
 try { p_delete_stmt.close(); } catch (Exception e) {}
 }

 if (m_conn != null) {
 try { m_conn.close(); } catch (Exception e) {}
 }
}
```

```
 }
}

private void insert(HttpServletRequest request, HttpServletResponse response, int cnt)
 throws Exception
{
 int insert_source_fieldCnt = Integer.parseInt(request.getParameter(cnt + ".SRC_CNT"));

 if (dac_insert_query.equalsIgnoreCase("")) {
 // prepared
 dac_insert_query = "INSERT INTO "+TABLENAME+" (";
 for (int i = 0; i < insert_source_fieldCnt; i++) {
 String s_fieldName = getEncode(request.getParameter(cnt + ".SF_" + i));
 if (i == insert_source_fieldCnt - 1) {
 dac_insert_query += s_fieldName;
 }
 else {
 dac_insert_query += s_fieldName + ",";
 }
 }
 dac_insert_query += ") VALUES (";

 for (int i = 0; i < insert_source_fieldCnt; i++) {
 if (i == insert_source_fieldCnt - 1) {
 dac_insert_query += "?";
 }
 else {
 dac_insert_query += "? ,";
 }
 }

 dac_insert_query += ")";
 System.out.println("dac_insert_query="+dac_insert_query);
 p_insert_stmt = m_conn.prepareStatement(dac_insert_query);
 }

 for (int i = 0; i < insert_source_fieldCnt; i++) {
 String value = getEncode(request.getParameter(cnt + ".SV_" + i));
 p_insert_stmt.setString(i + 1, value);
 }

 p_insert_stmt.execute();
}

private void update(HttpServletRequest request, HttpServletResponse response, int cnt)
 throws Exception
{
 int update_source_fieldCnt = Integer.parseInt(request.getParameter(cnt + ".SRC_CNT"));
 int update_target_fieldCnt = 0;

 try {
 update_target_fieldCnt = Integer.parseInt(request.getParameter(cnt + ".TRG_CNT"));
 } catch (Exception ex) {}

 if (dac_update_query.equalsIgnoreCase("")) {
 // prepared
 dac_update_query = "UPDATE "+TABLENAME+" set ";

 for (int i = 0; i < update_source_fieldCnt; i++) {
 String s_fieldName = getEncode(request.getParameter(cnt + ".SF_" + i));

 if (i == update_source_fieldCnt - 1) {
```

```
 dac_update_query += s_fieldName + " = ? ";
 }
 else {
 dac_update_query += s_fieldName + " = ?, ";
 }
}
dac_update_query += " WHERE ";

for (int i = 0; i < update_target_fieldCnt; i++) {
 String t_fieldName = getEncode(request.getParameter(cnt + ".DF_" + i));

 if (i == update_target_fieldCnt - 1) {
 dac_update_query += t_fieldName + " = ? ";
 }
 else {
 dac_update_query += t_fieldName + " = ? AND ";
 }
}

System.out.println("dac_update_query="+dac_update_query);
p_update_stmt = m_conn.prepareStatement(dac_update_query);
}

int i = 0;
for (i = 0; i < update_source_fieldCnt; i++) {
 String value = getEncode(request.getParameter(cnt + ".SV_" + i));
 p_update_stmt.setString(i + 1, value);
}

for (int j = 0; j < update_source_fieldCnt; j++) {
 String value = getEncode(request.getParameter(cnt + ".DV_" + j));
 p_update_stmt.setString(i + 1, value);
 i++;
}
p_update_stmt.execute();
}

private void delete(HttpServletRequest request, HttpServletResponse response, int cnt)
 throws Exception
{
 int delete_target_fieldCnt = Integer.parseInt(request.getParameter(cnt + ".TRG_CNT"));

 if (dac_delete_query.equalsIgnoreCase("")) {
 // prepared
 dac_delete_query = "DELETE FROM "+TABLENAME+" WHERE ";

 for (int i = 0; i < delete_target_fieldCnt; i++) {
 String t_fieldName = getEncode(request.getParameter(cnt + ".DF_" + i));

 if (i == delete_target_fieldCnt - 1) {
 dac_delete_query += t_fieldName + " = ? ";
 }
 else {
 dac_delete_query += t_fieldName + " = ? AND ";
 }
 }

 System.out.println("dac_delete_query="+dac_delete_query);
 p_delete_stmt = m_conn.prepareStatement(dac_delete_query);
 }
}
```

```

 int i = 0;
 for (i = 0; i < delete_target_fieldCnt; i++) {
 String value = getEncode(request.getParameter(cnt + ".DV_" + i));
 p_delete_stmt.setString(i + 1, value);
 }

 p_delete_stmt.execute();

 }

 public void destroy()
 {
 super.destroy();
 }

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

```

**Step 3**

- " 1" DataModuleSampleServlet.class
- DataActionSampleServlet.class WAS .
- DataModuleSampleServlet.class, DataActionSampleServlet.class
- WAS . Tomcat 5.0
- webapps\ROOT\WEB-INF\classes sample
- DataModuleSampleServlet.class, DataActionSampleServlet.class
- WAS . Tomcat 5.0 webapps\ROOT\WEB-INF
- web.xml

```

...

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

<servlet>
 <servlet-name>sample.DataActionSampleServlet</servlet-name>

```

```

 <servlet-class>sample.DataActionSampleServlet</servlet-class>
 </servlet>

 ...

 <servlet-mapping>
 <servlet-name>sample.DataModuleSampleServlet</servlet-name>
 <url-pattern>/sample.DataModuleSampleServlet</url-pattern>
 </servlet-mapping>

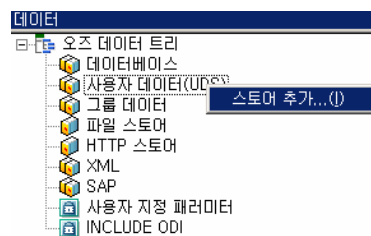
 <servlet-mapping>
 <servlet-name>sample.DataActionSampleServlet</servlet-name>
 <url-pattern>/sample.DataActionSampleServlet</url-pattern>
 </servlet-mapping>

 ...
 <!-- JSPC servlet mappings end -->

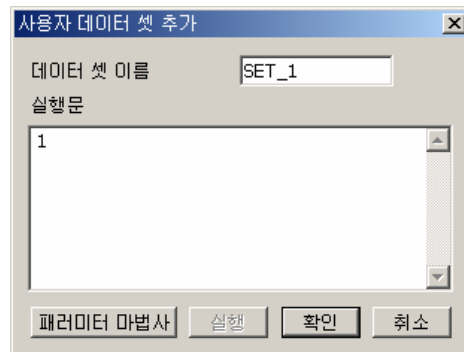
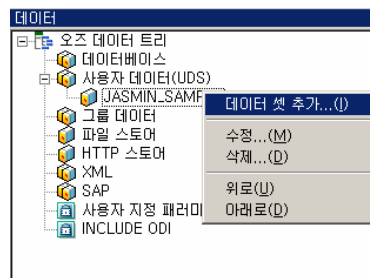
```

#### Step 4 ODI

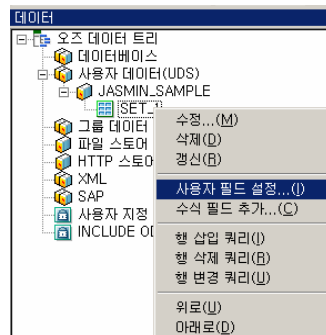
Servlet API (UDS) ,  
가 .  
> (UDS) 가



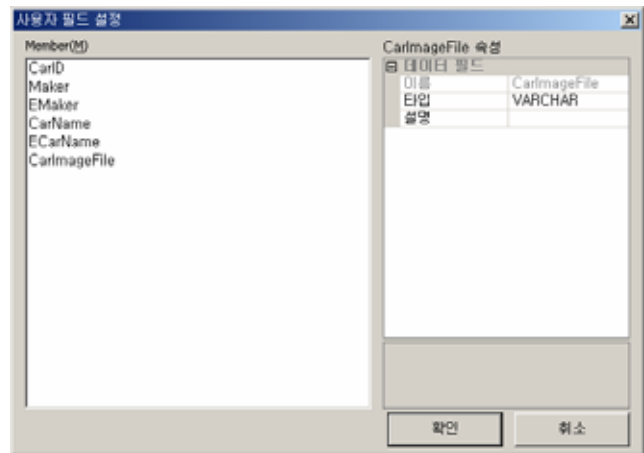
> 가 가 .



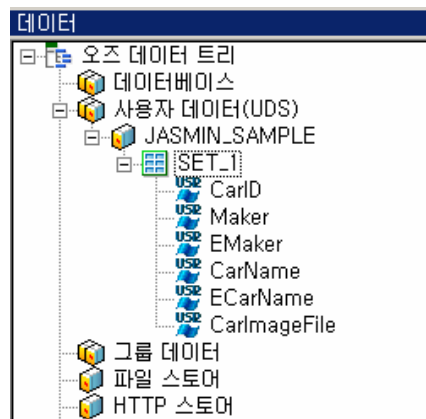
➤ 가 . 가  
[ ]



➤ 가 [ ]



가 가 .



➤ SET\_1 [ ], [ ],  
[ ] , ,

```
INSERT INTO car(##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#')
```

```
DELETE FROM car WHERE [##ARG_DF1#] = '##ARG_DV1#';
```

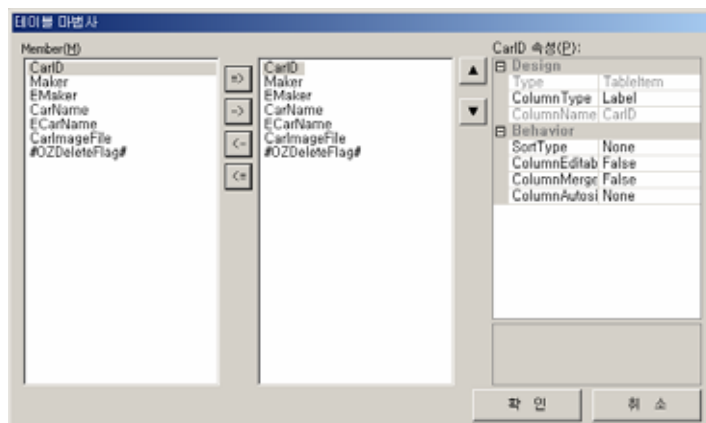
```
UPDATE car SET
[#@ARG_SF1#] = '#@ARG_SV1#', [#@ARG_SF2#] = '#@ARG_SV2#',
[#@ARG_SF3#] = '#@ARG_SV3#', [#@ARG_SF4#] = '#@ARG_SV4#',
[#@ARG_SF5#] = '#@ARG_SV5#', [#@ARG_SF6#] = '#@ARG_SV6#'
WHERE [#@ARG_DF1#] = '#@ARG_DV1#'
```

- ODI "JASMIN\_SAMPLE4.odi"

### Step 5

- "JASMIN\_SAMPLE4.odi" "1" "MyFrameworkURLUDS.ozf" 가 .

- Board Table 가 Table ODIKey "JASMIN\_SAMPLE4" , DataSet "SET\_1"



- Table , Table "AllowInsert", "AllowDelete", "AllowUpdate" "True"

- Table OnInitialize

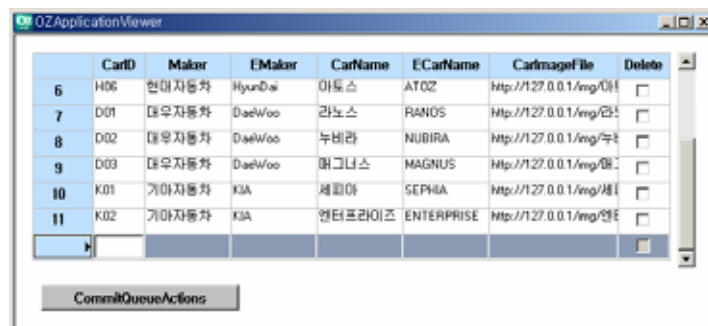
```
var uds = new
MyFrameworkURLUDS("http://127.0.0.1:8088/sample.DataModuleSampleServlet","http://127.0.0.1:8088/sample.DataActionSampleServlet");
var datamanager = _GetDataManager();
var datamodule = datamanager.GetDataModule("JASMIN_SAMPLE4");
datamodule.RegisterUserDataModule(uds);
```

➤ Board Button 가 Button OnClick

```
var result = Table1.CommitQueuedActions();
if(result == "") {
 Table1.GetDataModule().RefreshAllDataSet();
} else {
 _MessageBox(result);
}

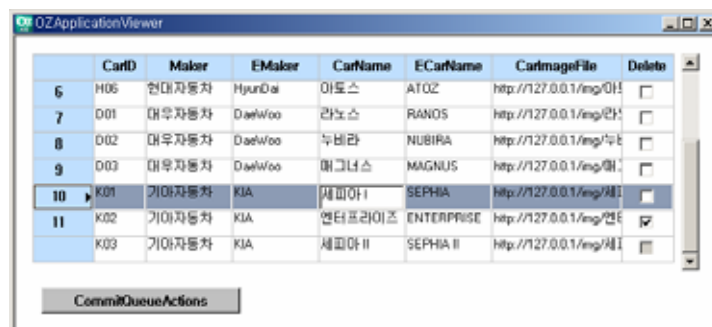
//var myODIObject =
 _GetDataManager().GetDataModule("odiName").GetUserDataModule();
//Select
 가
//
 myODIObject.url_param = "key1=value1&key2=value2";
//DataAction
 가
//
 myODIObject.cud_url_param = "key1=value1&key2=value2";
//_GetDataManager().GetDataModule("odiName").RefreshAllDataSet();
```

Step 7



가

Delete



DataAction

[CommitQueueActions]

|    | CarID | Maker | EMaker  | CarName | ECarName  | CarImageFile            | Delete                   |
|----|-------|-------|---------|---------|-----------|-------------------------|--------------------------|
| 6  | H05   | 현대자동차 | Hyundai | 아토스     | ATOZ      | http://127.0.0.1/img/0H | <input type="checkbox"/> |
| 7  | D01   | 대우자동차 | DaeWoo  | 라노스     | RANOS     | http://127.0.0.1/img/2라 | <input type="checkbox"/> |
| 8  | D02   | 대우자동차 | DaeWoo  | 누비라     | NUBIRA    | http://127.0.0.1/img/누비 | <input type="checkbox"/> |
| 9  | D03   | 대우자동차 | DaeWoo  | 매그니스    | MAGNIUS   | http://127.0.0.1/img/매  | <input type="checkbox"/> |
| 10 | K01   | 기아자동차 | KIA     | 세피아 I   | SEPHIA    | http://127.0.0.1/img/세  | <input type="checkbox"/> |
| 11 | K03   | 기아자동차 | KIA     | 세피아 II  | SEPHIA II | http://127.0.0.1/img/세  | <input type="checkbox"/> |

CommitQueueActions

### 3 : Report -

Servlet API

. Servlet API

frameworkurl

(HTTP URL)

odi.odi .frameworkurl, odi.frameworkurl, connection.frameworkurl

: Servlet API

#### Step 1

ozsdmapi.jar WAS

Servlet API

ozsdmapi.jar

WAS

Tomcat 5.0

webapps\ROOT\WEB-INF\lib

ozsdmapi.jar

#### Step 2

" 1" DataModuleSampleServlet.class WAS

➤ DataModuleSampleServlet.class WAS

Tomcat 5.0 webapps\ROOT\WEB-INF\classes\sample

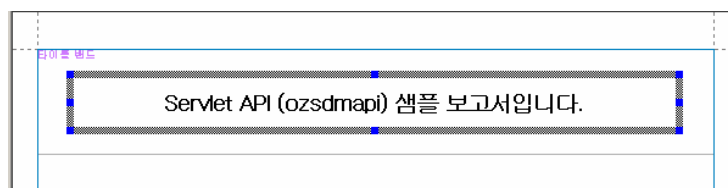
- DataModuleSampleServlet.class WAS  
Tomcat 5.0 webapps\ROOT\WEB-INF web.xml

```
...
<!-- 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 -->
```

**Step 3**

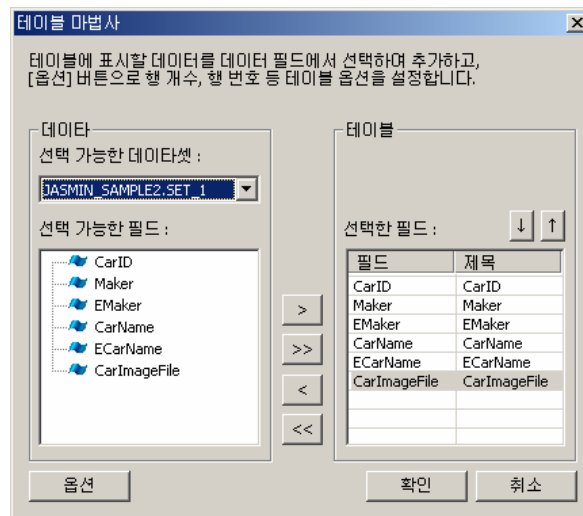
- " 1"  
JASMIN\_SAMPLE2.odi 가 .

- 가 , 가

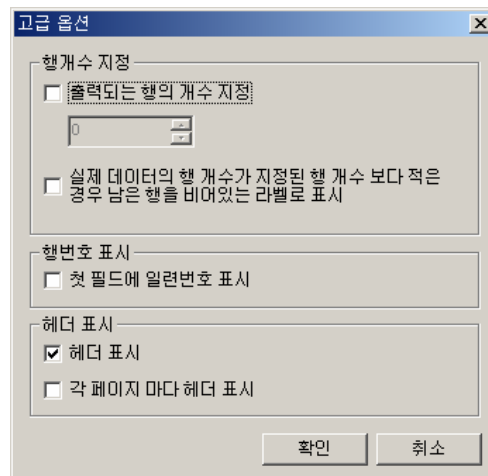


- 가 "ODI "  
"JASMIN\_SAMPLE2" , " " "SET\_1" .

- & 가 .  
가 .



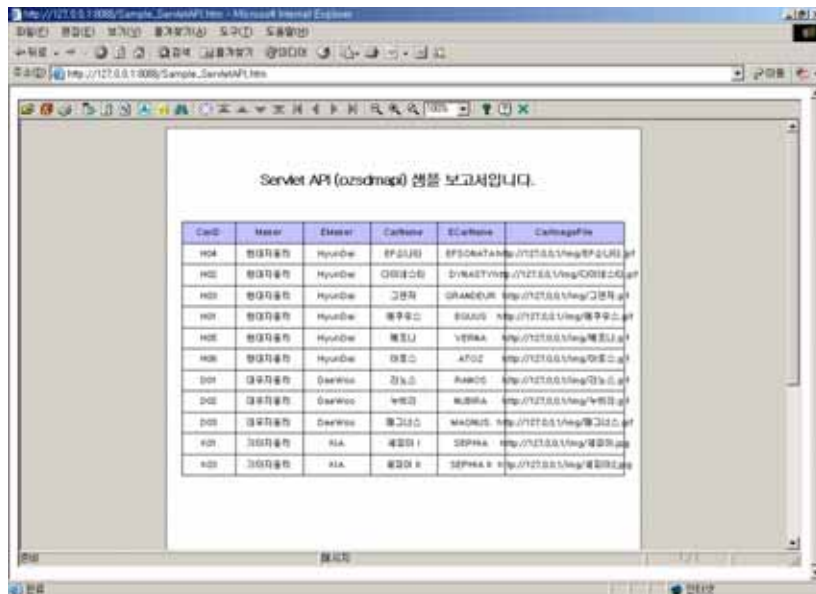
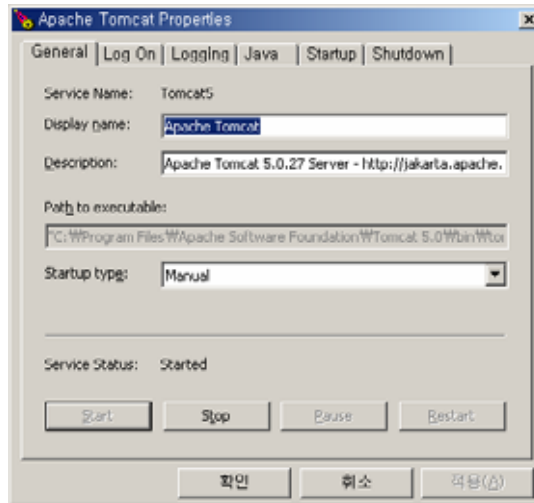
➤ [ ]  
" "



➤ 가 [ ]  
가

|                                   |               |                |                 |                  |                      |
|-----------------------------------|---------------|----------------|-----------------|------------------|----------------------|
| Servlet API (ozsdmapi) 샘플 보고서입니다. |               |                |                 |                  |                      |
| CarID                             | Maker         | EMaker         | CarName         | ECarName         | CarImageFile         |
| <SET_1:CarID>                     | <SET_1:Maker> | <SET_1:EMaker> | <SET_1:CarName> | <SET_1:ECarName> | <SET_1:CarImageFile> |





4 : XML SDM

Servlet API XML SDM  
Servlet API XML SDM ozsdmapi.jar,  
crimson.jar XML SDM WAS UDS  
ODI

Step 1 ozsdmapi.jar crimson.jar WAS

Servlet API ozsdmapi.jar, crimson.jar WAS  
Tomcat 5.0 webapps\ROOT\WEB-INF\lib  
ozsdmapi.jar, crimson.jar

Step 2 XML SDM

XML SDM  
java DebugDMSampleServlet.class  
package sample;  
  
import java.io.\*;  
import javax.servlet.\*;  
  
import javax.servlet.http.\*;  
import java.sql.\*;  
import java.util.\*;  
  
import oz.framework.api.DataModule;  
import oz.sdm.DataModuleFactory;  
  
public class DebugDMSampleServlet extends HttpServlet {  
 private static final String \_KEY\_ODI\_FETCH\_TYPE = "\_OZ\_ODIFetchType\_";  
 private static final String \_KEY\_ODIITEM = "\_OZ\_ODIITEM\_";  
 private static final String \_KEY\_OZ\_DATASET = "\_OZ\_DATASET\_";  
 private static final String \_KEY\_OZ\_DEBUG = "\_OZ\_DEBUG\_";  
  
 private static final String ODI\_NAME\_1 = "JASMIN\_SAMPLE2.odi";  
 private static final String ODI\_NAME\_2 = "JASMIN\_SAMPLE3.odi";  
 private static final String ODI\_NAME\_3 = "JASMIN\_SAMPLE4.odi";  
  
 private Connection m\_conn = null;  
  
 public void init(ServletConfig config) throws ServletException {  
 super.init(config);  
 System.out.println("init...");  
 }  
  
 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);
}

public void destroy() {
 super.destroy();
 System.out.println("destroy...");
}

private void process(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
 try {
 String _URL = "jdbc:odbc:ozdemokr30";
 Properties prop = new Properties();
 prop.put("user", "");
 prop.put("password", "");
 Driver driver =
(Driver)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
 m_conn = driver.connect(_URL, prop);
 System.out.println("connection ok...");
 } catch(Exception e) {
 System.out.println("connection error...");
 System.out.println(e.getMessage());
 DataModule.sendErrorMessage("connection error...",
response.getOutputStream());
 close(m_conn, null, null);
 throw new ServletException(e.getMessage());
 }

 try {
 System.out.println("start-----");
 Enumeration enum = request.getParameterNames();

 System.out.println("parameter list-----");
 while (enum.hasMoreElements()) {
 String temp = (String) enum.nextElement();
 System.out.println("name=" + temp + " value=" +
getEncode(request.getParameter(temp)));
 }
 System.out.println("-----");

 String odiFetchType =
getEncode(request.getParameter(_KEY_ODI_FETCH_TYPE));
 String item = getEncode(request.getParameter(_KEY_ODI_ITEM));
 String dataset = getEncode(request.getParameter(_KEY_OZ_DATASET));

 if (odiFetchType == null) {
 odiFetchType = "DM_CONCURRENT_FETCH";
 }

 if ((dataset == null) || (dataset.length() == 0)) {
 if (item.equals(ODI_NAME_2)) {
 Master_Detail_Style(request, response, odiFetchType);
 } else if (item.equals(ODI_NAME_1) || item.equals(ODI_NAME_3)) {
 default_Style(request, response, odiFetchType);
 } else {
 System.out.println("unsupported odi error...");
 DataModule.sendErrorMessage("unsupported odi error...",
response.getOutputStream());
 throw new ServletException("Unknown Item :" + item);
 }
 } else {
 if (odiFetchType.equalsIgnoreCase("DM_CONCURRENT_FETCH") ||
odiFetchType.equalsIgnoreCase("DM_PER_DATASET")) {
```

```
 setDefaultStyle(request, response, odiFetchType);
 //setResultSetQueryStyle(request, response, odiFetchType);
 } else {
 System.out.println("unsupported error...");
 DataModule.sendErrorMessage("unsupported error...",
response.getOutputStream());
 throw new ServletException("unsupported");
 }
}
} catch(Exception ex){
 System.out.println(ex.getMessage());
 DataModule.sendErrorMessage(ex.getMessage(), response.getOutputStream());
 throw new ServletException(ex.getMessage());
} finally {
 close(m_conn, null, null);
 System.out.println("end-----");
}
}

private void Master_Detail_Style(HttpServletRequest request, HttpServletResponse response,
String fetchType) throws ServletException, IOException {
 Statement stmt1 = null;
 Statement stmt4 = null;
 ResultSet rs1 = null;
 ResultSet rs4 = null;
 DataModule module = null;

 String[] _FN1 = {"FirstName"};
 int[] _FT1 = {java.sql.Types.VARCHAR};

 String[] _FN4 = {"ContactId", "FirstName", "LastName", "Phone", "Fax", "Email",
"OrgUnitId", "UserName" };
 int[] _FT4 = {4, 12, 12, 12, 12, 12, 5, 12 };
 try {
 module = DataModuleFactory.getDataModule(fetchType);

 // debug
 // init setDebug 7} true
 module.setDebug(parseBoolean(request), response);
 module.init(response.getOutputStream());

 module.addSetInfo("SET_1", "", _FN1, _FT1);
 module.addSetInfo("SET_4", "SET_1", _FN4, _FT4);
 } catch(Exception ex) {
 throw new ServletException(ex.getMessage());
 }

 try {
 module.startBinding();
 String query1 = "select distinct FirstName from contact";
 module.startSet("SET_1");

 stmt1 = m_conn.createStatement();
 rs1 = stmt1.executeQuery(query1);
 while(rs1.next()) {
 String f1 = rs1.getString(_FN1[0]);
 HashMap map1 = new HashMap();
 map1.put(_FN1[0], f1);
 module.addRow("SET_1", map1);

 String query4 = "select * from contact where FirstName='" + f1 + "'order by
contactid";

 module.startSet("SET_4");
 stmt4 = m_conn.createStatement();
 rs4 = stmt4.executeQuery(query4);
```

```
 while(rs4.next()) {
 HashMap map4 = new HashMap();
 map4.put(_FN4[0], new Integer(rs4.getInt(_FN4[0]]));
 map4.put(_FN4[1], rs4.getString(_FN4[1]));
 map4.put(_FN4[2], rs4.getString(_FN4[2]));
 map4.put(_FN4[3], rs4.getString(_FN4[3]));
 map4.put(_FN4[4], rs4.getString(_FN4[4]));
 map4.put(_FN4[5], rs4.getString(_FN4[5]));
 map4.put(_FN4[6], new Integer(rs4.getInt(_FN4[6]]));
 map4.put(_FN4[7], rs4.getString(_FN4[7]));
 module.addRow("SET_4", map4);
 }
 rs4.close();
 stmt4.close();
 module.endSet("SET_4");
 }
 module.endSet("SET_1");
 module.endBinding();
} catch(Exception e) {
 System.out.println(e.getMessage());
 if(module != null) {
 module.sendBindErrorMessage(e.toString());
 }
} finally {
 close(null, stmt1, rs1);
 close(null, stmt4, rs4);
}
}

private void default_Style(HttpServletRequest request, HttpServletResponse response, String
fetchType) throws ServletException, IOException {
 System.out.println("select time==" + System.currentTimeMillis());
 Statement stmt = null;
 ResultSet rs = null;
 DataModule module = null;

 String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName", "CarImageFile"};
 int[] _FT = {java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR, java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR};

 try {
 module = DataModuleFactory.getDataModule(fetchType);

 // debug
 // init
 module.setDebug(parseBoolean(request), response);
 module.init(response.getOutputStream());
 module.addParameter("PARAM1", java.sql.Types.VARCHAR,
getEncode(request.getParameter("PARAM1")));
 module.addParameter("PARAM2", java.sql.Types.VARCHAR,
getEncode(request.getParameter("PARAM2")));
 module.addSetInfo("SET_1", "", _FN, _FT);

 } catch(Exception ex){
 throw new ServletException(ex.getMessage());
 }

 try {
 module.startBinding();
 String query = "select * from car";
 module.startSet("SET_1");

 stmt = m_conn.createStatement();
 rs = stmt.executeQuery(query);
 }
```

```
 while(rs.next()) {
 HashMap map = new HashMap();
 String a = rs.getString(_FN[0]);
 String b = rs.getString(_FN[1]);
 String c = rs.getString(_FN[2]);
 String d = rs.getString(_FN[3]);
 String e = rs.getString(_FN[4]);
 String f = rs.getString(_FN[5]);

 map.put(_FN[0], a);
 map.put(_FN[1], b);
 map.put(_FN[2], c);
 map.put(_FN[3], d);
 map.put(_FN[4], e);
 map.put(_FN[5], f);

 module.addRow("SET_1", map);

 System.out.print("select FN0="+_FN[0]+" name="+a);
 System.out.print(" FN1="+_FN[1]+" name="+b);
 System.out.print(" FN2="+_FN[2]+" name="+c);
 System.out.print(" FN3="+_FN[3]+" name="+d);
 System.out.print(" FN4="+_FN[4]+" name="+e);
 System.out.print(" FN5="+_FN[5]+" name="+f);
 System.out.println("");
 }

 module.endSet("SET_1");
 module.endBinding();
 } catch(Exception e) {
 System.out.println(e.getMessage());
 module.sendBindErrorMessage(e.toString());
 } finally {
 close(null, stmt, rs);
 }
}

private void setDefaultStyle(HttpServletRequest request, HttpServletResponse response, String
fetchType) throws ServletException, IOException {
 System.out.println("1setDefaultStyle fetchType == "+fetchType);
 Statement stmt = null;
 ResultSet rs = null;
 DataModule module = null;

 String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName", "CarImageFile"};
 int[] _FT = {java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR, java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
java.sql.Types.VARCHAR};
 try {
 module = DataModuleFactory.getDataModule(fetchType);
 // debug
 // init setDebug 가 true
 module.setDebug(parseBoolean(request), response);
 module.init(response.getOutputStream());
 module.addSetInfo("SET_1", _FN, _FT);
 } catch(Exception ex){
 throw new ServletException(ex.getMessage());
 }

 try {
 module.startBinding();
 String query = "select * from car";

 module.startSet("SET_1");
 stmt = m_conn.createStatement();
```

```
rs = stmt.executeQuery(query);
while(rs.next()) {
 HashMap map = new HashMap();
 String a = rs.getString(_FN[0]);
 String b = rs.getString(_FN[1]);
 String c = rs.getString(_FN[2]);
 String d = rs.getString(_FN[3]);
 String e = rs.getString(_FN[4]);
 String f = rs.getString(_FN[5]);

 map.put(_FN[0], a);
 map.put(_FN[1], b);
 map.put(_FN[2], c);
 map.put(_FN[3], d);
 map.put(_FN[4], e);
 map.put(_FN[5], f);

 module.addRow("SET_1", map);

 System.out.print("select FN0="+_FN[0]+" name="+a);
 System.out.print(" FN1="+_FN[1]+" name="+b);
 System.out.print(" FN2="+_FN[2]+" name="+c);
 System.out.print(" FN3="+_FN[3]+" name="+d);
 System.out.print(" FN4="+_FN[4]+" name="+e);
 System.out.print(" FN5="+_FN[5]+" name="+f);
 System.out.println("");
}

module.endSet("SET_1");
module.endBinding();
} catch(Exception e) {
 System.out.println(e.getMessage());
 module.sendBindErrorMessage(e.toString());
} finally {
 close(null, stmt, rs);
}
}

private void setResultsetQueryStyle(HttpServletRequest request, HttpServletResponse
response, String fetchType) throws ServletException, IOException {
 System.out.println("setResultSetQueryStyle...");
 Statement stmt = null;
 ResultSet rs = null;
 DataModule module = null;

 try {
 module = DataModuleFactory.getDataModule(fetchType);
 // debug true
 // init setDebug 가
 module.setDebug(parseBoolean(request), response);
 String query = "select * from car";
 stmt = m_conn.createStatement();
 rs = stmt.executeQuery(query);

 module.makeSDM_SET("SET_1", rs, response.getOutputStream());
 } catch(Exception e) {
 System.out.println(e.getMessage());
 throw new ServletException(e.getMessage());
 } finally {
 close(null, stmt, rs);
 }
}

private String getEncode(String value) {
 try {
```

```

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

private void close(Connection con, Statement stmt, ResultSet rs) {
 try {
 if (rs != null) {
 rs.close();
 }

 if (stmt != null) {
 stmt.close();
 }

 if (con != null) {
 con.close();
 }
 } catch (SQLException se) {}
}

private boolean parseBoolean(HttpServletRequest request) {
 String tmp = (String) request.getParameter(_KEY_OZ_DEBUG_);
 if ("true".equalsIgnoreCase(tmp)) {
 return true;
 } else {
 return false;
 }
}
}

```

### Step 3

- DebugDMSampleServlet.class      WAS      .
- DebugDMSampleServlet.class      WAS      .
    - Tomcat 5.0      webapps\ROOT\WEB-INF\classes\sample
  - DebugDMSampleServlet.class      WAS      .
    - Tomcat 5.0      webapps\ROOT\WEB-INF      web.xml

```

...

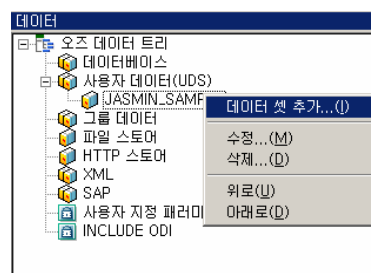
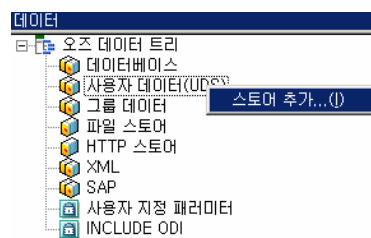
<!-- JSPC servlet mappings start -->
...
<servlet>
 <servlet-name>sample.DebugDMSampleServlet </servlet-name>
 <servlet-class>sample.DebugDMSampleServlet</servlet-class>
</servlet>
...
<servlet-mapping>

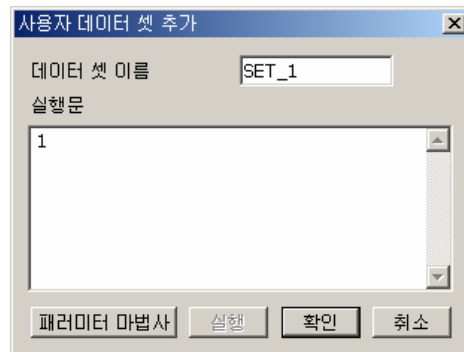
```

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

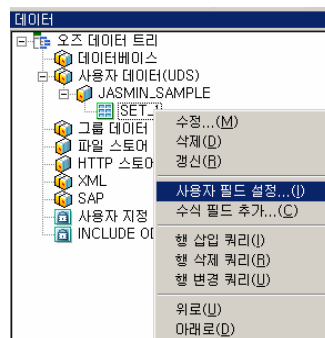
#### Step 4 ODI

Servlet API (UDS) ,  
가 .  
가 POST  
" \_OZ\_ODIITEM\_" 가 .  
가 .  
(UDS) 가

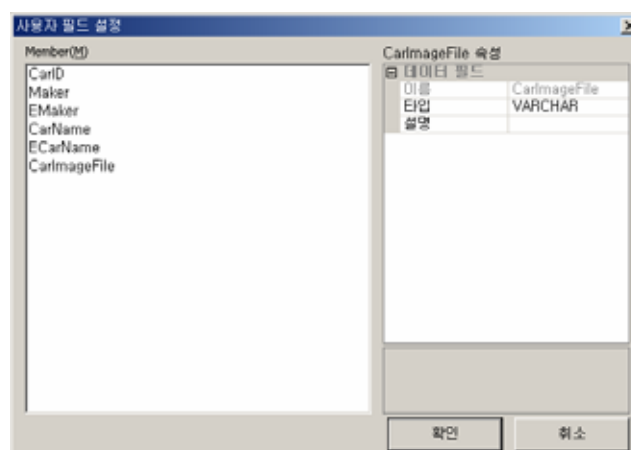




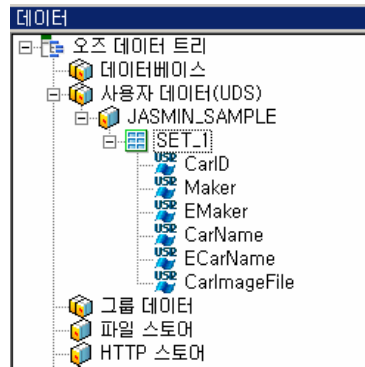
➤ 가 . 가  
[ ]



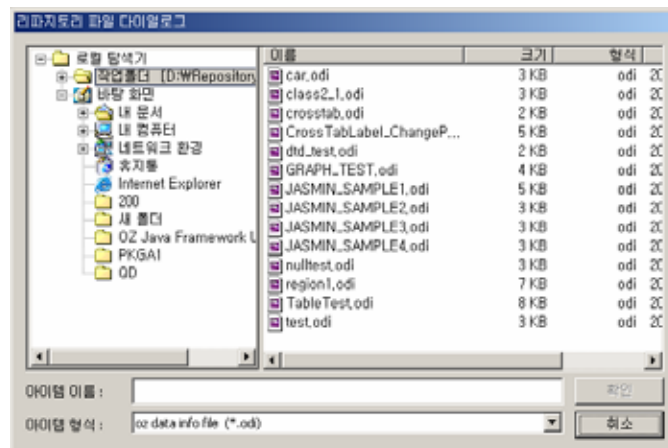
➤ 가 [ ]



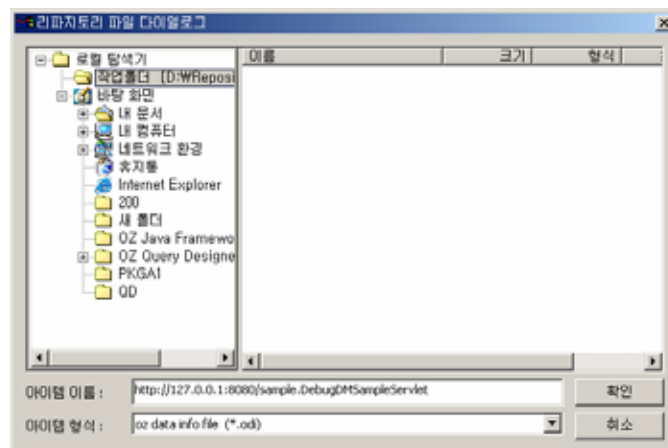
가 가 .



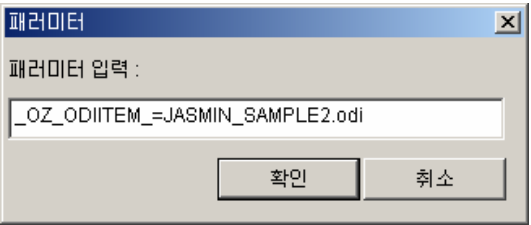
가



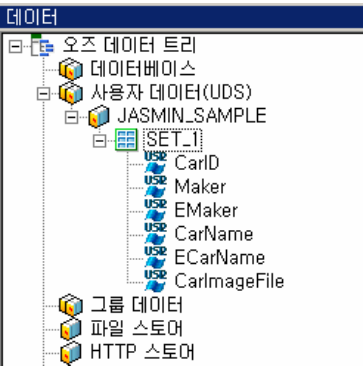
URL [ ]



➤ "\_OZ\_ODIITEM\_=ODI" [ ]



➤ 가 가



➤ ODI "JASMIN\_SAMPLE2.odi"

Step 5

XML SDM

➤



```
5 : FXDataModule
```

Servlet API                      FX Data Accesses    "\_OZData\_"

가         Table                      ,         ,

.

XML                  Servlet API

ozsdmapi.jar, crimson.jar         WAS                  .

| Step 1 | ozsdmapi.jar | crimson.jar | WAS |
|--------|--------------|-------------|-----|
|--------|--------------|-------------|-----|

|                           |                           |                          |
|---------------------------|---------------------------|--------------------------|
| Servlet API               | ozsdmapi.jar, crimson.jar | WAS                      |
| .                         | Tomcat 5.0                | webapps\ROOT\WEB-INF\lib |
| ozsdmapi.jar, crimson.jar | .                         |                          |

## Step 2 FXDataModule

```
FXDataModule . java
```

Test.class

```
package sample;

import java.io.IOException;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import oz.fxapi.core.ConvertException;
import oz.fxapi.core.DataActionParser;
import oz.fxapi.dm.FX_DataAction;
import oz.fxapi.dm.FX_DataFieldMeta;
import oz.fxapi.dm.FX_DataModule;
import oz.fxapi.dm.FX_DataSet;
import oz.fxapi.dm.FX_DataSetMeta;
import oz.fxapi.dm.FX_DataTypes;
import oz.fxapi.dm.FX_Parameter;
import oz.fxapi.dm.FX_Record;
import oz.fxapi.dm.FX_UnmatchingMetaException;
import oz.util.OZSQL;
import oz.util.OZString;

/** <p>Title: Framework test module</p>
 * <p>Description: Tests framework functionality </p>
 * <p>Copyright: Copyright (c) 2006 by FORCS All rights reserved </p>
 * <p>Company: FORCS Co.,LTD.</p>
```

```
* @author kgn
*/
public class Test extends HttpServlet
{
 private static final long serialVersionUID = 8673861145524516336L;

 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
 {
 FX_DataModule dm = new FX_DataModule();

 // 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)
 {
 dm.sendErrorMessage(response.getOutputStream(), isSDM, isCompressed, e
 .getMessage());
 OZSQL.close(con);
 OZSQL.close(con2);
 throw new ServletException(e.getMessage());
 }

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

 FX_DataSetMeta fxMeta = new FX_DataSetMeta("IDs");
 fxMeta.addDataFieldMeta(new FX_DataFieldMeta("CarID",
 FX_DataTypes.FX_DT_VARCHAR));
 dm.addDataSetMeta(fxMeta);

 fxMeta = new FX_DataSetMeta("Informations");
 fxMeta.addDataFieldMeta(new FX_DataFieldMeta("Maker",
 FX_DataTypes.FX_DT_VARCHAR));
 fxMeta.addDataFieldMeta(new FX_DataFieldMeta("CarName",
 FX_DataTypes.FX_DT_VARCHAR));
 fxMeta.addDataFieldMeta(new FX_DataFieldMeta("ECarName",
 FX_DataTypes.FX_DT_VARCHAR));
 fxMeta.setMasterSetName("IDs");
 dm.addDataSetMeta(fxMeta);

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

 ResultSet master = stmt.executeQuery("Select CarID from car");
 try
 {
 FX_DataSet masterSet = new FX_DataSet("IDs");
 while (master.next())
 {
 FX_Record record = new FX_Record();
 id = master.getString(1);
 record.addColumn("CarID", id);

 if (!OZString.isNullOrEmpty(id))
 {
 ResultSet detail = stmt2
 .executeQuery("Select Maker, CarName, ECarName from
car where CarID = "
 + id + "");
 try
 {
 FX_DataSet detailSet = new FX_DataSet("Informations");
 while (detail.next())
 {
 FX_Record detailRecord = new FX_Record();
 detailRecord
 .addColumn("Maker", detail.getString(1));
 detailRecord.addColumn("CarName", detail
 .getString(2));
 detailRecord.addColumn("ECarName", detail
 .getString(3));
 detailSet.addRecord(detailRecord);
 }
 }
 }
 }
 }
}
```

```
 }
 record.addDetailSet(detailSet);
 } finally
 {
 detail.close();
 }
}

masterSet.addRecord(record);
}

dm.addDataSet(masterSet);

} catch (Exception e)
{
 dm.sendMessage(response.getOutputStream(), isSDM, isCompressed, e
 .getMessage());
 throw e;
} finally
{
 master.close();
 stmt.close();
 stmt2.close();
 OZSQL.close(con);
 OZSQL.close(con2);
}

try
{
 String error = request.getParameter("error");
 if(!OZString.isNullOrEmpty(error))
 {
 throw new FX_UnmatchingMetaException(error);
 }
 dm.write(response.getOutputStream(), isSDM, isCompressed);
}
catch(FX_UnmatchingMetaException ue)
{
 dm.sendMessage(response.getOutputStream(), isSDM, isCompressed, ue
 .getMessage());
}
catch(ConvertException ce)
{
 dm.sendMessage(response.getOutputStream(), isSDM, isCompressed, ce
 .getMessage());
}
catch(IOException e)
{
 // ignore
}

}

private void processDAC(HttpServletRequest request,
 HttpServletResponse response) 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);
 throw new ServletException(e.getMessage());
 }

 FX_DataModule dm = new FX_DataModule();
 FX_DataAction[] dacs = dm.getDataAction(request);

 for (int i = 0; i < dacs.length; i++)
 {
 FX_DataAction 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);
 }
 }
}

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(FX_DataAction dac, Connection con) throws Exception
{
 FX_Parameter[] srcFields = dac.getSourceFields();

 if (OZString.IsNullOrEmpty(m_insert))
 {
 // prepared
 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()));
}

m_insertStmt.execute();
}

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

 if (OZString.isNullOrEmpty(m_update))
 {
 // prepared statement
 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()));
 }

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

```

 i++;
 }
 m_updateStmt.execute();
}

private void delete(FX_DataAction dac, Connection con) throws Exception
{
 FX_Parameter[] 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()));
 }

 m_deleteStmt.execute();
}

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

```

**Step 3**

- Test.class WAS .
- Test.class . Tomcat
- 5.0 webapps\ROOT\WEB-INF\classes sample
- Test.class WAS . Tomcat 5.0

webapps\ROOT\WEB-INF web.xml

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

**Step 4**

가

➤ FX Data Accesses "\_OZData\_"

[ FX DataSet]

가

[ ]

가 .

➤ 가

[

가]

가 .

➤ 가

[ Detail FX DataSet]

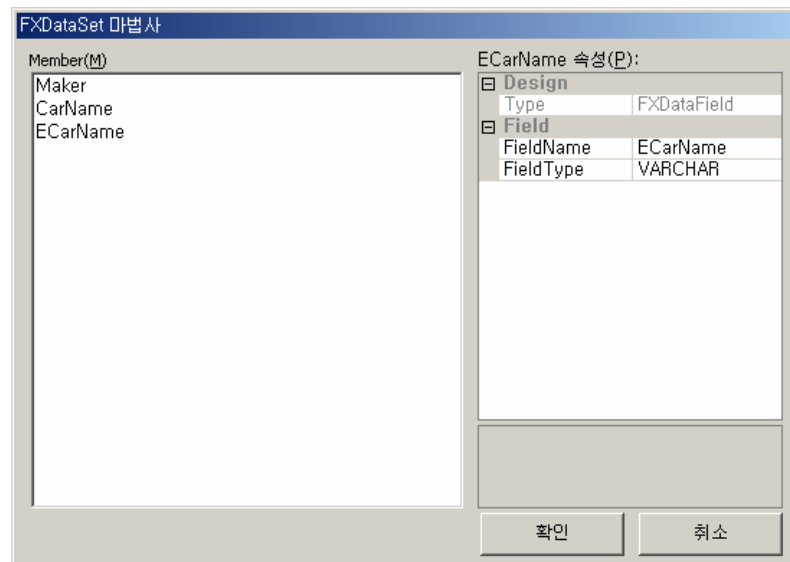
가

[ ]

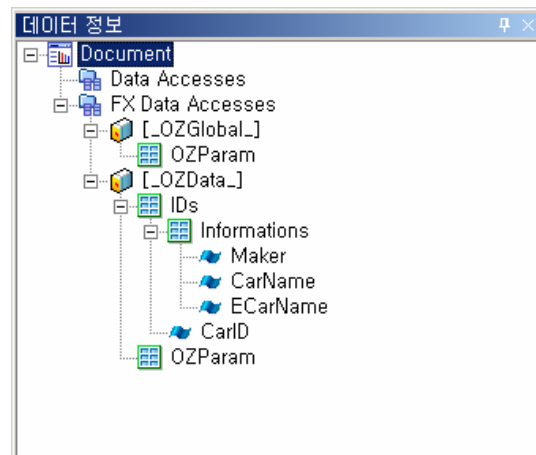
가 .

➤ 가

[ 가] 가 .



➤ - 가 .



### Step 5 OZA

Board Panel, Label, TextBox, Button, Table 가 ,

➤ Label 가 Lable "Text" "CarID", "Maker", "CarName", "ECarName" .

- Button 4 가 Button "Text" "XML", "Insert", "Update", "Delete" .
- TextBox 가 (Location), (Size) TextBox  
 TextBox1 "ODIKey" "\_OZData\_" , "DataSet" "IDs" , "Field" "CarID" "TextBox2", "TextBox3", "TextBox4" "ODIKey" "\_OZData\_" , "DataSet" "Informations" "Field" "Maker", "CarName", "ECarName" .
- Table 2 가 Table1 "ODIKey" "\_OZData\_" , "DataSet" "IDs" , "FireRowCursorChange" "True" Table2 "ODIKey" "\_OZData\_" , "DataSet" "Informations" .
- Table1 "CarID" 가 , Table2 "Maker", "CarName", "ECarName" 가 .

#### Step 5

- XML DataAction .  
 [XML ] "OnClick" 가 XML .

```

_GetFXDataModule().RemoveAllDataSet();
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

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

```

- [Insert] "OnClick" "CarID: K04" 가 .

```

_GetFXDataModule().RemoveAllDataSet();

```

```
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

// XML,
xmlhttp.Open("POST",
http://127.0.0.1:8080/Test?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=
&0.SF_2=CarName&0.SFT_2=VARCHAR&0.SV_2= &0.SF_3=ECarName&0.SFT_3
=VARCHAR&0.SV_3=SEPHIA, false);
xmlhttp.Send("");
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
 _MessageBox(dm.FXErrorMessage);
}
```

➤ [Update]                      'OnClick'                      "CarID: K04"                      가

```
_GetFXDataModule().RemoveAllDataSet();
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

// XML,
xmlhttp.Open("POST",
"http://127.0.0.1:8080/Test?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("");
var dm = _GetFXDataModule();
dm.ApplyData(xmlhttp.responseStream);
if(dm.FXErrorMessage != ""){
 _MessageBox(dm.FXErrorMessage);
}
```

➤ [Delete]                      "OnClick"                      "CarID: K04"                      가

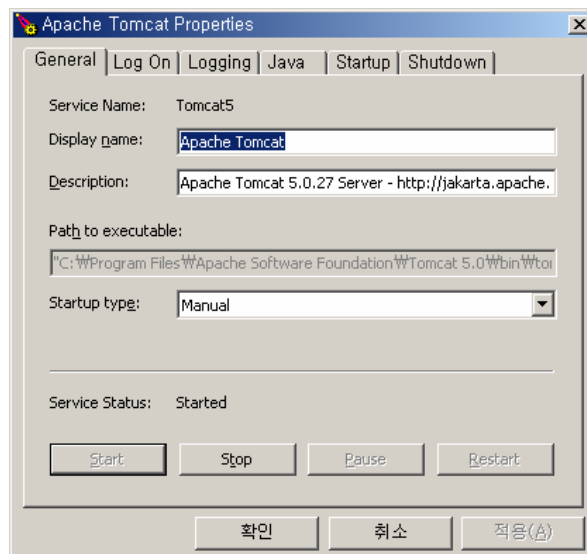
```
_GetFXDataModule().RemoveAllDataSet();
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

// XML,
xmlhttp.Open("POST",
http://127.0.0.1:8080/Test?type=dac&_OZ_DAC_CNT_=1&0.TYPE=delete&0.TRG_CNT=1&0.D
F_0=CarID&0.DFT_0=VARCHAR&0.DV_0=K04, false);
xmlhttp.Send("");
var dm = _GetFXDataModule();
```

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

**Step 6**

➤ WAS



➤ [File] [Preview] (  )

