ColdFusion 9.0.1 Now Available - With ORM Goodies

Posted At : July 13, 2010 11:48 AM | Posted By : Bob Silverberg
Related Categories: ColdFusion, CF ORM Integration

ColdFusion 9.0.1 is now available for download at **http://www.adobe.com/go/getcf901**, and, in addition to fixing a number of issues with ColdFusion 9.0, it's packed full of goodies as well. The details of all the new features can be found in the **New Feature Notes**, and the bug fixes and outstanding items can be found in the **Release Notes**. I think my favourite single new feature is the ability to do a *for - in* loop with an array, as looping through an array using script has always been a pain. I'm also very happy with some of the improvements to ORM. Here's a high-level summary of the new features, followed by some details on the ORM changes:

- **Language enhancements** - including *for-in* loops for arrays.
- **New script functions implemented as CFCs** - including dbinfo, imap, pop, ldap, and feed.
- **Caching enhancements** - including the ability to get a handle on the ehCache session.
- **Support for IIS 7**
- **ColdFusion Ajax enhancements** - including updates cfmap, cfgrid, file uploading and JavaScript functions.
- **ORM enhancements** - more details below.
- **Amazon S3 support** - the ability to use Amazon S3 storage with most tags and functions that use a file or directory as input or output.
- Various other enhancements covering areas such as Spreadsheets, AIR Integration, Flash Remoting, Blaze DS, Solr, Logging, Server Monitoring, and more.

ORM Enhancements

Here's a summary of the ORM enhancements in ColdFusion 9.0.1, followed by some details about each one:

- Support for Multiple Datasources
- Transaction Management Improvements
- skipCFCWithError Flag in ormSettings
- mappedSuperClass Attribute for Components
- Use EntityNew to Populate a New Entity
- Support for HQL in cfquery

Support for Multiple Datasources

In ColdFusion 9.0, you could only use the ORM with one datasource across an entire application. Now you can have as many datasources as you like. In a multiple datasource application you tell each persistent cfc which datasource to use via the*datasource* attribute of the component. You can define ormSettings individually for each datasource in your application using a structure. For example, assuming with have datasources called Bob, Dan and Ezra, and we want each of those to have a different value for the *dbcreate* ormSetting, we would specify that like so, in our Application.cfc:

```
this.ormSettings.dbcreate = {

 Bob = "dropcreate",

 Dan = "update",

 Ezra = "none"

}
```

You can use that format for the *schema*, *catalog*, *dialect*, *dbcreate* and *sqlscript* ormSettings.

ColdFusion supports multiple ORM datasources by creating a new Hibernate Session for each datasource. One of the implications of this is that related cfcs must use the same datasource. That is, all cfcs that have relationships defined with one another must use the same datasource. Another implication of this is that, within a transaction, you may only modify entities within a single datasource. You can read in entities from another datasource inside the transaction, but if you attempt to modify entities belonging to different datasources within a single transaction an error will be thrown.

Because, in a multiple datasource application, you will now have multiple Hibernate sessions, many of the existing session management functions have been enhanced to allow you to specify which session to affect, by specifying the datasource name. The following functions now accept a datasource name, which is only applicable if you are using multiple datasources:

- ORMGetSessionFactory
- ORMGetSession
- ORMCloseSession
- ORMClearSession
- ORMFlush
- ORMEvictQueries
- ORMExecuteQuery

In addition, the following functions were added that affect all Hibernate sessions in the request:

- ORMCloseAllSessions
- ORMFlushAll

Transaction Management Improvements

In ColdFusion 9.0, there was a serious issue, in my opinion, with the way the Hibernate session was managed during transactions. ColdFusion would flush and close an existing session when you started a transaction, and would also close an existing session when a transaction ended, whether via a commit or a rollback. This has been improved in ColdFusion 9.0.1. Now, by default, when you start a new transaction, any open sessions are flushed, but they are not closed, so they are available to be used within the transaction. As well, when a transaction completes, the session is no longer closed. If the transaction was committed the session is flushed, and if the transaction is rolled back the session is cleared.

I say above that this is the new default behaviour, as you can have even more control over when flushing happens by using the new *automanageSession* ormSetting. The default value is *true* and results in the above behaviour. If you set automanageSession to *false*, however, ColdFusion will only flush the session when a transaction commits. It will neither flush nor clear a session automatically in the other scenarios (i.e., when a transaction starts or is rolled back). For this reason running with automanageSession="false" is my preference, as it gives me the greatest amount of control.

skipCFCWithError Flag in ormSettings

This flag can be set to *true* (the default is *false*) to ignore any syntax errors in cfcs when ColdFusion is searching for persistent cfcs when the application starts or the ORM is reloaded.

mappedSuperClass Attribute for Components

You can now create a base (non-persistent) object which has properties, and then extend that object with other persistent objects, which will then inherit those properties. For example, let's say that every object in my system should have an auto-generated primary key called *id* and a *dateCreated* property. I could create a base object and a user object like so:

```
component mappedSuperclass="true" hint="This is Base.cfc" {

 property name="id" fieldtype="id" generator="native";

 property name="dateCreated" ormtype="timestamp";

}


component persistent="true" extends="Base" hint="This is User.cfc" {

 property name="firstName";

 property name="lastName";

}
```

The above mappings would give me a User object with four properties: id, dateCreated, firstName and lastName.

Use EntityNew to Populate a New Entity

You can now pass a structure into the *EntityNew* function, as an optional second argument, and it will use the key-value pairs to populate the properties of the object. For example, assuming we have the User object mapped as above, we could call EntityNew like so:

```
user = EntityNew("User",{

 dateCreated = Now(),

 firstName = "Bob",

 lastName = "Silverberg"

});
```

The above code will return a new User object to me, with the dateCreated, firstName and lastName prepopulated with the values from the structure.

Support for HQL in cfquery

Rather than having to build an HQL statement using string concatenation, you can now construct your HQL statement inside a *cfquery* tag, similar to what you do for SQL queries. For example, to return all Users from our User table with a first name of *Bob*, we could write the following code:

```
<cfquery name="Users" dbtype="hql">

 from User where firstName = <cfqueryparam value="Bob">

</cfquery>
```

One thing to note about the above code is that it will return an array of User objects, not a query.

All in all I think this is an excellent release and I applaud the Adobe ColdFusion team for their efforts and their responsiveness to the ColdFusion community.