DataMgr is Better than ColdFusion ORM

Posted At : September 22, 2009 9:00 AM | Posted By : Steve
Related Categories: ColdFusion, DataMgr

I have been thinking this for some time, but it seemed like hubris (and flame bait) to say it. So be it - it needs to be said. I worry that ColdFusion ORM is being / will be marketed where it isn't appropriate.

(**What is DataMgr?**)

What is Better?

It can't be helped, "better" is a word that only makes sense in the context of goals to be met. After all, it is impossible to tell if a sedan is better than a truck unless you know your goals. Do you need to get to work on low gas mileage or haul cargo? I will use the criteria that Adobe ColdFusion engineer, **Rupesh Kumar**, has laid out in "**ColdFusion ORM - An Evolution in building datacentric application**":

- [Dependency] Database vendor dependency - The queries that are written are DB specific. If you ever need to switch the database or if you need to make the application database agnostic, it becomes too much of pain. You will end up changing the entire application code or lot of conditional code to handle queries for different databases.
- [Coupling] Very tight coupling with the database - The queries have very tight coupling with the database. This means that if there is any change in the table name or column name, you will end up with changes all over your application code.
- [Repetition] Repititive and hence low on productivity - For any CRUD (Create, read, update and delete) operation, you need to write queries which are more or less same for all the tables. You must have felt that you are repeating the same code everywhere in the application. There is too much of repititiveness which reduces productivity.
- [Errors] Error prone - How many times did you write insert/update queries and then you had to struggle to debug because the order of column and its values were different? And how many times did you get frustrated because there was a datatype mismatch between the value you had specified and the one expected? How many times did you write a query that was prone to sql injection and you were reminded to use query param? I am sure all of this has happened to all of us and its completely natural given the way queries are written. They are error prone.

So, do how do ColdFusion ORM and DataMgr stack up on these goals?

- Dependency: Both allow you to write database neutral code.
- Coupling: Both help with this, but each will result in code that references database table and column names.
- Repetition: Both greatly reduce repetition.
- Errors: Both solve this issue in much the same way.

ORM or DAL?

Note that what is not on that list is "Map Objects to a Relational Database". Where that is a criterion, an ORM is the solution (after all it is Object Relation Mapping). DataMgr isn't an ORM, so ColdFusion ORM is a better solution where that is a problem/requirement. Where it isn't, however, DataMgr is the better fit.

DataMgr is better (as a **Data Access Layer**) because it *isn't* an ORM. (see also **Ike Dealey**'s blog entry on **Pros and Cons of an ORM**).

If you are doing pure OO development, an ORM is great (but the reality is that many of us are not). If you aren't, it presents some challenges:

What's Wrong with ORM?

Availability

DataMgr is available now and will run on versions of ColdFusion stretching back to ColdFusion MX 6.1 as well as Railo and OpenBD. ColdFusion ORM is part of ColdFusion 9, so you will have to wait until then. It may be included in Railo or OpenBD later as well, but in any event, it isn't available in full release yet.

Existing Table Structure

Hibernate (in particular) has opinions about table structure. If your table structure doesn't match how it things tables should be arranged, you have extra work to do.

- **Learning ColdFusion 9: From SQL To ORM - A Conceptual Shift In Relationships**

"Stop Thinking About the Database"

Development using an ORM is supposed to make you "stop thinking about the database". A fine thing except that most ColdFusion applications are all about the database. Often any problems will be met with that perspective (this isn't a criticism of those who give that advice, but rather a reality of the tool itself).

- **Learning ColdFusion 9: From SQL To ORM - A Conceptual Shift In Relationships**

SQL Execution Timing

The SQL doesn't execute immediately in Hibernate, instead executing at the end of a Hibernate "session" which almost matches a ColdFusion request. This means you have to start thinking about Hibernate sessions.

- **ColdFusion 9 ORM - Explaining Hibernate Object State**
- **Learning ColdFusion 9: When Does An ORM-Enabled Object Get Persisted**
- **Some thoughts on ColdFusion 9 ORM and Persistent CFCs**

Extra Code

You have to write extra code in ColdFusion ORM (compared to DataMgr):

As an example, consider a Contacts table with the following fields: "ContactID, FirstName, LastName, Email, Phone" being updated with the following structure {ContactID=1, FirstName="Steve", LastName="Bryant", Email="steve@example.com", Phone="918-555-1212"} (which I'll call "sData", but could come from a Form post or Arguments scope, for example.

(I'll skip the Application.cfc settings and the single line to **instantiate DataMgr** as those things are done just once per application.)

To save a record with ColdFusion ORM, first need a CFC:

```
<cfcomponent persistent="true" table="Contacts"></cfcomponent>
```

In fairness, I don't call this a drawback of ColdFusion 9 ORM because I would use a CFC with DataMgr myself and the code is minute here (assuming you are willing to let ColdFusion 9 introspect the table).

Then the code would be:

```
<cfset oContact = EntityLoad("Contact",sData.ContactID, True)>
<cfset oContact.setFirstName(sData.FirstName)>
<cfset oContact.setLastName(sData.LastName)>
<cfset oContact.setEmail(sData.Email)>
<cfset oContact.setPhone(sData.Phone)>
<cfset EntitySave(oContact>
```

(note that is just for 4 fields, imagine how much longer a real world scenario would be)

To do the same thing in DataMgr would be:

```
<cfset Application.DataMgr.updateRecord("Contacts",sData)>
```

I should note, however, that you could use Bob Silverberg's **Base Persistent Object** to get this syntax:

```
<cfset oContact = EntityLoad("Contact",sData.ContactID, True)>
<cfset oContact.populate(sData)>
<cfset EntitySave(oContact>
```

Single Datasource

ColdFusion ORM allows you to use only one datasource per application. While each instance of DataMgr can use only one datasource, you can have multiple instances of DataMgr in one application (as many as you like, in fact). (**Update: Multiple Datasources** is available in ColdFusion ORM as of version 9.0.1)

Access to Source Code

DataMgr is written in CFML. You can extend it yourself. Or, if you find a bug, you can fix it yourself. With ColdFusion ORM, even if you find a bug you will have to wait for at least a hotfix to install in ColdFusion itself (not quite as easy as updating a .cfc file).

The "N+1 Select Problem"

ColdFusion ORM is more likely to run a query and then another query for every row in the first query than is DataMgr, which only does this for "list" relation fields.

- **ColdFusion ORM : What is "N+1 Select problem"**

Learning Curve

Most of these issues can be summarized by quoting a comment from **Ben Nadel** "It looks like we are really going to have to know a bit about how Hibernate works to really leverage this integration efficiently. - we can't just start using it blindly." - Learning Hibernate is a big undertaking. I would argue (keeping in mind that I am biased) that DataMgr is much, much easier to learn.

What's the Flip Side?

Of course, ColdFusion ORM has its advantages over DataMgr as well.

OO

If you are doing pure OO, then you should probably use an ORM. ColdFusion ORM is an ORM, but DataMgr isn't. Case closed.

Tested

ColdFusion 9 will be the first version to include the ORM, so expect some bugs (with the ColdFusion piece - not Hibernate itself). Even so, it has probably gotten more use and testing - already - than DataMgr has in its nearly 4 years in full release. ColdFusion ORM is also built on top of Hibernate which has a massive user base. Expect fewer bugs in ColdFusion ORM than in DataMgr.

Paging

As of now, **paging** is supported (by use off an "offset" argument) in ColdFusion ORM, but not natively supported in DataMgr. Of course, that is a known and fixable issue... (**Update:** **Efficient paging** is available in DataMgr 2.5)

Features?

I know ColdFusion 9 has plenty of other features (as does DataMgr), but I think most of them are specific to an ORM (which DataMgr isn't) and so not really valid for comparison. I think DataMgr makes many things easier than ColdFusion 9 ORM, but I have too much bias and not enough experience to make any valuable commentary on that.

What Else?

Then again, I've changed my mind before...

In the meantime, **DataMgr** is open source and free for any use if you are interested.

Links:

Learn More about ORM

- **ColdFusion ORM - An Evolution in building datacentric application**
- **ColdFusion ORM : Performance tuning - Lazy loading**
- **ORM - Rethinking ColdFusion Database Integration**
- **How Much Should Our ColdFusion Applications Actually Know About ORM?**
- **Learning ColdFusion 9: Exploring Object-Relational Mapping (ORM)**

More ORM Debates

- **Pros and Cons of an ORM**
- **http://blog.pelcosolutions.com/2009/08/coldfusion-9-orm-overkill.html**
- **CFArgument: Hibernate or Hiber-NOT :-)**