

# Migrating from Casbah 1.x -> 2.0

Brendan W. McAdams

10gen, Inc.

Dec 14, 2010



# Casbah 2.0

## What Changed?

- Now a 10gen supported driver
  - Package Change: now 'com.mongodb.casbah'
- Sanity Cleanups and Improvements
  - Modularized: Decouples some dependencies, pick and choose needed components
  - Removed 'configgy', replaced with a bare minimum 'slf4j' implementation for logging
  - Type conversions now always loaded when Commons is imported, rather than on Connection creation
  - Removal of 'global' implicit Tuple/Product ->DBObject Conversions
  - Added Scala 2.8 Collections-style DBList implementation, 'MongoDBList'
- Updated to Java Driver 2.3/2.4 w/ full API compatibility
  - Introduced Write Concern functionality including execute-around safe mode
- Significantly improved Query DSL, including support for all operators
  - Introduced Type Class Context Boundaries for DSL to allow user implementation of custom types while still maintaining type safety
  - Operators now return DBObject instead of Tuples



# Casbah 2.0

## What Changed?

- Now a 10gen supported driver
  - Package Change: now 'com.mongodb.casbah'
- Sanity Cleanups and Improvements
  - Modularized: Decouples some dependencies, pick and choose needed components
  - Removed 'configgy', replaced with a bare minimum 'slf4j' implementation for logging
  - Type conversions now always loaded when Commons is imported, rather than on Connection creation
  - Removal of 'global' implicit Tuple/Product ->DBObject Conversions
  - Added Scala 2.8 Collections-style DBList implementation, 'MongoDBList'
- Updated to Java Driver 2.3/2.4 w/ full API compatibility
  - Introduced Write Concern functionality including execute-around safe mode
- Significantly improved Query DSL, including support for all operators
  - Introduced Type Class Context Boundaries for DSL to allow user implementation of custom types while still maintaining type safety
  - Operators now return DBObject instead of Tuples



# Casbah 2.0

## What Changed?

- Now a 10gen supported driver
  - Package Change: now 'com.mongodb.casbah'
- Sanity Cleanups and Improvements
  - Modularized: Decouples some dependencies, pick and choose needed components
  - Removed 'configgy', replaced with a bare minimum 'slf4j' implementation for logging
  - Type conversions now always loaded when Commons is imported, rather than on Connection creation
  - Removal of 'global' implicit Tuple/Product ->DBObject Conversions
  - Added Scala 2.8 Collections-style DBList implementation, 'MongoDBList'
- Updated to Java Driver 2.3/2.4 w/ full API compatibility
  - Introduced Write Concern functionality including execute-around safe mode
- Significantly improved Query DSL, including support for all operators
  - Introduced Type Class Context Boundaries for DSL to allow user implementation of custom types while still maintaining type safety
  - Operators now return DBObject instead of Tuples



# Casbah 2.0

## What Changed?

- Now a 10gen supported driver
  - Package Change: now 'com.mongodb.casbah'
- Sanity Cleanups and Improvements
  - Modularized: Decouples some dependencies, pick and choose needed components
  - Removed 'configgy', replaced with a bare minimum 'slf4j' implementation for logging
  - Type conversions now always loaded when Commons is imported, rather than on Connection creation
  - Removal of 'global' implicit Tuple/Product -> DBObject Conversions
  - Added Scala 2.8 Collections-style DBList implementation, 'MongoDBList'
- Updated to Java Driver 2.3/2.4 w/ full API compatibility
  - Introduced Write Concern functionality including execute-around safe mode
- Significantly improved Query DSL, including support for all operators
  - Introduced Type Class Context Boundaries for DSL to allow user implementation of custom types while still maintaining type safety
  - Operators now return DBObject instead of Tuples



## 1 Introduction

## 2 The Details

- Modularization and Packaging
- Sanity Improvements
- Java Driver Updates
- DSL Improvements

# New Packaging

As part of the move to a 10gen supported package, Casbah's package has changed. The now defunct (it was rolled into 2.0) 1.1 branch was changing the package as well, and rolled forward.

**Table:** Casbah Packaging

Casbah 1.0.x	Casbah 1.1.x	Casbah 2.0+
com.novus.casbah.mongodb	com.novus.casbah	com.mongodb.casbah



# Modularization

Some users (such as those using a framework like Lift which already provides MongoDB wrappers) wanted access to certain parts of Casbah (e.g. MongoDBObject & MongoDBList) without importing the whole system. As a result, Casbah has been broken out into several modules which make it easier to pick and choose the features you want.





# Casbah Modules

Commons (casbah-commons)

- Module: 'casbah-commons' ("Commons") Package:  
`com.mongodb.casbah.commons`
  - Dependent upon `mongo-java-driver`, `scalaj-collection`, `scalaj-time`, `JodaTime`, `slf4j-api`
  - Provides core Scala bindings via the Conversions API (now autoloaded when Commons is imported) for converting to and from common Scala and Java types transparently
  - Provides Scala 2.8 Collections compatible wrappers for 'DBObject' and 'DBList' as 'MongoDBObject' and 'MongoDBList' respectively
- Raw import of just Commons via:

```
import com.mongodb.casbah.commons.Imports._
```



# Casbah Modules

## Query DSL (casbah-query)

- Module: 'casbah-query' ("Query DSL") Package: `com.mongodb.casbah.query`
  - Dependent upon casbah-commons along with any transitive dependencies
  - Provides the Scala syntax DSL mode for creating MongoDB query objects via '\$ Operators', e.g.:  
`"foo" $lt 50 $gte 12`
  - Using this gives you only type conversions,DBObject and DBList wrappers and the DSL.
- Raw import of just Query DSL (provides Commons automatically) via:

```
import com.mongodb.casbah.query.Imports._
```



# Casbah Modules

Core (casbah-core)

- Module: 'casbah-core' ("Core") Package:  
`com.mongodb.casbah.core`
  - Dependent upon casbah-commons and casbah-query along with their dependencies transitively
  - Provides Scala bindings to the Java driver for actually communicating with MongoDB e.g. 'DB' and 'DBCollection' and MapReduce jobs, wraps the Mongo wire format with Scala type support.
  - Importing Core will give you identical functionality to 1.x
- Raw import of Core (provides Commons and QueryDSL automatically) via:

```
import com.mongodb.casbah.Imports._
```



# Casbah Modules

## GridFS (casbah-gridfs)

- Module: 'casbah-gridfs' ("GridFS") Package:  
`com.mongodb.casbah.gridfs`
  - Dependent upon casbah-core and casbah-commons along with their dependencies transitively
  - Provides Scala enhanced wrappers to MongoDB's GridFS filesystem.
  - NOT imported by default, as many people don't use or need it. You need to explicitly import this if you want GridFS support.
- Raw import of GridFS via:

```
import com.mongodb.casbah.gridfs.Imports._
```



## 1 Introduction

## 2 The Details

- Modularization and Packaging
- **Sanity Improvements**
- Java Driver Updates
- DSL Improvements

# Removal of global Product/Tuple Conversions I

Previously, it was possible with Casbah to cast Tuples toDBObject:

```
val x: DBObject = ("foo" -> "bar", "x" -> 5, "y" -> 238.1)
```

This feature was provided by implicit conversions which attempt to target Product which is the base class of all Tuples. Buggy & unreliable: often targeted the wrong things for conversion (Such as instances of Option[\_]). As Casbah 2.0 includes wrappers for DBObject which follow Scala 2.8's Collection interfaces including builders and constructors, replaced these conversions. Previous syntax is possible by passing the Tuple pairs to MongoDBObject.apply:



# Removal of global Product/Tuple Conversions II

## code/tuple2.scala

```
val x: DBObject = MongoDBObject("foo" -> "bar", "x" -> 5, "y" ->
    238.1)
/* x: com.mongodb.casbah.Imports.DBOBJECT = { "foo" : "bar" , "x" : 5
    , "y" : 238.1} */
val y = MongoDBObject("foo" -> "bar", "x" -> 5, "y" -> 238.1)
/* y: com.mongodb.casbah.commons.Imports.DBOBJECT = { "foo" : "bar" ,
    "x" : 5 , "y" : 238.1} */
```



# Removal of global Product/Tuple Conversions III

We also provide a builder pattern which follows Scala 2.8's Map Builder:

code/tuple3.scala

```
val b = MongoDBObject.newBuilder
/* b: com.mongodb.casbah.commons.MongoDBObjectBuilder =
    com.mongodb.casbah.commons.MongoDBObjectBuilder@113f25e3 */
b += "x" -> 5
b += "y" -> 238.1
b += "foo" -> "bar"
val x: DBObject = b.result
/* x: com.mongodb.casbah.commons.Imports.DBOBJECT = { "x" : 5 , "y" :
    238.1 , "foo" : "bar" } */
```





# Removal of global Product/Tuple Conversions IV

Finally, any Scala map can still be cast to a DBObject without issue:

code/tuple4.scala

```
val x: DBObject = Map("foo" -> "bar", "x" -> 5, "y" -> 238.1)
/* x: com.mongodb.casbah.Imports.DBObject = { "foo" : "bar" , "x" : 5
    , "y" : 238.1} */
```



# Removal of global Product/Tuple Conversions V

It is still possible to use Tuples in the Query DSL however, as there is less need for broad implicit conversions to accomplish that functionality.



## 1 Introduction

## 2 The Details

- Modularization and Packaging
- Sanity Improvements
- **Java Driver Updates**
- DSL Improvements

# Various Improvements to Java Driver Support on Core

- Brought Casbah's Core API in line with Java Driver 2.3
- Support for `MongoURI`, `MongoOptions`, `slaveOK` & `WriteConcern`
- Support for Replica Sets
- Expanded many methods which previously took `DBObject` to take a View Boundary e.g.:

```
def foo[A <% DBObject] (arg: A) = { /* .. */ }
```



# Write Concern I

- Write Concern allows users to specify at a Connection, DB, Collection or individual write level what the behavior should be with regards to `w`, `wtimeout`, and `fsync`
  - `w`: # of servers MongoDB must replicate the write to before returning 'OK'. Default behavior is 0, 1 waits for write and raises any errors as exceptions. Setting it to -1 ignores even network errors.
  - `wtimeout`: # of milliseconds to wait for the write to complete. Default behavior is 0—wait forever.
  - `fsync`: When `true`, forces MongoDB to sync the write to disk before returning. Use with caution! Defaults to `false`.



# Write Concern I

## Constants

- Casbah includes a Scala convenience wrapper in `com.mongodb.casbah.WriteConcern`, along with common values as predefs
  - `WriteConcern.Normal`: Default behavior, raises exceptions for Network errors but not server errors.
  - `WriteConcern.Safe`: Exceptions are raised for network issues and server errors, Casbah blocks and waits for the write to complete (equiv. to calling `getLastError`)
  - `WriteConcern.ReplicasSafe`: Exceptions are raised for network issues and server errors, Casbah blocks and waits for the write to complete on *at least 2* servers in the replica set.
  - `WriteConcern.FsyncSafe`: Exceptions are raised for network issues and server errors, Casbah blocks and waits for the write to be flushed to disk.



# request mode I

- In the case where you want a single write to block for safety, there is `request` mode available on DB and Collection
- 'Execute Around' pattern takes a function argument, calls `requestStart` and `requestDone` then invokes `getLastError.throwOnError` which throws an exception if errors occurred. Your function is handed a copy of the DB or Collection at invocation.
- Example:

## code/request.scala

```
coll.request { c =>
  c.insert(MongoDBObject("foo" -> "bar"))
}
```



## 1 Introduction

## 2 The Details

- Modularization and Packaging
- Sanity Improvements
- Java Driver Updates
- **DSL Improvements**



# Query DSL Improvements I

## New Operators

- Added support for all currently supported operators. A list of some of the new operators added in 2.0 include:
  - \$slice
  - \$or
  - \$not
  - \$each (special operator only supported nested inside '\$addToSet')
  - \$type (Uses type arguments and class manifests to allow a nice fluid Scala syntax)
  - \$elemMatch
  - Array Operators
  - All GeoSpatial Operators including \$near and \$within



# Query DSL Improvements I

## Type Boundaries

- Pushing harder on type safety for the DSL to prevent errors
- Type Safety is good but sometimes users introduce their own types
- Using Context Bounds w/ Type Classes to expand on this.
- Most operators now take a Type Class such as `ValidDateOrNumericType`:

### code/neTypeClassDemo.scala

```
def $ne[T : ValidDateOrNumericType](target: T) = op(oper,
    target)

trait ValidDateOrNumericTypeHolder extends ValidDateTypeHolder
    with ValidNumericTypeHolder {
    implicit object JDKDateDoNotOk extends JDKDateOk with
        ValidDateOrNumericType[java.util.Date]
```



# Query DSL Improvements II

## Type Boundaries

```
implicit object JodaDateTimeDoNok extends JDKDateOk with
  ValidDateOrNumericType[org.joda.time.DateTime]
implicit object BigIntDoNok extends BigIntOk with
  ValidDateOrNumericType[BigInt]
implicit object IntDoNok extends IntOk with
  ValidDateOrNumericType[Int]
implicit object ShortDoNok extends ShortOk with
  ValidDateOrNumericType[Short]
implicit object ByteDoNok extends ByteOk with
  ValidDateOrNumericType[Byte]
implicit object LongDoNok extends LongOk with
  ValidDateOrNumericType[Long]
implicit object FloatDoNok extends FloatOk with
  ValidDateOrNumericType[Float]
implicit object BigDecimalDoNok extends BigDecimalOk with
  ValidDateOrNumericType[BigDecimal]
implicit object DoubleDoNok extends DoubleOk with
  ValidDateOrNumericType[Double]
}
```



# Query DSL Improvements III

## Type Boundaries

- Type Classes are easily expanded, if you try to pass a non-working type.

### code/newTypeClass.scala

```
"foo" $ne true
/* error: could not find implicit value for evidence parameter
   of type
   com.mongodb.casbah.query.ValidDateOrNumericType[Boolean]
   "foo" $ne true
       ^
*/

implicit object BoolOk extends
  com.mongodb.casbah.query.ValidDateOrNumericType[Boolean]
/* defined module BoolOk */

"foo" $ne true
/* com.mongodb.DBObject with
   com.mongodb.casbah.query.DSLDBObject = { "foo" : { "$ne" :
   true}} */
```



# Query DSL Improvements IV

## Type Boundaries

- Please let me know if I missed any boundaries on specific operators (e.g. I think I forgot to add Boolean support in a few places it should be, although \$ne isn't one of them)

