Advanced ORM
& HQL

Hemant Khandelwal
Adobe
Who am I

- Sr Engg Mgr - CF server & CF Builder
- CF8, CF9 & CFB 1
- Was expert group member on EJB3.0 & J2EE 1.4 spec committee
- hkhandel@adobe.com

And proud to have some of the very best engineers on my team!
Agenda

- ORM basics – recap
- Relationships
- HQL
- What happens internally
- Advanced Mapping
- ORM tuning parameters
ORM Basics

- **Object - Relational Mapping**
  - A persistence framework to map objects to relational database w/o writing SQL
  - Converts data between type systems in relational databases and object-oriented programming languages

- **ColdFusion ORM**
  - Based on Hibernate, one of the most mature and popular persistence framework
  - A CFC is mapped to a table
  - Powerful & easy!
Database Model for Demo

**EMPLOYEES**
- EmployeeID(PK)
- LastName
- FirstName
- Title

**ORDERS**
- OrderID(PK)
- EmployeeID
- OrderDate

**ADDRESSES**
- AddressID(PK)
- HouseNumber
- Street
- City
- State
- Country
- EmployeeID

**EMPLOYEE TERRITORIES**
- EmployeeID(PK)
- TerritoryID(PK)

**TERRITORIES**
- TerritoryID(PK)
- TerritoryDescription
Demo

ORMSettings
Mapping a Simple CFC
Relationships
CRUD Methods
How ORM is Enabled

Application Start

ORM Enabled?

Proceed with other activities

false

Create/Load Hibernate configuration if specified

true

Load Hibernate mapping files (*.hbm.xml)

Search for persistent CFCs

Generate Hibernate Mapping for persistent CFCs

Generate DDL based on dbcreate

Build Hibernate Session Factory

Proceed with other activities

Inspect

Copyright 2009 Adobe Systems Incorporated. All rights reserved. Adobe confidential.
HQL
Hibernate Query Language
HQL

- Similar to SQL
- Object-oriented
  - Understands notions like inheritance, polymorphism & association
- When do you need it?
  - Complex joins, nested WHERE clause
- How?
  - ORMExecuteQuery
  - cfquery with dbtype = "hql"
- CF logs underlying SQLs
  - Useful in debugging & perf optimization
HQL – How

- ORMExecuteQuery (hql, params, unique, queryOptions)
- `<cfquery dbtype="hql" datasource="dsn" name="q1" ormoptions=#{...}>`
  - Ormoptions – same as queryOptions
  - cfqueryparams – for params
- Case sensitive (Hibernate restriction)
  - CFC & property names
  - FirstName not same as FIRSTNAME
- Demo
HQL – Query Options

- QueryOptions in ORMExecuteQuery or ORMOptions in cfquery
  - maxResults – max objects to be retrieved
  - Offset - start index of resultset
  - Cacheable - If result to be cached in the secondary cache
  - Cachename - name of the secondary cache
  - Timeout - timeout (in seconds) for query

- Demo
Advanced Mapping

Inheritance Mapping
Embedded Mapping
Collection Mapping
Join Mapping
Using Hibernate XML
Inheritance Mapping

- Three Types
- Table per hierarchy

Payment
+ID
+Amount
+PaymentType
+Date

CreditCardPayment
+ CardNo
+ CardType

ChequePayment
+ ChequeNo
+ BankName
+ City

paymentId <<PK>>
Amount
Date
PaymentType (discriminator)
CardNo
CardType
ChequeNo
BankName
City
Inheritance Mapping

- Three Types
- Table per hierarchy
- Table per subclass

Payment
  +ID
  +Amount
  +Date

CreditCardPayment
  +CardNo
  +CardType

ChequePayment
  +ChequeNo
  +bankName
  +City

paymentId <<PK>>
Amount
Date

paymentId
CardNo
CardType

paymentId
ChequeNo
BankName
City
Inheritance Mapping

- Three Types
  - Table per hierarchy
  - Table per subclass
  - Table per subclass with discriminator
- Demo

**Payment**
- ID
- Amount
- Date

**CreditCardPayment**
- CardNo
- CardType

**ChequePayment**
- ChequeNo
- bankName
- City

**Payment Table**
- paymentId <<PK>>
- Amount
- Date
- PaymentType (discriminator)

**Payment ID**
- CardNo
- CardType

**Cheque Payment Table**
- ChequeNo
- BankName
- City
Embedded Mapping

- A cfproperty refers to another cfc
- Mapping needs to be specified in *.hbm.xml
- Name.cfc
  - firstName, lastName, title
- Employee
  - Refers to name.cfc to get additional details
- Demo

```
<table>
<thead>
<tr>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ID</td>
</tr>
<tr>
<td>+Designation</td>
</tr>
<tr>
<td>+Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+fName</td>
</tr>
<tr>
<td>+lName</td>
</tr>
<tr>
<td>+Title</td>
</tr>
</tbody>
</table>
```

EmployeeID <<PK>>
Designation

FirstName
LastName
Title
Collection Mapping

- Similar to 1:n relationship
- Useful when target table need not be mapped as persistent CFC
  - Just need information from the target table
- Mapping defined similar to 1:n relationship
- Demo
Join Mapping

- Useful when using one CFC for multiple tables

```xml
<cfcomponent persistent="true" table="Person">
  <cfproperty name="id">
  <cfproperty name="name">
  <cfproperty name="city" table="Address" joincolumn="personID">
  <cfproperty name="country" table="Address" joincolumn="personID">
</cfcomponent>
```

- Demo

![Database diagram showing the relationship between Person and Address tables]
ORM Tuning

Session mgmt & transaction
Concurrency control
Fetching Strategy
Caching
ORM Session

- Represents a unit of work – typically a transaction
  - All the ORM operations happen in a session

- Provides First Level Caching

- Tracks changes made to the objects
  - SQLs are executed when session is flushed
  - Can use ORMFlush to force

- Automatically managed by CF
  - In most cases, you don't need to worry about it
  - Manage your own using `automanageSession`
ORM Session Management

- **New Request**
  - `<cfset empObj = EntityNew("employees")>`
  - `<cfset empObj.setFirstName("John")>`
  - `<cfset empObj.setLastName("Grisham")>`
  - `<cfset EntitySave(empObj)>`

- **New ORM Session**
  - `<cfset employees = EntityLoad("employees")>`
  - `<cfdump var="#employees#">`

- **Batch all the operations**
  - `<cfset employees = EntityLoad("employees")>`
  - `<cfdump var="#employees#">`

- **Call ORMFlush Close Session**
ORM Session Management – Transactions

Application Start

New Request

```
<cfset orderObj = EntityLoad("orders", 5, true)>
<cfdump var="#orderObj#">

<cftransaction action="begin">
    <cfset empObj = EntityNew("employees")>
    <cfset empObj.setFirstName("John")>
    <cfset empObj.setLastName("Grisham")>
    <cfset EntitySave(empObj)>

    <cfset orderObj = EntityLoad("orders", 1, true)>
    <cfset EntityDelete(orderObj)>
</cftransaction>
```

New ORM Session

Call ORMFlush

Reuse ORM Session

Batch all the operations

Call ORMFlush
Manage your own Sessions

- In Application.cfc set `ormSetting.automanagesession=false`
  - Added in CF901

- Quick reference table

<table>
<thead>
<tr>
<th>Scenario</th>
<th>automanagesession = true</th>
<th>automanagesession = false</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cftransaction begin</code></td>
<td>CF flushes all session</td>
<td>No action by CF</td>
</tr>
<tr>
<td><code>cftransaction commits</code></td>
<td>CF flushes all sessions</td>
<td>CF flushes all sessions</td>
</tr>
<tr>
<td><code>cftransaction rolls back</code></td>
<td>CF clears session</td>
<td>No action by CF</td>
</tr>
<tr>
<td><code>Page request ends</code></td>
<td>CF flushes and closes session</td>
<td>CF closes session but does not flush it</td>
</tr>
</tbody>
</table>
Concurrent Control

- Optimistic lock for high concurrency
  - Update only if the entity is not modified by other thread or externally

- **optimisticlock** attribute on cfc
  - All - All properties are included in where clause of update
    - Update myTbl set col1= newVal, col2= newVal2
      where col1= oldVal and col2= oldVal2
  - Dirty - Includes only modified fields in the current session
  - Version - Checks only version or timestamp column
    - Default
  - None
Fetching Strategy

- **Immediate fetching**
  - Fetch target relationship in a separate SQL, immediately
  
  ```
  <cfproperty name="emp" fieldtype="one-to-many" cfc="order"
  fkcolumn="EMPID" lazy="false" fetch="select">
  ```

- **Lazy fetching**
  - Default strategy, `lazy=true`
  - On demand, fetch related entities
  - `Lazy = "extra"` gets pk of orders and then all order columns from db (1:n, n:n)
  - Proxy for 1:1 & n:1

- **Eager fetching**
  - Fetch together in a single SQL
  - Useful for 1:1 frequently used relationships

- **Batch fetching**
  - When fetching relationship, get some more that are requested by others
  - Use `batchsize` attribute
Caching

- **Session Level**
  - Provides first level caching
  - Next request in the same session will use cached data
  - EntityReload to refresh

- **Secondary Level**
  - Caches data across sessions
  - Cache Component, relationships & HQL results

- **Use caching for**
  - Data that changes occasionally
  - Data that is local to an application & not modified by other applications
Secondary Level caching

- **Specify in Application.cfc**
  - `ormsettings.secondarycacheenabled`
  - `ormsettings.Cacheprovider`
    - JBossCache, OSCache, SwarmCache, Hashtable, DEFAULT - ehcache
  - `ormsettings.cacheconfig`
    - Appropriate config file for cache, e.g. ehcache.xml

- **In ORM cfc**
  - "cacheuse" defines caching strategy
  - "cachename" cache region, a bucket for this data

```html
<cfcomponent persistent="true"
    cachename="foo_region" cacheuse="read-only">
```
Cache -

- Component

  ```
  <cfcomponent persistent="true"
    cachename="foo_region" cacheuse="read-only">
  ```

- Relationship

  - Primary Key of the associated object is cached
  - The associated object itself is cached if coded as above

  ```
  <cfproperty name="arts" fieldtype="one-to-many"
    cachename="foo_region" cacheuse="read-write">
  ```

- Query data

  ```
  ORMExecuteQuery("from Art where issold=0", {}, false,
    {cacheable=true, cachename="foo_region"});
  ```
Caching – cacheuse

- **Read-only**
  - Best performance for read only data

- **Nonrestrict-read-write**
  - Use when data is updated occasionally

- **Read-write**
  - Use if your data needs to be updated
  - More overhead than the two preceding strategies

- **Transactional**
  - Transactional cache
  - Can only be used if the cache provider is transaction aware
Caching - cleanup

- ORMEvictEntity
  
  ORMEvictEntity("<component_name>", [primarykey])

- ORMEvictCollection
  
  ORMEvictCollection("<component_name>", "<relation_name>", [primarykey])

- ORMEvictQueries
  
  ORMEvictQueries([cachename])
CF 901 ORM Enhancements

- Multiple datasource support
  - Application default, override in CFC

- New attribute MappedSuperClass for CFC
  - ORM CFCs inherit properties from non ORM ones

- New attributes skipCFCWithError
  - Development setting

- automanagesession

- Use **EntityNew** to create a new entity
  ```
  user = EntityNew("User",{
    firstName = "Mark",
    lastName = "Mandel"
  });
  ```

- Improved error messaging
Thank You

Questions?
Resources

- Rupesh’s blog - [http://rupeshk.org/](http://rupeshk.org/)
- Manju’s blog - [www.manjukiran.net](http://www.manjukiran.net)
Event Handling

- Set `ormsettings.eventhandling="true"`
- CFC level
  - `preinsert` and `postinsert`
  - `predelete` and `postdelete`
  - `preupdate` and `postupdate`
  - `preload` and `postload`
- Application Level
  - Set `ormsettings.eventhandler="AppEventHandler.cfc"`
  - Should implement the `CFIDE.orm.IEventHandler` interface
Auto-generating tables

- Tables created on Application startup
  - `ormsettings.dbcreate`
    - Update – create new or update if table exists
    - Dropcreate – drop and then create table
    - None – do nothing
  - `ormsettings.namingStrategy`
    - Default – CFC names matches table/column name
    - Smart – CFC "OrderProduct" is "ORDER_PRODUCT" in db
    - Your own CFC
- Your CFC
  - Implements `cfide.orm.INamingStrategy`
    - `getTableName(tableName)`
    - `getColumnColumnName(string columnName)`
Concurrency Control @ cfproperty

- Timestamp & Version properties
  
  `<cfproperty name="lastModified" 
  fieldtype="timestamp|version">

- Optimisticlock – include only this field for check
  
  `<cfproperty name="foo" 
  optimisitclock="true">

  - Default

- Automatically check during save
  
  - If check successful, value is updated