DataMgr 2.2 Compatibility for Custom Relations

Posted At : December 5, 2008 7:00 AM | Posted By : Steve
Related Categories: DataMgr

(**What is DataMgr?**)

I have tried to ensure that each version of DataMgr is completely backward compatible with every previous version of DataMgr. With version 2.2, however, I did introduce a small compatibility issue that may require a change in code.

The 2.2 code will work on previous versions, but code written for previous versions may need a small adjustment for new versions.

The reason for this is that while relation fields can call other relation fields, sometimes the combination doesn't make sense. For example, it does make sense to have a "label" relation that gets the value of a "sum" relation for another table. It doesn't make sense, however, to do a sum on a "concat" field (or any normal field that isn't numeric). It also isn't possible for any relation field to get the value from a "list" relation.

Now DataMgr checks for these problems.

If you did have a relation field using a list relation before 2.2 then you would have gotten an invalid/meaningless result. As of 2.2, you would get an error.

Custom relation fields, however, add a small wrinkle. Because DataMgr is just executing SQL that it is given, it has now way to know what type of data that field retrieves. Consequently, the relation element now has a CF_Datatype attribute (the same name as the attribute in the field element) to indicate the data type.

So, this code would work prior to 2.1:

```
<tables>
    <table name="tblusers">
        <field ColumnName="iduser" CF_Datatype="CF_SQL_INTEGER" PrimaryKey="true" Increment="true" />
        <field ColumnName="idusercreate" CF_Datatype="CF_SQL_INTEGER" />
        <field ColumnName="name" CF_Datatype="CF_SQL_VARCHAR" Length="50" />
        <field ColumnName="surname" CF_Datatype="CF_SQL_VARCHAR" Length="50" />
        <field ColumnName="usercreatename">
            <relation
                type="custom"
                sql="select name from tblusers u1 where u1.iduser = tblusers.idusercreate"
            />
        </field>
        <field ColumnName="usercreatesurname">
            <relation
                type="custom"
                sql="select surname from tblusers u1 where u1.iduser = tblusers.idusercreate"
            />
        </field>
        <field ColumnName="usercreatefullname">
            <relation
                type="concat"
                fields="usercreatename,usercreatesurname"
                delimiter=" "
            />
        </field>
    </table>
</tables>
```

But, would need to be written as this as of 2.2 (note the addition of the CF_Datatype attributes):

```
<tables>
    <table name="tblusers">
        <field ColumnName="iduser" CF_Datatype="CF_SQL_INTEGER" PrimaryKey="true" Increment="true" />
```

```
        <field ColumnName= "idusercreate"  CF_Datatype= "CF_SQL_INTEGER"  />
        <field ColumnName= "name"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field ColumnName= "surname"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field ColumnName= "usercreatename" >
            <relation
                type= "custom"
                CF_Datatype=  "CF_SQL_VARCHAR"
                sql= "select name from tblusers u1 where u1.iduser = tblusers.idusercreate"
            />
        </field>
        <field ColumnName= "usercreatesurname" >
            <relation
                type= "custom"
                CF_Datatype=  "CF_SQL_VARCHAR"
                sql= "select surname from tblusers u1 where u1.iduser = tblusers.idusercreate"
            />
        </field>
        <field ColumnName= "usercreatefullname" >
            <relation
                type= "concat"
                fields= "usercreatename,usercreatesurname"
                delimiter= " "
            />
        </field>
    </table>
</tables>
```

As an aside, DataMgr actually does handle self-referential relationships without any trouble, so this example could have been written without custom relation fields:

```
<tables>
    <table name= "tblusers" >
        <field ColumnName= "iduser"  CF_Datatype= "CF_SQL_INTEGER"  PrimaryKey= "true"  Increment= "true"  />
        <field ColumnName= "idusercreate"  CF_Datatype= "CF_SQL_INTEGER"  />
        <field ColumnName= "name"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field ColumnName= "surname"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field ColumnName= "usercreatename" >
            <relation
                type= "label"
                table= "tblusers"
                field= "name"
                join-field-local=  "idusercreate"
                join-field-remote=  "iduser"
            />
        </field>
        <field ColumnName= "usercreatesurname" >
            <relation
```

```
                type= "label"
                table= "tblusers"
                field= "surname"
                join-field-local=  "idusercreate"
                join-field-remote=  "iduser"
            />
        </field>
        <field ColumnName= "usercreatefullname" >
            <relation
                type= "concat"
                fields= "usercreatename,usercreatesurname"
                delimiter= " "
            />
        </field>
    </table>
</tables>
```

Of course, if the "usercreatename" and "usercreatesurname" fields only existed for the purpose of creating a "usercreatefullname" field, then you could just use a "fullname" relation and reference that:

```
<tables>
    <table name= "tblusers" >
        <field ColumnName= "iduser"  CF_Datatype= "CF_SQL_INTEGER"  PrimaryKey= "true"  Increment= "true"  />
        <field ColumnName= "idusercreate"  CF_Datatype= "CF_SQL_INTEGER"  />
        <field ColumnName= "name"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field ColumnName= "surname"  CF_Datatype= "CF_SQL_VARCHAR"  Length= "50"  />
        <field columnname= "fullname" >
            <relation
                type= "concat"
                fields= "name,surname"
                delimiter= " "
            />
        </field>
        <field columnname= "usercreatefullname" >
            <relation
                type= "label"
                table= "tblusers"
                field= "fullname"
                join-field-local=  "idusercreate"
                join-field-remote=  "iduser"
            />
        </field>
    </table>
</tables>
```

In the meantime, I will try to keep compatibility issues to a minimum. Hopefully this won't cause any one too much trouble.

DataMgr is open source and free for any use. Download the 2.2 Beta from the **DataMgr page** on my site.