Scala with MongoDB

Brendan W. McAdams

Novus Partners, Inc.

New York Scala Enthusiasts
Aug. 8, 2010

Outline

Outline

NOVUS™

Who Am I?

- Started in Perl - lots of MySQL + Sybase (some PostgreSQL + Oracle)
- Spent time in C, PHP, Java, Python, C#
- Built a bond trading platform with Perl and Java
- Last few years - lots of Python
- Last year - IronPython, C#, MongoDB...
- Research into NoSQL Tools led to MongoDB

Briefly - Sluggy.com Rundown

- I've maintained systems (and some code) for Sluggy.com since 1999
- Original system was Perl + FreeBSD (flat files)
- Eventually migrated to Linux + PHP w/ MySQL
- Last Year: Rewrote with Python (Pylons) + MongoDB (pymongo + MongoKit), went live 1 year ago.
- With Mongo, serving 50GB/day (1.5TB of traffic/month on a single slicehost virtual machine)
- Opened my eyes to the new world of NoSQL
- See my talk from MongoNYC 2010 on our migration + what was learned

Current Gig

- Started at Novus Partners last fall
- Was toying with Scala while looking for work
- Dove into first project with Scala - haven't looked back.
- Existing Java development team now working in Scala
- Just expanded development team to another Scala developer
- Introduced MongoDB for rapid & flexible data interaction for frontend.
- Several Open Source packages yielded...
  - Casbah (mongoDB + Scala driver layer)
  - Luau (mongoDB + Hadoop integration, written in Scala)

Outline

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
    - Features of both Key-Value Stores & RDBMS’
    - Rich query interface.
    - Works with JSON-like Documents
    - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
    - Stored Javascript functions server-side
    - Powerful aggregation - Map/Reduce, Group Commands
    - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Introducing MongoDB

- Categorized as a “Document-Oriented Database”
  - Features of both Key-Value Stores & RDBMS’
  - Rich query interface.
  - Works with JSON-like Documents
  - Favors embedding related data over “foreign key” relationships
- Free license (A-GPL) cross-platform (Packages for Linux, Windows, Mac OS X, Windows, FreeBSD & Solaris)
- Cursor-based query results
- Serverside Javascript
  - Stored Javascript functions server-side
  - Powerful aggregation - Map/Reduce, Group Commands
  - JS Statements in queries (no indexes though)
- Indexing system is much like RDBMS, includes Geospatial support.
- Scalable file storage with GridFS
- Data scalability with Replica Sets & Autosharding

Programming with MongoDB

- Provides a native API which allows interaction to adapt to the programming language (rather than vice versa).
- Official drivers for...
  - C
  - C++
  - Java
  - JavaScript
  - Perl
  - PHP
  - Python
  - Ruby
- Community supported drivers include...
  - .Net: C# & F#
  - JVM: Clojure, Scala, Groovy
  - Erlang
  - Haskell
  - Go
  - ... and many more.

Programming with MongoDB

- Provides a native API which allows interaction to adapt to the programming language (rather than vice versa).
- Official drivers for...
  - C
  - C++
  - Java
  - JavaScript
  - Perl
  - PHP
  - Python
  - Ruby
- Community supported drivers include...
  - .Net: C# & F#
  - JVM: Clojure, Scala, Groovy
  - Erlang
  - Haskell
  - Go
  - ... and many more.

Programming with MongoDB

- Provides a native API which allows interaction to adapt to the programming language (rather than vice versa).
- Official drivers for...
  - C
  - C++
  - Java
  - JavaScript
  - Perl
  - PHP
  - Python
  - Ruby
- Community supported drivers include...
  - .Net: C# & F#
  - JVM: Clojure, Scala, Groovy
  - Erlang
  - Haskell
  - Go
  - ... and many more.

But is anyone actually *using* it?!?

MongoDB is deployed in production at companies including...

- New York Times
- Foursquare
- bit.ly
- SourceForge
- Etsy
- Disqus
- Github
- ... The Large Hadron Collider.

But is anyone actually *using* it?!?

MongoDB is deployed in production at companies including...

- New York Times
- Foursquare
- bit.ly
- SourceForge
- Etsy
- Disqus
- Github
- ... The Large Hadron Collider.

Outline

NOVUS

The basics of Querying

Query Objects

Geospatial Support

Finally, Data Scalability.

- Replica Sets
- AutoSharding

Outline

Using Scala with the official Java Driver I

- JVM Object are JVM Objects...

```
import com.mongodb._

val conn = new Mongo()
val db = conn.getDB("test")
val coll = db.getCollection("testData")

val pies = new BasicDBList()
pies.add("cherry")
pies.add("blueberry")
pies.add("apple")
pies.add("rhubarb")
pies.add("3.14")

val doc = new BasicDBObject()
doc.put("foo", "bar")
doc.put("spam", "eggs")
doc.put("up", "down")
doc.put("pie", pies)

coll.insert(doc)
```

- ... Not terribly “Scala-ey”.

Using Scala with the official Java Driver II

- The Java driver works, but doesn't fit well in Scala.
- You need to convert your Scala objects to Java Objects, and get nothing but Java Objects out.
- Gets messy quickly.

The Scala Community Adapted... I

Compare the previous with various Scala drivers.

- mongo-scala-driver wraps & enhances the Java driver:

```
import com.mongodb._
import com.osinka.mongodb._

val conn = new Mongo()
val db = conn.getDB("test")
val coll = db.getCollection("testData").asScala

coll << Map(
  "foo" -> "bar",
  "spam" -> "eggs",
  "up" -> "down",
  "pie" -> List(
    "cherry",
    "blueberry",
    "apple",
    "rhubarb",
    "3.14"
  )
)
```

The Scala Community Adapted... II

- .. Much better, although I was confused initially. Has a object<->MongoDB mapping layer.
- lift-mongodb has more than one way to do it... here's just a taste:

```
import com.mongodb._

import net.liftweb.mongodb._
import net.liftweb.json._
import net.liftweb.json.JsonAST.JObject
import net.liftweb.json.JsonDSL._

implicit val formats = DefaultFormats.lossless

MongoDB.defineDb(DefaultMongoIdentifier,
                MongoAddress(MongoHost("localhost", 27017)), "test")

val json = JsonParser.parse("""
{ "foo": "bar",
  "spam": "eggs",
  "up": "down",
  "pie": [
    "cherry",
    "blueberry",
    "apple",
    "rhubarb",
    "3.14"
```

The Scala Community Adapted... III

```
  ]
}
""").asInstanceOf[JObject]

MongoDB.useCollection("testData")( coll => {
  coll.save(JObjectParser.parse(json))
})
```

- ... Lift's JS & JSON tools make it very flexible, as we'll see later. Also has an ActiveRecord style Object<->MongoDB Mapping layer.
- Casbah reflects my own attempt at creating a sane interface between Scala & MongoDB. Influenced by pymongo:

NOVUS

The Scala Community Adapted... IV

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

val coll = MongoConnection()("test")("testData")

val builder = MongoDBObject.newBuilder
builder += "foo" -> "bar"
builder += "spam" -> "eggs"
builder += "up" -> "down"
builder += "pie" -> List("cherry", "blueberry",
                         "apple", "rhubarb", "3.14")

coll += builder.result
```

- ... The syntax is still growing but is meant to match Scala syntax sanely. Object<->MongoDB Mapping coming soon.
- We're going to cover several tools, although I know Casbah best.

Outline

Helping Java + Scala Interact

- Implicits, “Pimp My Library” and various conversion helper tools simplify the work of interacting with Java.
- Scala and Java have their own completely different collection libraries.
- Some builtins ship with Scala to make this easier.

Helping Java + Scala Interact

- Implicits, “Pimp My Library” and various conversion helper tools simplify the work of interacting with Java.
- Scala and Java have their own completely different collection libraries.
- Some builtins ship with Scala to make this easier.

Helping Java + Scala Interact

- Implicits, “Pimp My Library” and various conversion helper tools simplify the work of interacting with Java.
- Scala and Java have their own completely different collection libraries.
- Some builtins ship with Scala to make this easier.

Interoperability in Scala 2.7.x

- Scala 2.7.x shipped with `scala.collection.jcl`.
- `scala.collection.jcl.Conversions` contained some implicit converters, but only to and from the wrapper versions - no support for "real" Scala collections.
- Neglected useful base interfaces like `Iterator` and `Iterable`
- @jorgeortiz85 provided `scala-javautils`, which used "Pimp My Library" to do a better job.

Interoperability in Scala 2.7.x

- Scala 2.7.x shipped with `scala.collection.jcl`.
- `scala.collection.jcl.Conversions` contained some implicit converters, but only to and from the wrapper versions - no support for “real” Scala collections.
- Neglected useful base interfaces like `Iterator` and `Iterable`
- @jorgeortiz85 provided `scala-javautils`, which used “Pimp My Library” to do a better job.

Interoperability in Scala 2.7.x

- Scala 2.7.x shipped with `scala.collection.jcl`.
- `scala.collection.jcl.Conversions` contained some implicit converters, but only to and from the wrapper versions - no support for “real” Scala collections.
- Neglected useful base interfaces like `Iterator` and `Iterable`
- @jorgeortiz85 provided `scala-javautils`, which used “Pimp My Library” to do a better job.

Interoperability in Scala 2.8.x

- Scala 2.8.x improves the interop game significantly.
- JCL is gone - focus has shifted to proper interoperability w/ built-in types.
- `scala.collection.jcl.Conversions` replaced by `scala.collection.JavaConversions` - provides implicit conversions to & from Scala & Java Collections.
- Includes support for the things missing in 2.7 (`Iterable`, `Iterator`, etc.)
- Great for places where the compiler can guess what you want (implicits); falls short in some cases (like BSON Encoding, as we found in Casbah)
- @jorgeortiz85 has updated `scala-javautils` for 2.8 with `scalaj-collection`
- Explicit `asJava` / `asScala` methods for conversions. Adds `foreach` method to Java collections.

Interoperability in Scala 2.8.x

- Scala 2.8.x improves the interop game significantly.
- JCL is gone - focus has shifted to proper interoperability w/ built-in types.
- `scala.collection.jcl.Conversions` replaced by `scala.collection.JavaConversions` - provides implicit conversions to & from Scala & Java Collections.
- Includes support for the things missing in 2.7 (`Iterable`, `Iterator`, etc.)
- Great for places where the compiler can guess what you want (implicits); falls short in some cases (like BSON Encoding, as we found in Casbah)
- @jorgeortiz85 has updated `scala-javautils` for 2.8 with `scalaj-collection`
- Explicit `asJava` / `asScala` methods for conversions. Adds `foreach` method to Java collections.

Interoperability in Scala 2.8.x

- Scala 2.8.x improves the interop game significantly.
- JCL is gone - focus has shifted to proper interoperability w/ built-in types.
- `scala.collection.jcl.Conversions` replaced by `scala.collection.JavaConversions` - provides implicit conversions to & from Scala & Java Collections.
- Includes support for the things missing in 2.7 (`Iterable`, `Iterator`, etc.)
- Great for places where the compiler can guess what you want (implicits); falls short in some cases (like BSON Encoding, as we found in Casbah)
- @jorgeortiz85 has updated `scala-javautils` for 2.8 with `scalaj-collection`
- Explicit `asJava` / `asScala` methods for conversions. Adds `foreach` method to Java collections.

Interoperability in Scala 2.8.x

- Scala 2.8.x improves the interop game significantly.
- JCL is gone - focus has shifted to proper interoperability w/ built-in types.
- `scala.collection.jcl.Conversions` replaced by `scala.collection.JavaConversions` - provides implicit conversions to & from Scala & Java Collections.
- Includes support for the things missing in 2.7 (`Iterable`, `Iterator`, etc.)
- Great for places where the compiler can guess what you want (implicits); falls short in some cases (like BSON Encoding, as we found in Casbah)
- @jorgeortiz85 has updated `scala-javautils` for 2.8 with `scalaj-collection`
- Explicit `asJava` / `asScala` methods for conversions. Adds `foreach` method to Java collections.

So WTF is an ‘Implicit’, anyway?

- Implicit Arguments
  - ‘Explicit’ arguments indicates a method argument you pass, well *explicitly*.
  - ‘Implicit’ indicates a method argument which is... *implied*. (But you can pass them explicitly too.)
  - Implicit arguments are passed in Scala as an additional argument list:

```
import com.mongodb._
import org.bson.types.ObjectId

def query(id: ObjectId)(implicit coll: DBCollection) = coll.findOne(id)

val conn = new Mongo()
val db = conn.getDB("test")
implicit val coll = db.getCollection("testData")

// coll is passed implicitly
query(new ObjectId())

// or we can override the argument
query(new ObjectId())(db.getCollection("testDataExplicit"))
```

- How does this differ from default arguments?

So WTF is an ‘Implicit’, anyway?

- Implicit Arguments
  - ‘Explicit’ arguments indicates a method argument you pass, well *explicitly*.
  - ‘Implicit’ indicates a method argument which is... *implied*. (But you can pass them explicitly too.)
  - Implicit arguments are passed in Scala as an additional argument list:

```
import com.mongodb._
import org.bson.types.ObjectId

def query(id: ObjectId)(implicit coll: DBCollection) = coll.findOne(id)

val conn = new Mongo()
val db = conn.getDB("test")
implicit val coll = db.getCollection("testData")

// coll is passed implicitly
query(new ObjectId())

// or we can override the argument
query(new ObjectId())(db.getCollection("testDataExplicit"))
```

  - How does this differ from default arguments?

So WTF is an ‘Implicit’, anyway?

- Implicit Methods/Conversions
  - If you try passing a type to a Scala method argument which doesn’t match...

```
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212") // won’t compile
```

  - A fast and loose example, but simple. Fails to compile.
  - But with implicit methods, we can provide a conversion path...

```
implicit def strToNum(x: String) = x.toInt
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212")
```

  - In a dynamic language, this may be called “monkey patching”. Unlike Perl, Python, etc. Scala resolves implicits at compile time.

NOVUS

So WTF is an ‘Implicit’, anyway?

- Implicit Methods/Conversions
  - If you try passing a type to a Scala method argument which doesn’t match...

```
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212") // won't compile
```

  - A fast and loose example, but simple. Fails to compile.
  - But with implicit methods, we can provide a conversion path...

```
implicit def strToNum(x: String) = x.toInt
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212")
```

  - In a dynamic language, this may be called “monkey patching”. Unlike Perl, Python, etc. Scala resolves implicits at compile time.

NOVUS

So WTF is an 'Implicit', anyway?

- Implicit Methods/Conversions
  - If you try passing a type to a Scala method argument which doesn't match...

```
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212") // won't compile
```

  - A fast and loose example, but simple. Fails to compile.
  - But with implicit methods, we can provide a conversion path...

```
implicit def strToNum(x: String) = x.toInt
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212")
```

  - In a dynamic language, this may be called "monkey patching". Unlike Perl, Python, etc. Scala resolves implicits at compile time.

So WTF is an ‘Implicit’, anyway?

- Implicit Methods/Conversions
  - If you try passing a type to a Scala method argument which doesn’t match...

```
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212") // won't compile
```

  - A fast and loose example, but simple. Fails to compile.
  - But with implicit methods, we can provide a conversion path...

```
implicit def strToNum(x: String) = x.toInt
def printNumber(x: Int) = println(x)

printNumber(5)
printNumber("212")
```

  - In a dynamic language, this may be called “monkey patching”. Unlike Perl, Python, etc. Scala resolves implicits at compile time.

Pimp My Library

- Coined by Martin Odersky in a 2006 Blog post. Similar to C# extension methods, Ruby modules.
- Uses implicit conversions to tack on new methods at runtime.
- Either return a new “Rich_” or anonymous class...

```
import com.mongodb.gridfs.{GridFSInputFile => MongoGridFSInputFile}

class GridFSInputFile protected[mongodb](override val underlying:
     MongoGridFSInputFile) extends GridFSFile {
  def filename_=(name: String) = underlying.setFilename(name)
  def contentType_=(cT: String) = underlying.setContentType(cT)
}

object PimpMyMongo {
  implicit def mongoConnAsScala(conn: Mongo) = new {
    def asScala = new MongoConnection(conn)
  }

  implicit def enrichGridFSInput(in: MongoGridFSInputFile) =
    new GridFSInputFile(in)
}

import PimpMyMongo._
```

NOVUS

Pimp My Library

- Coined by Martin Odersky in a 2006 Blog post. Similar to C# extension methods, Ruby modules.
- Uses implicit conversions to tack on new methods at runtime.
- Either return a new “Rich_” or anonymous class...

```
import com.mongodb.gridfs.{GridFSInputFile => MongoGridFSInputFile}

class GridFSInputFile protected[mongodb](override val underlying:
      MongoGridFSInputFile) extends GridFSFile {
  def filename_=(name: String) = underlying.setFilename(name)
  def contentType_=(cT: String) = underlying.setContentType(cT)
}

object PimpMyMongo {
  implicit def mongoConnAsScala(conn: Mongo) = new {
    def asScala = new MongoConnection(conn)
  }

  implicit def enrichGridFSInput(in: MongoGridFSInputFile) =
    new GridFSInputFile(in)
}

import PimpMyMongo._
```

Pimp My Library

- Coined by Martin Odersky in a 2006 Blog post. Similar to C# extension methods, Ruby modules.
- Uses implicit conversions to tack on new methods at runtime.
- Either return a new "Rich_" or anonymous class...

```
import com.mongodb.gridfs.{GridFSInputFile => MongoGridFSInputFile}

class GridFSInputFile protected[mongodb](override val underlying:
     MongoGridFSInputFile) extends GridFSFile {
  def filename_=(name: String) = underlying.setFilename(name)
  def contentType_=(cT: String) = underlying.setContentType(cT)
}

object PimpMyMongo {
  implicit def mongoConnAsScala(conn: Mongo) = new {
    def asScala = new MongoConnection(conn)
  }

  implicit def enrichGridFSInput(in: MongoGridFSInputFile) =
    new GridFSInputFile(in)
}

import PimpMyMongo._
```