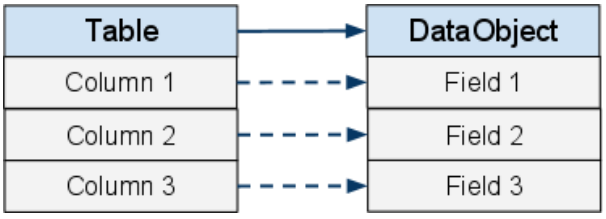
OpenbizFrameworkDataObject
describe Openbiz Data Object
Phase-Implementation

Updated Dec 8, 2012 by agus.suhartono

Openbiz Data Object

Openbiz maps database tables to data object (DO) with BizDataObj class. Each DO declares a table name and mapping between table columns to DO fields. The following chart shows the simplest DO which maps to one table.



DO Metadata

Openbiz DO has a core class “BizDataObj”. This class metadata is described below.

```
<BizDataObj ...>
    <BizFieldList>
        <BizField ...> *
    <TableJoins>
        <Join ...> *
    <ObjReferences>
        <Object ...> *
```

BizDataObj Element

BizDataObj element is the root element of a DO metadata. It has 3 children elements - BizFieldList, TableJoins and ObjReferences. BizDataObj element has the following attributes:

Name

Name is the identifier of a DO object. Name should be same as the DO file name.

Description

Description describes the functions and behavior of the DO.

Class

Class specifies the name of its implementation class.

DBName

DBName is the Database Name defined in application Config.xml

Table

Table is a database table name which maps to the DO. We also call the “Table” in the BizDataObj element as “base table” of the DO.

IDGeneration

IDGeneration specifies generation algorithm of the primary key of the base table. The value can be "Openbiz”, “Identity”, “Sequence:seqname”, “GUID”, “UUID”, and other.

SearchRule

SearchRule applies search criteria to fetch the data from the data source (database tables). Example:

```
SearchRule="[start_time]>'1999-10-20'"
```

SortRule

SortRule applies sort criteria to fetch the data from the data source (database tables). Example:

```
SortRule="[start_time] ASC"
```

!OtherSQLRule

!OtherSQLRule applies user specified SQL expression to fetch the data from the data source (database tables). Example:

```
OtherSQLRule = "GROUP BY [position] HAVING [age]>30"
```

AccessRule

AccessRule gives additional search criteria to limit the data according to the access control. Example:

```
AccessRule="[owner_id]='{@profile:Id}'"
```

CreateCondition

CreateCondition sets the access control (ACL) for creating record. Example:

```
CreateCondition=“project.create”
```

Here “project” is the resource name and “create” is the action name. They are defined in the ACL section of the Cubi module xml file.

UpdateCondition

UpdateCondition sets the access control (ACL) for updating record. Example:

```
UpdateCondition=“project.update”
```

DeleteCondition

DeleteCondition sets the access control (ACL) for deleting record. Example:

```
DeleteCondition=“project.delete”
```

Uniqueness

Uniqueness sets the validation condition that checks if values of certain fields are unique. Syntax:

```
Uniquess = "fld1,fld2;fld3,fld4;...."
```

Openbiz will check the uniqueness of (fld1,fld2) and (fld3,fld4). Exception will be thrown if check fails.

MessageFile

MessageFile specifies the file path that provides message strings for the form. Message strings are used in the class source code for displaying messages like error, alert and log message.

CacheLifetime

CacheLifetime sets the DO cache life time in seconds. If CacheLifetime is larger than 0, DO query results are cached for the length given by the value. By default the cache is turned off.

BizField Element

BizField element describes the Field of a DO. A Field usually maps to a table column. It can also be a SQL expression or calculation value. BizField element has the following attributes:

Name

Name is the identifier of a BizField.

Join

Join is the name defined in the

```
<join>
```

element. If “Join” is given in BizField element, the “Column” attribute points to the column from the joined table.

Column

Column is the column of the base table by default. If “Join” is given in BizField element, the “Column” attribute points to the column from the joined table

SQLExpr

SQLExpr maps a field to the SQL functions provided by the database engine. If SQLExpr is given, make sure that the Column attribute is empty. Example:

```
<BizField Name="FullName" Column="" SQLExpr="CONCAT([FirstName],' ',[LastName])"...>
```

Type

Type sets the data type of the field. DO types supported by Openbiz include Text, Number, Date, Datetime, Currency, Phone, Blob.

Format

Format specifies the display format of given field type. Different type can have different format. We will cover more details in the “Data type and format” section.

Value

Value sets the calculated value of the given field. Value supports Simple Expression. Example:

```
<BizField Name="Precentage" Column="" Value="[corrects]/[total]"...>
```

Where corrects and total are 2 fields defined in the same DO.

DefaultValue

DefaultValue defines a default value of a field when a new record is created.

Validator

Validator defines a validation rule that is used in validating the field value on record insert or update. Example:

```
<BizField Name="Fee" Column="fee" Type="Currency" Format="Currency" Validator="{[Fee]>=15}" >
```

ValueOnCreate

ValueOnCreate sets the value of a field on record creation. Example:

```
<BizField Name="changetime"
          Column="changetime"
          Type="Datetime"
          ValueOnCreate="{date('Y-m-d H:i:s')}"
          ValueOnUpdate="{date('Y-m-d H:i:s')}"/>
```

ValueOnUpdate

ValueOnUpdate sets the value of a field on record update.

Join Element

Join element describe the a table join to the base table of the current DO. Through Join, a DO can have Fields mapping to columns of more than one tables. Join element has the following attributes:

Name

Name is the identifier of a Join element.

Table

Table is the name of a database table that joins to the DO’s base table.

Column

Column defines a column of the joined table. This column is usually the column with foreign key pointing to the primary key of the base table.

JoinRef

JoinRef defines a reference of a table join. Use JoinRef, we can define a join of another table join. It is also called second join.

ColumnRef

ColumnRef refers to the column of the base table (or the JoinRef table). ColumnRef usually is column with primary key of the base table.

JoinType

JoinType defines the type of table join. The value can be INNER JOIN, LEFT JOIN, RIGHT JOIN, or FULL OUTER JOIN.

Object Element

Object element describe the another DO that holds certain relationship to the current DO. The relationship can be Many-to-One, One-to-Many,

One-to-One and Many-to-Many. Object element has the following attributes:

Name

Name is the identifier of a data object reference

Relationship

Relationship defines the relationship between this DO and the referred DO. The value can be M-1, 1-M, M-M and 1-1.

Table

Table defines the table of the referred DO

Column

Column points to the column of the referred DO's table. This column contains the foreign key to the base table

OnDelete

OnDelete tells what action to take before deleting a record. The options are:

- Cascade. Whenever rows in the master (referenced) table are deleted, the respective rows of the child (referencing) table with a matching foreign key column will get deleted as well. This is called a cascade delete. Same as CascadeDelete = Y
- Restrict. A value cannot be deleted when a row exists in a foreign key table that references the value in the referenced table.
- SetNull. The foreign key values in the referencing row are set to NULL when the referenced row is deleted

OnUpdate

OnUpdate tells what action to take before updating a record. The options are:

- Cascade. Whenever rows in the master (referenced) table are modified, the respective rows of the child (referencing) table with a matching foreign key column will get updated with same value as well.
- Restrict. A value cannot be updated when a row exists in a foreign key table that references the value in the referenced table.
- SetNull. The foreign key values in the referencing row are set to NULL when the referenced row is updated

XTable

XTable defines the intersection table name if the relationship of this DO and the reference DO is many to many.

XColumn1

XColumn1 is the intersection table column that has the foreign key of the base table

XColumn2

XColumn2 is the the intersection table column that has the foreign key of the referred table

XDataObj

XDataObj is a DO whose base table is the intersection table that is XTable. When user associates one record from referred DO to the base DO, a new record is created from the XDataObj and added in the intersection table.

Data Object API and Samples

Openbiz DO provides intuitive high level API for CURD operations. In order to assist developers to avoid hand coding SQL statements, Openbiz suggests using its Query Language.

Query Language

DO supports simple query language at the object level. The basic syntax is

```
"[FieldName] opr 'Value' AND/OR [fieldName] opr 'Value'..."
```

Here "opr" is a SQL operator. At run-time, openbiz converts FieldName to column name and generate SQL statement. Openbiz also puts table relationship in generated query statements.

The Query Language is not only used in the API, but also widely used in DO's metadata, such as SearchRule, Sort Rule, OtherSQLRule …

Data Query

Direct Query

```
// fetch all record with firstname starting with Mike
$dataSet = $do->directFetch("[FirstName] LIKE 'Mike%'");
// fetch first 10 records with firstname starting with Mike
$dataSet = $do->directFetch("[FirstName] LIKE 'Mike%'", 10);
// fetch 20th-30th records with firstname starting with Mike
$dataSet = $do->directFetch("[FirstName] LIKE 'Mike%'", 10, 20);
```

Query by Id

```
// fetch one record given Id as primary key
$dataRecord = $do->fetchById("12");
```

Query One Record

```
// fetch one record given search rule
$dataRecord = $do->fetchOne("[FirstName] LIKE 'Mike%'");
```

Prepare and Query

```
// Fetch a data set according the query rules set before
$do->resetRules();
$do->setSearchRule($search_rule1);
$do->setSearchRule($search_rule2);
$do->setSortRule($sort_rule);
$do->setOtherRule($groupby);
$total = $do->count();
$do->setLimit($count, $offset=0);
$dataSet = $do->fetch();
```

Query by Zend DB Statement

```
// Do the search query and return  PDOStatement
$do->resetRules();
$do->setSearchRule($search_rule1);
$do->setSearchRule($search_rule2);
$do->setSortRule($sort_rule);
$do->setOtherRule($groupby);
$total = $do->count();
$do->setLimit($count, $offset=0);
$dbStmt = $do->find();
$do->getDBConnection()->setFetchMode(PDO::FETCH_ASSOC);
while ($record = $dbStmt->fetch())
{
  print_r($record);
}
```

Query by Zend DB instance

```
$db = $do->getDbConnection();
// Start a transaction explicitly.
$db->beginTransaction();
try {
   // Attempt to execute one or more queries:
   $db->query(...);
   $db->query(...);
   // If all succeed, commit the transaction
   $db->commit();
} catch (Exception $e) {
   // rollback in case of exception
   $db->rollBack();
   echo $e->getMessage();
}
```

Insert, Update and Delete Record

Even though BizDataObj has public methods for insert, update and delete, it is highly recommended to use DataRecord for such tasks.

Insert a record

```
// Insert a record
$dataRec = new DataRecord(null, $dataobj);
$dataRec->first_name = 'Steve';
$dataRec->last_name = 'Jobs';
$dataRec->save( );
/* Note: the following code works too. */
$dataRec['first_name'] = 'Steve';
$dataRec['last_name'] = 'Jobs';
$dataRec->save( );
```

Update a record

```
// Update a record. $recordArray is the to-be-updated record
$dataRec = new DataRecord($recordArray, $dataobj);
$dataRec->first_name = 'Steve';
$dataRec->last_name = 'Jobs';
$dataRec->save( );
/* Note: the following code works too. */
$dataRec['first_name'] = 'Steve';
$dataRec['last_name'] = 'Jobs';
$dataRec->save( );
```

Delete a record

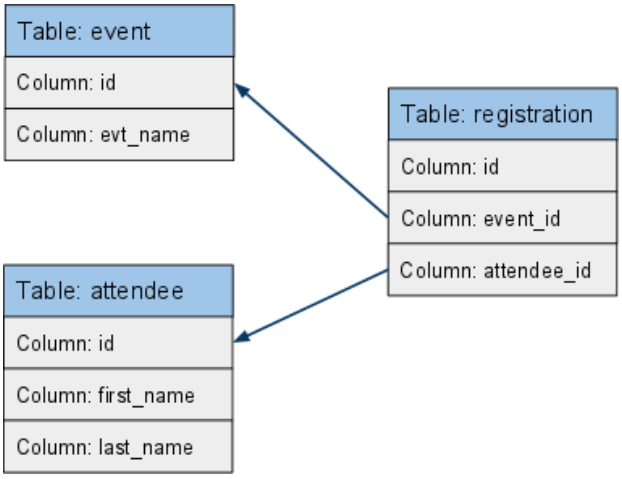
```
// Delete a record. $recordArray is the to-be-deleted record
$dataRec = new DataRecord($recordArray, $dataobj);
$dataRec->delete( );
```

Object Relational Mapping (ORM)

Openbiz supports Object Relational Mapping (ORM) in Data Object metadata. In the DO metadata

- Join element is to join multiple tables data into a single DO
- Object element is to link other DOs to the main DO.

A simple sample database schema is used in the following chapters to help understanding how Openbiz ORM works.



attendee:registration = 1-M (one to many), registration:attendee = M-1 (many to one) events:registration = 1-M (one to many), registration:events = M-1 (many to one) attendee:events = M-M (many to many)

Map a table to a DO

To map a table to a DO, you only need to configure the BizFieldList part in the metadata. For event DO, the metadata is like.

```
<BizDataObj Name="EventDO" Table="event" ...>
 <BizFieldList>
   <BizField Name="Id" Column="id"/>
   <BizField Name="name" Column="name"/>
 </BizFieldList>
</BizDataObj>
```

Map multiple tables to a DO

You can map multiple tables to a DO through Join. Joined table is referred by a foreign key column in the base table of the DO, that says the base table has a foreign key points to joined table's column. If Join is not given, the column is from the main table of DO. For registration DO, the metadata is like

```
<BizDataObj Name="RegistrationDO" Table="registration" ...>
 <BizFieldList>
   <BizField Name="Id" Column="id"/>
   <BizField Name="event_id" Column="event_id"/>
   <BizField Name="attendee_id" Column="attendee_id"/>
   <BizField Name="attd_first_name" Join="attendee" Column="first_name"/>
   <BizField Name="attd_last_name" Join="attendee" Column="last_name"/>
 </BizFieldList>
 <TableJoins>
   <Join Name="attendee" Table="attendee" Column="id" ColumnRef="attendee_id" JoinType="INNER JOIN"/>
 </TableJoins>
</BizDataObj>
```

You can see from the above metadata, with the definition of Join, registration DO contains the data from attendee table.

The query SQL of a DO with join is like

```
SELECT ...
FROM BaseTable
INNER JOIN JoinedTable ON BaseTable.ColumnRef=JoinedTable.JoinedColumn ...
```

In the above example, the SQL of query registration DO is

```
SELECT T0.id, T0.event_id, T0.attendee_id, T1.first_name, T1.last_name
FROM registration as T0
INNER JOIN attendee as T1 ON T0.attendee_id = T1.id
```

Use JoinRef for cascade join

JoinRef is to support cascade-join (or second join). For example, in a query, table A joins B and table B joins C. Table A and C is joined through table B. User can use the following part to link C and A together. Assume the base table of a DO is table A, the following xml shows how to use JoinRef to join table C columns to this DO.

```
<TableJoins>
 <Join Name="join_b" Table="B" Column="B_PK" ColumnRef="A_FK_B" JoinType="LEFT JOIN"/>
 <Join Name="join_c" Table="C" JoinRef="join_b" Column="C_PK" ColumnRef="B_FK_C" JoinType="LEFT JOIN"/>
</TableJoins>
```

Many to one relationship between DOs

Many to one relationship between two tables means that table 1 has a column that contains the foreign key pointing to a key/unique column in table 2. To map such relationship between two DOs, Object in ObjectReference section is used. With the sample schema, the registration DO metadata is like.

```
<BizDataObj Name="RegistrationDO" Class="BizDataObj" Table="registration" ...>
 <BizFieldList>
   <BizField Name="Id" Column="id"/>
   <BizField Name="event_id" Column="event_id"/>
   <BizField Name="attendee_id" Column="attendee_id"/>
   <BizField Name="attd_first_name" Join="attendee" Column="first_name"/>
   <BizField Name="attd_last_name" Join="attendee" Column="last_name"/>
 </BizFieldList>
 <TableJoins>
   <Join Name="attendee" Table="attendee" Column="id" ColumnRef="attendee_id" JoinType="INNER JOIN"/>
 </TableJoins>
 <ObjReferences>
   <Object Name="EventDO" Relationship="M-1" Table="event" Column="id" FieldRef="event_id"/>
 </ObjReferences>
</BizDataObj>
```

One to many relationship between DOs

One to many relationship between two tables means table 2 has a column that contains the foreign key pointing to a key/unique column in table 1. To map such relationship between two DOs, we can use an Object in ObjectReference section.

```
<BizDataObj Name="EventDO" Class="BizDataObj" Table="event" ...>
 <BizFieldList>
   <BizField Name="Id" Column="id"/>
   <BizField Name="name" Column="name"/>
 </BizFieldList>
 <ObjReferences>
   <Object Name="RegistrationDO" Relationship="1-M" Table="registration" Column="event_id" FieldRef="Id"/>
 </ObjReferences>
</BizDataObj>
```

Many to many relationship between DOs

Many to many relationship between two tables means an intersection table (also called cross reference table, or correlation table) contains the foreign key columns pointing to a key/unique column in table 1 and table 2. A reference object is used to map such relationship between two DOs.

```
<BizDataObj Name="EventDO" Table="event" ...>
 <BizFieldList>
   <BizField Name="Id" Column="id"/>
   <BizField Name="Name" Column="name"/>
 </BizFieldList>
 <ObjReferences>
   <Object Name="BOAttendee"
           Description=""
           Relationship="M-M"
           Table="attendee"
           Column="SYSID"
           FieldRef="Id"
           XDataObj="BORegist"
           XColumn1="EVENT_ID"
           XColumn2="ATTENDEE_ID"/>
 </ObjReferences>
</BizDataObj>
```

Use Object Reference

The following line is to get the instance of the reference DO from the base DO.

```
$refObject = $do->getRefObject($objectName);
```

Query on Reference DO

The reference DO instance keeps the data association to the base DO. For example,

```
$registrationDO->setActiveRecordId(101);
$refEventDO = $registrationDO->getRefObject('EventDO');
$dataSet = $refEventDO->fetch();
```

Assume registration DO current record (id=101) has event_id as 2110, the reference event DO will have id as 2110 in its DO search rule.

Add Record at Reference DO

Adding a record at Reference DO means associating a ref DO record to the base DO. For example, if we want to associate an attendee to an event (here AttendeeDO is a ref DO of EventDO with M-M relationship), we can do

```
$attendeeRefDO = $eventDO->getRefObject('AttendeeDO');
$attendeeRefDO->addRecord($attendeeRecord);
```

In case of M-M relationship, a new record is added in the intersection table on addRecord.

Remove Record at Reference DO

Removing a record at Reference DO means un-associating a ref DO record to the base DO. For example, if we want to un-associate an attendee to an event (here AttendeeDO is a ref DO of EventDO with M-M relationship), we can do

```
$attendeeRefDO = $eventDO->getRefObject('AttendeeDO');
$attendeeRefDO->removeRecord($attendeeRecord);
```

In case of M-M relationship, a record is removed from the intersection table on removeRecord.

Data types and formats

Openbiz DO Field supports following data types and formats in its metadata.

Text

Format: none. Text is the default Field Type.

Number

- Format: %... same syntax supported in sprintf function. Example:

  ```
  <BizField ... Type="Number" Format="%5.2f" ...>
  ```

- Format: Int. Integer format according to system locale. Example:

  ```
  <BizField ... Type="Number" Format="Int" ...>
  ```

If locale=enu, 12345.678 is displayed as 12,346.

- Format: Float. Float format according to system locale. Example:

  ```
  <BizField ... Type="Number" Format="Float" ...>
  ```

If locale=enu, 12345.678 is displayed as 12,345.68

Date

Date format is same as http://www.php.net/manual/en/function.strftime.php. Example:

```
<BizField ... Type="Date" Format="%A, %b %d %Y " ...>
```

6/21/2003 can be formatted as Saturday, Jun 21 2003 if system locale is enu

Datetime

Date format is same as http://www.php.net/manual/en/function.strftime.php. Example:

```
<BizField ... Type="Date" Format="%m/%d/%Y %H:%M:%S" ...>
```

6/22/2003 9:30am can be formatted as 06/22/2003 09:30:00 if system locale is enu

Currency

Currency type data is formatted according to locale setting. For example: 1456.89 is formatted as $1,456.89 (enu) 1456.89 is formatted as F1 456,89 (fra)

Phone

Phone type data is formatted according to given mask # Example:

```
<BizField ... Type="Phone" Format="mask string" ...>
```

1234567890 is formatted as mask=(###) ###-####, phone=(123) 456-7890 mask=### ### ####, phone=123-456-7890

If a phone number starting with "**", it represents international phone number, it won't be formatted.** 123 4567890 is treated as international number

Blob

Field with type “Blob” usually means a block of text or binary data. No format is for Blob type field.

Data validation

Data validation can be set in DO metadata. This is mainly to check the data input in record insert and update. Data validation in DO includes uniqueness check, required field check and apply custom validation rule to fields. If validation fails, DO throws a validation exception to the caller.

Uniqueness check

Openbiz can check the uniqueness of the given DO fields.

Syntax:

```
<BizDataObj Name=”...” Uniquess = "fld1,fld2;fld3,fld4;...." >
```

Openbiz will check the uniqueness of (fld1,fld2) and (fld3,fld4)

Example:

```
<BizDataObj Name="User" ...  Uniqueness="LastName,FirstName"/>
```

Required Field

Openbiz checks if a field can accept empty input value if “Required” attribute is set in Field metadata.

Syntax:

```
<BizField Name="..." Required="Y|N" .../>
```

Example:

```
<BizField Name="name" Required="Y" Column="name"/>
```

Validator

Field can use “Validator” to set flexible validation rules for this field. Validate field with simple expression Simple Expression can be used in validator. Example:

```
<BizField Name="Fee" Type="Currency" Format="Currency" Validator="{[Fee]>=15}" Column="FEE">
```

Validate field with validation service Validation service (Openbiz uses Zend Validation) can be use to validate the input data in Validator attribute. Examples:

```
<BizField Name="Email" Column="EMAIL" Validator="{@validate:email('[Email]')}"/>
<BizField Name="Phone" Column="PHONE" Validator="{@validate:phone('[Phone]')}"/>
```

Handle Primary Keys

Openbiz DO has build field called "Id". This field is the identifier of each record. Openbiz suggests a table should have a single primary key column that maps to "Id" field, while composite primary keys are supported as well.

Primary key (Id) generation

Openbiz DO supports multiple id generation algorithms. “Id” is a required Field of every DO. It usually maps to the primary key (PK) column of a database table. Different application might use different algorithm to create primary key. DO’s attribute IdGeneration support the following Id generation algorithms.

IdGeneration	**how id (PK) is generated**	**Supported database types**
Identity	Primary key column is generated by database engine. Id is called "Identity" column in SQL Server, also called "auto-increment" column in MySQL.	MySQL, SQL Server, Sybase, DB2
Sequence:seqname	Primary key column is generated by database sequence. "seqname" is the sequence name used to generate PK	Oracle, PostgreSQL, DB2
GUID	Primary key column is database-generated GUID	MySQL, SQL Server, Oracle
UUID	Primary key column is generated with php uuid function	Apply to all database types
Other	Developers can write customer specific Id generation algorithm by modifying **function GetNewID(...)** in genIdService class under openbiz/bin/service/genIdService.php	decided by customer

Composite primary keys

In order support composite primary key which consists of 2 or more attributes that uniquely identify an entity occurrence, the "Id" field can be written as

```
<BizField Name="Id" Column="key_column_1,key_column_2,key_column_3"  SQLExpr="1" Type="Text"/>
```

Thus DataObject can understand the composite key and make appropriate changes in SQL statements.

Value Set on Record Create/Update

Sometimes people want to set value to certain fields on record creation and update. For example, when a record is create, we want to set the current time in "create_time" column. And when this record is updated, we want to set the current time in "update_time" column. Openbiz introduces attributes in DO Field metadata to take are of such work.

- ValueOnCreate can set the value on record creation
- ValueOnUpdate can set the value on record update.

For example:

```
<BizField Column="changetime"
          Name="changetime"
          Type="Datetime"
          ValueOnCreate="{date('Y-m-d H:i:s')}"
          ValueOnUpdate="{date('Y-m-d H:i:s')}"/>
```