Software Developer's JOURNAL

new ideas & solutions for professional programmers

Vol.1 No.2 Monthly Issue 2/2013 (2) ISSN 1734-3933

OPEN

Programming Tips&Tricks

USING MONGODB FOR DATA REPLICATION FOR DISASTER RECOVERY

GRIDFS – STORE AND STREAM FILES WITH MONGODB

COMBINING PHP WITH JAVA IN THE SAME APPLICATION USING PHP/JAVA BRIDGE

LOCALIZATION WEBSITE WITH PHP AND GETTEXT (L10N)

IOS SOFTWARE DEVELOPMENT WITH ADOBE AIR

RFC FOR A NEW INIT MECHANISM

ACUNU ANALYTICS

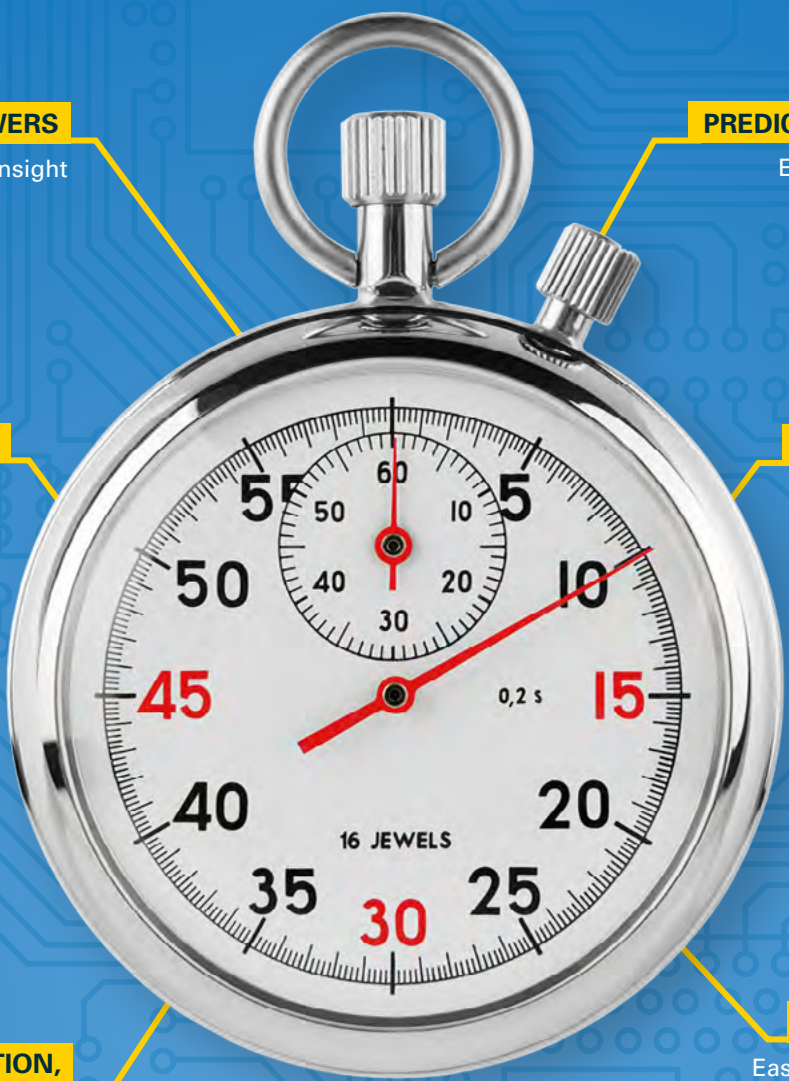
WHEN SECONDS MATTER... WE PROVIDE IMMEDIATE BUSINESS INSIGHT

LOW-LATENCY ANSWERS
Sub-second from source to insight

PREDICTABLE FAST QUERIES
Even as data volumes grow

HISTORICAL, FRESH DATA
Combined for deeper insight

POWERFUL ANALYTICS
Qualitative and quantitive



RICH VISUALIZATION
Easy to assemble dashboards and APIs

SCALABILITY, DISTRIBUTION, AND AVAILABILITY
Round the world and into the peta-scale

Acunu Analytics on Cassandra delivers instant answers from event streams, making it simple to build analytic applications and live dashboards that deliver fresh insight from live data.

Our unique approach of pre-building the cubes, on ingest, that satisfy the queries, means predictable, very short query times that are not dependent on data volumes.

There is no lag introduced by batching up events on arrival nor by indexing loaded data before it is available to queries. ***Results with millisecond latency.***

Read about our customers' success and hear them talk in our webinars at www.acunu.com

www.acunu.com | @acunu
contact@acunu.com | US office: +1 866-487-2650 | UK office: +44 203-1760143

Acunu

Big data goes in. Business insight comes out.

Our software turns your massive streams of machine data into business insights by making sense of website clickstreams, mobile devices, applications and other technologies that drive your business. It's what our market-leading customers call real-time Operational Intelligence.

Over half of the Fortune 100™ use Splunk software and have the business results to prove it. In days or weeks, not months or years.

www.splunk.com/goto/listen

www.splunk.com

splunk> listen to your data™

© 2013 Splunk Inc. All Rights Reserved. Fortune 100 is a trademark of the FORTUNE magazine division of Time Inc.

Dear Readers,

A new issue of SDJ Magazine is released! This time it is Open version for all our members, available after registration. Please just open it and explore articles, stories and resources for programmers including all useful advices as well as tips and tricks. Please start from the beginning and read the article about how to use MongoDB for data replication for disaster recovery, written by Steve Francis. When talking about disaster recovery and backups, MongoDB is not the first tool you would think of deploying to address that issue. However, it is exactly the tool that authors reached for to solve this problem. In the next article, GridFS – Store and Stream files with MongoDB, the author Stephane Godbillon tells you what GridFS is and how it works. What will you need to do to share a GridFS store and how to use GridFS to stream files from/to MongoDB with ReactiveMongo? This article answers all those questions.

If you want to know more about PHP language, the next 2 articles are definitely for you. Please read: Combining PHP with Java in the Same Application Using PHP/Java Bridge by Octavia Andrea Anghel and Localization Website with PHP and GETTEXT (L10N) by Claudio Corlatti. If you like them, you need to read the whole issue dedicated to PHP and you will find them on our website.

For those of you who are extremely interested in iOS solution, I recommend the article written by Sosenyuk Oleg about how to develop iOS software with Adobe AIR. Our author claims that Adobe AIR is a great tool for porting games to mobile devices or even to design directly for them. AIR technology allows running your flash game on IOS, Android and yes ... BlackBerry with small modifications in your code.

The last article: RFC for a New Init Mechanism (Another Way to Initialize Objects in Objective C) written by Yannick Cadin presents how to write one's own initialization methods in an Objective-C program while respecting the Apple – originally NeXT – because the official writing rules can be a tedious task.

As always, we have a couple of good articles for you. I hope you will find them useful and practical.

I thank the Beta Testers and Proofreaders for their excellent work and dedication to help make this magazine even better. Special thanks to all authors that help me create each issue. Please keep up the great work and send in your articles, tutorials and product reviews, questions, ideas or advises.

Enjoy reading!

Ewa & SDJ team

Software Developer's JOURNAL
new ideas & solutions for professional programmers

team

Editor in Chief: Anna Lakomy
anna.lakomy@sdjournal.org

Editorial Advisory Board:
Laszlo Acs, Dawid Esterhuien

Special thanks to Raul Mesa

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Paweł Marciniak

Managing Director: Ewa Dudzic

Production Director: Andrzej Kuca
andrzej.kuca@sdjournal.org

Art. Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@sdjournal.org

DTP: Ireneusz Pogroszewski

Marketing Director: Ewa Dudzic

Publisher: Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Phone: 1 917 338 3631
http://en.sdjournal.org/

Whilst every effort has been made to ensure the highest quality of the magazine, the editors make no warranty, expressed or implied, concerning the results of the content's usage. All trademarks presented in the magazine were used for informative purposes only.

All rights to trademarks presented in the magazine are reserved by the companies which own them.

DISCLAIMER!
The techniques described in our magazine may be used in private, local networks only. The editors hold no responsibility for the misuse of the techniques presented or any data loss.

MONGODB

PHP

GETTING STARTED

Using MongoDB for Data Replication for Disaster Recovery

When talking about disaster recovery and backups, MongoDB is not the first tool you would think of deploying to address that issue. However, it is exactly the tool that our TechOps team reached for to solve this problem at LogicMonitor, and it has proven very well suited to this uncommon use.

When talking about disaster recovery and backups, MongoDB is not the first tool you would think of deploying to address that issue. However, it is exactly the tool that our TechOps team reached for to solve this problem at LogicMonitor, and it has proven very well suited to this uncommon use.

A brief bit of background: LogicMonitor runs a SaaS based monitoring service, and provides monitoring insight into the performance, availability and capacity of tens of thousands of hosts, for hundreds of customers, across all sorts infrastructure (on premise, in cloud, and hybrid).

We wanted copies of customer data files both within a data center and outside of it. The former was to protect against the loss of individual servers within a facility, and the latter for recovery in the event of the complete loss of a data center.

Our requirements for a backup system were mainly around simplicity, automation and 'assurability'. We're a rapidly growing company, so we didn't have people we could spare to be managing and checking on a backup process. We required a system to easily deal with:

- multiple datacenters
- a frequently changing set of systems, with new ones being regularly added, and old ones occasionally being retired and used for lab or development systems
- an evergrowing set of customers (and hence data),
- customer accounts frequently being migrated to a different set of servers, or even a different datacenter location

Most importantly, however, we needed the backup system to be trusted – which means we needed to be able to monitor it thoroughly. So long as it was monitored, we could 'set it and forget it', relying on the monitoring to alert us when it needed attention, or if a customer had not been backed up or replicated offsite when it should have been.

Where we were: Rsync

Like most everyone who starts off in a Linux environment, we initially used rsync to copy data around (Figure 1).

Rsync (scheduled via cron) is tried and tested, and works well when the number of servers, the amount of data, and the number of files is not horrendous. But we rapidly outgrew those constraints, and were running into issues such as:

- backup job times increasing, leading to:
- backup jobs overlapping, leading to:
- too many simultaneous jobs overloading servers or network, and
- no easy way to monitor job counts, job statistics, and get alerted on failures.

Our experience running infrastructure for our own and prior SaaS companies has taught us that everything in your infrastructure needs to be monitored, so the inability to easily monitor the status of rsync jobs was particularly vexing. We needed to get better statistics and alerting, both to be confident that every customer was being correctly backed up, but also to be able to put logic into the jobs themselves to prevent issues like too many running simultaneously.

A database repository for backup job metadata, where jobs themselves can report their status, and where other backup components can get information in order to

coordinate tasks such as removing old jobs, was clearly needed. It would also enable us to monitor backup job status via simple queries for information such as the number of jobs running (total, and on a per-server basis), the time since the last backup, the size of the backup jobs, etc.

MongoDB as a Backup Job Metadata Store

The type of backup job statistics was likely going to evolve over time, so MongoDB, with its "schemaless" document store design, seemed appropriate. It seemed the perfect fit: easy to setup, easy to query, schemaless, and a simple JSON style structure for storing job information. As an added bonus, MongoDB replication is excellent: it is robust and extremely easy to implement and maintain.

So the first idea was to keep using rsync for data replication, but track the status of jobs in MongoDB. This was not an ideal solution, though – it required wrapping all sorts of reporting and querying logic into scripts surrounding rsync. The backup job metainfo and the actual backed up files were still separate and decoupled: the metadata in MongoDB and the backed up files were resident on different systems, in some cases, leading to potential issues with coherence. Further, replicating between datacenters was a completely separate process than replicating within the datacenter – requiring duplicated rsyncs and time windows, and again leading to possible issues where the metadata differed from the actual replicated data.

Having some experience now with MongoDB, we realized how nice it would be if the data and the database were combined; if we could query for a specific backup job, then use the same query language for an actual backed up file and get it; if restoring data files was just a simple query away… Enter GridFS.

Why GridFS

GridFS is a simple file system overlay on top of MongoDB, that allows files of practically unlimited size to be stored within a MongoDB database. Normal documents in MongoDB are limited to 16MB, but GridFS divides a file into chunks, and stores each of those as a separate document. GridFS is not a POSIX file system, but it does make it simple to copy files to and from a GridFS system via the mongo drivers that are available for many languages. We scripted our backup system in Ruby, so the act of copying files and their associated metadata into the GridFS system is as simple as: Listing 1.

So now, our backup scripts store the data and the metadata into MongoDB at the same time and into the same place, so everything is easily queried (Listing 2).

Replicable Advantages

Of course GridFS, lying on top of MongoDB, takes advantage of MongoDB replication. This means backed

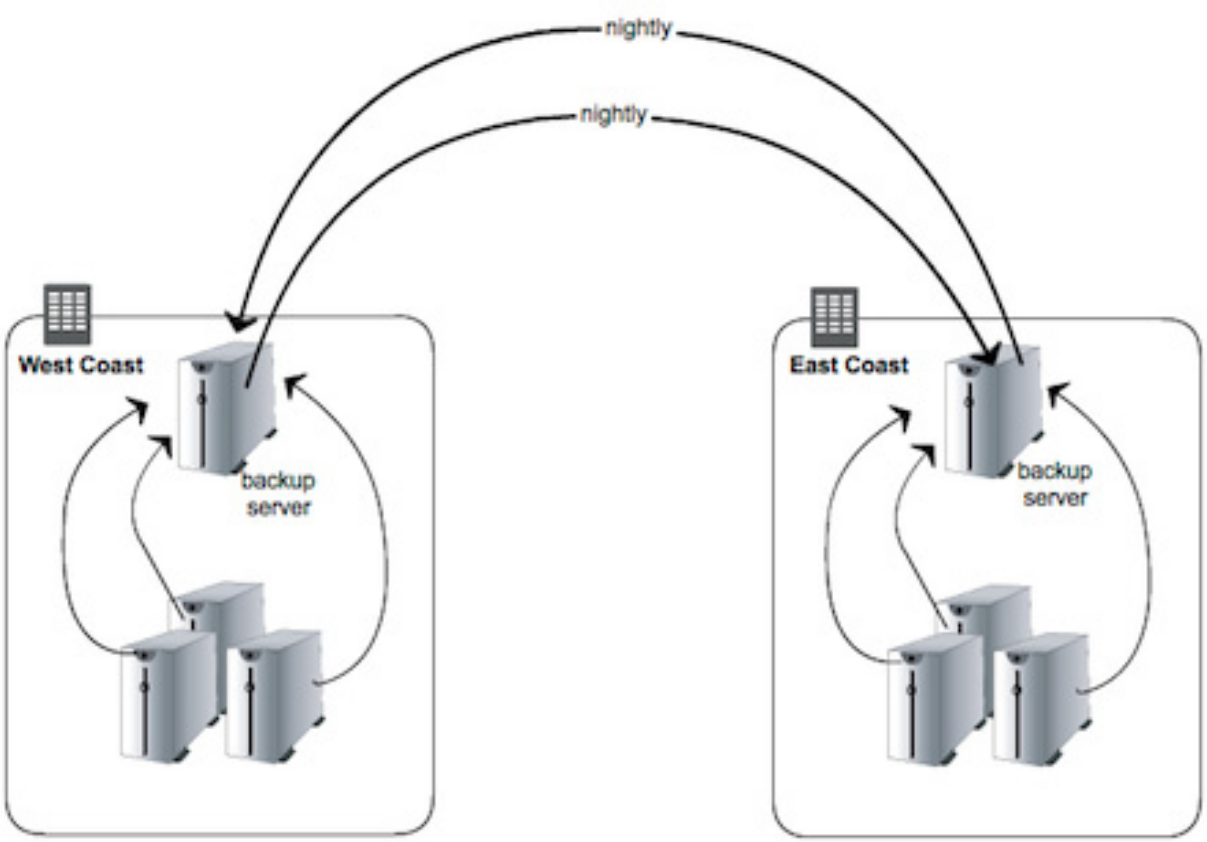


Figure 1. *Original Rsync based backups*

up files are immediately replicated both within the data center and off-site. We employ a MongoDB replica set per datacenter, and set the replica priorities so that primary replica (which controls writes) is always local to the datacenter, but also ensure there is at least one replica in a remote data center. With a replica inside of Amazon EC2, we use EBS snapshots (point in time, frozen views of the file system, replicated to Amazon's S3 service) to keep as many historical backups as desired. Our setup now looks like this: Figure 2.

Listing 1. *Snippet to copy a tar file into GridFS*

```
def backup_file(tar, name)
    # some exceptions we might want to re-try
    retries = 0

    begin
      start  = Time.now
      @grid.put(tar,
      :filename   => name,
      :hostname  => @job_info.hostname,
      :start_time   => start,
      :job_id    => self.job_info.job_id,
      :safe      => false
      )
    end

  end
```

Listing 2. *Backup job metadata*

```
util:PRIMARY> db.backup_jobs.
                    find({company:"ryanmerrill"}).
                    limit(10)[0]
{
    "_id" : ObjectId("5161ae8f3be1d56848000001"),
    "company" : "ryanmerrill",
    "hostname" : "demo1.iad.logicmonitor.net",
    "job_id" : ObjectId("5161ae8f3be1d56848000001"),
    "restore_status" : null,
    "status" : "completed",
    "status_info" : null,
    "time_end" : ISODate("2013-04-07T17:36:18.015Z"),
    "time_start" : ISODate("2013-04-
                    07T17:36:15.766Z"),
    "meta_info" : {
       "file_count" : 2,
       "total_size" : 19614481
    },
    "heartbeat" : ISODate("2013-04-07T17:36:18.015Z")
}
```

This set up gives us most of what we want:

- job status information available via simple queries,
- backup jobs themselves (including files) can be retrieved and deleted via queries,
- replication to off-site location is practically immediate,
- sharding is now possible,
- with EBS volumes, MongoDB backups (metadata AND actual backed up data) via snapshots is easy and limitless,
- automated monitoring of status is easy.

This lets us ensure that every customer has a valid set of their data within GridFS, replicated to all datacenters, within the defined tolerance for backup frequency that we set. Further, we can ensure that the most recent backup is replicated into EC2 (essentially a remote datacenter); and snapshotted onto an EBS snapshot. We keep only the most recent backup within the GridFS system. As soon as a more recent backup has been completed into GridFS, and replicated, we delete the older backup for that customer. (They are still available as EBS snapshots if we need to restore to a prior point in time).

So what's not to like?

There are a few issues with using GridFS in this manner – none insurmountable.

Bandwidth Usage

The first thing we came across was not something you'd usually think of as a problem – there was no rate limit on the replication of data, and the bandwidth achievable by GridFS/MongoDB replication is quite high. While replication of data within datacenter is effectively free, we pay for Internet transit usage, billed at 95 percentile of traffic Mbps, for traffic that transits between our datacenters. We had been using the rate limiting features of rsync to throttle down the backup transfer rates between data centers to limit cost. MongoDB and GridFS do not implement a rate limiting mechanism, so we saw our ISP fee's jump. While this could have been addressed using iptables and its traffic control system, we instead just elected to let the replication run full speed for simplicity and improved recoverability. We generally get 40Mbps of replication traffic sustained from a West coast to an East coast datacenter. Luckily our customer base is now large enough that this cost is no longer significant, and the benefit of immediate replication is well worth it.

Security

While MongoDB now supports SSL transport, and keyfile authentication, security is always a concern. As noted on the MongoDB site, "the most effective ways to

control access and to secure the connection between members of a *replica set* depend on network-level access control. Use your environment's firewall and network routing to ensure that traffic *only* from clients and other replica set members can reach your *mongod* instances." We certainly implemented the security controls, but also ensured that the files we were copying into GridFS were either encrypted in advance, or contained data that was already encrypted. In our case, virtually all data we host is encrypted already, so this was not a great concern.

Space Usage

Our customers are different sizes (some monitor 5 systems; some monitor 5000 systems) with differing complexity in those systems (monitoring a storage array with 2000 volumes involves considerably more data than a Windows server with a single IIS instance and one physical drive.) Which means no two customers – nor their backups – are alike. Initially, we ran into issues where the MongoDB disk usage would just keep growing. Like other databases, MongoDB does not release the space from deleted objects back to the operating system – it will keep it allocated, note that it is free, and reuse it if it finds a similar size object it can put in there later. In practice, we found that this space reuse did not occur very efficiently. As we are always adding and deleting objects – we actually replace the entire contents of the MongoDB database several times a day, as backups for every customer are replicated across datacenters multiple times a day - we found a great deal of fragmentation in the underlying MongoDB. We added code to try to 'pre-chunk' our backup files into consistent sizes where possible, but even so, we found that the space re-use was not very good. We worked around this by running a compact operation on the database nightly. Each replica – including the primary - runs a compact operation (at different times), which will consolidate all the freed space. This stopped the constant growth of file system usage – although at a cost of about 40 minutes a night when backups are halted. If this was a bigger impact, we could address that – Mongo now supports allocating space for documents on disk in sizes that are powers of 2 (see the usePowerOf2Sizes collMod flag), which should considerably help space reuse. It would also be fairly easy to force another node to become primary, run the compact, then resume the primary role, thus alleviating any pauses in backups at all. However, we haven't felt the need to implement these tweaks as yet.

EBS Snapshot Performance

Not at all Mongo specific, but we've found that, even with reserved EBS performance, triggering snapshots

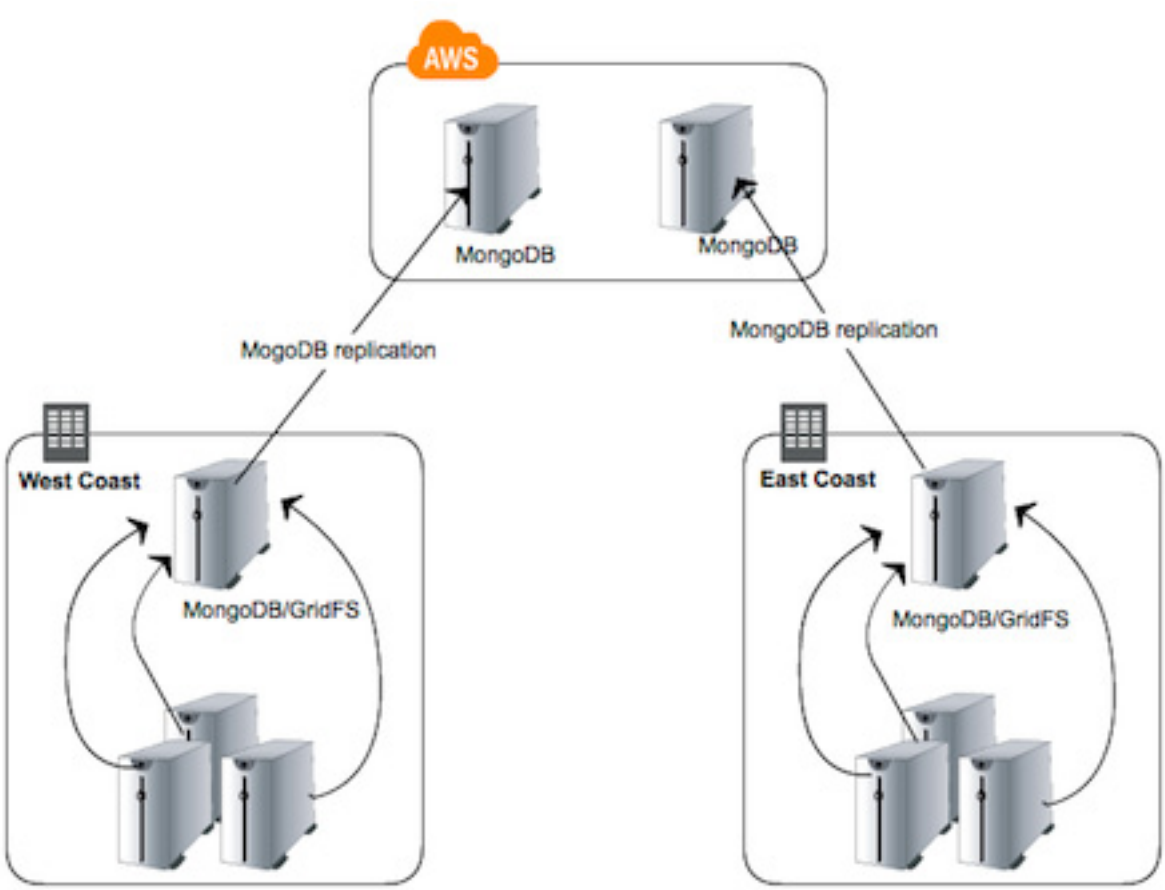


Figure 2. *GridFS based replication*

on Amazon EBS volumes kills the IOps performance, which leads to significant lag in the replication onto disk in EC2. In the below graph, you can see the compaction occurred at midnight, then at 2:00 am a snapshot was triggered, leading to a period of increased replication lag (Figure 3).

However, as most of our infrastructure is not within EC2, but on our own servers in our own datacenters, this is not an issue we've needed to address yet either.

Goals being met

Revisiting our initial needs, we have certainly met them. Whenever systems are deployed, in any datacenter, with customers on them, the system provisioning scripts (based on Puppet, in our case) automatically configure the system as a client for the MongoDB based backup system. Because the servers can use simple MongoDB queries to ask the backup system things like "When was this customer last backed up? How many active jobs are running right now?" it's very

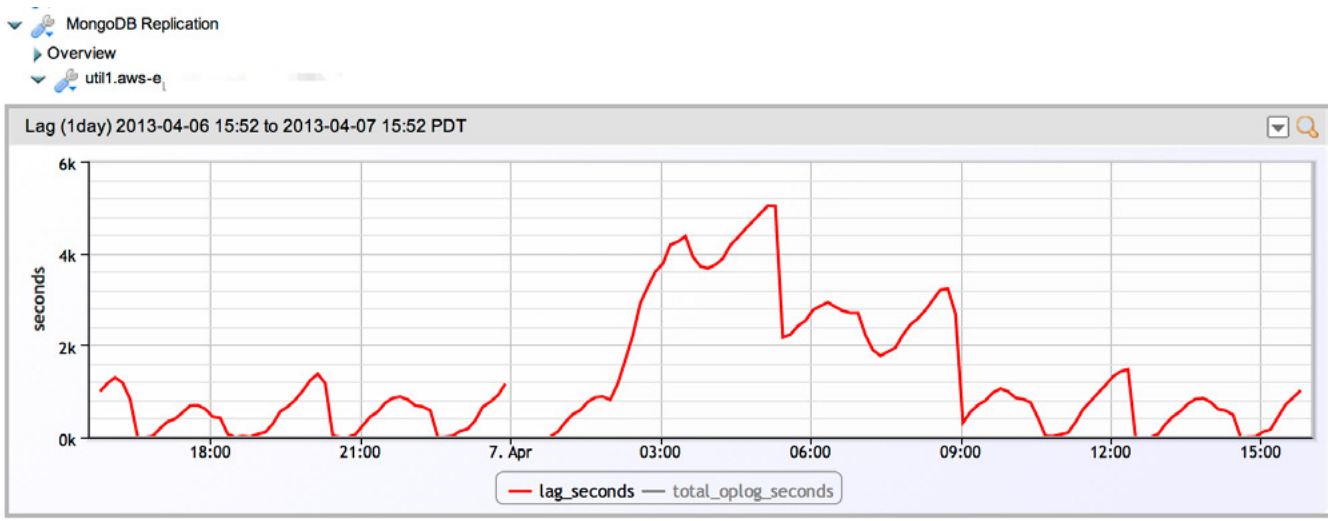


Figure 3. *MongoDB Replication*

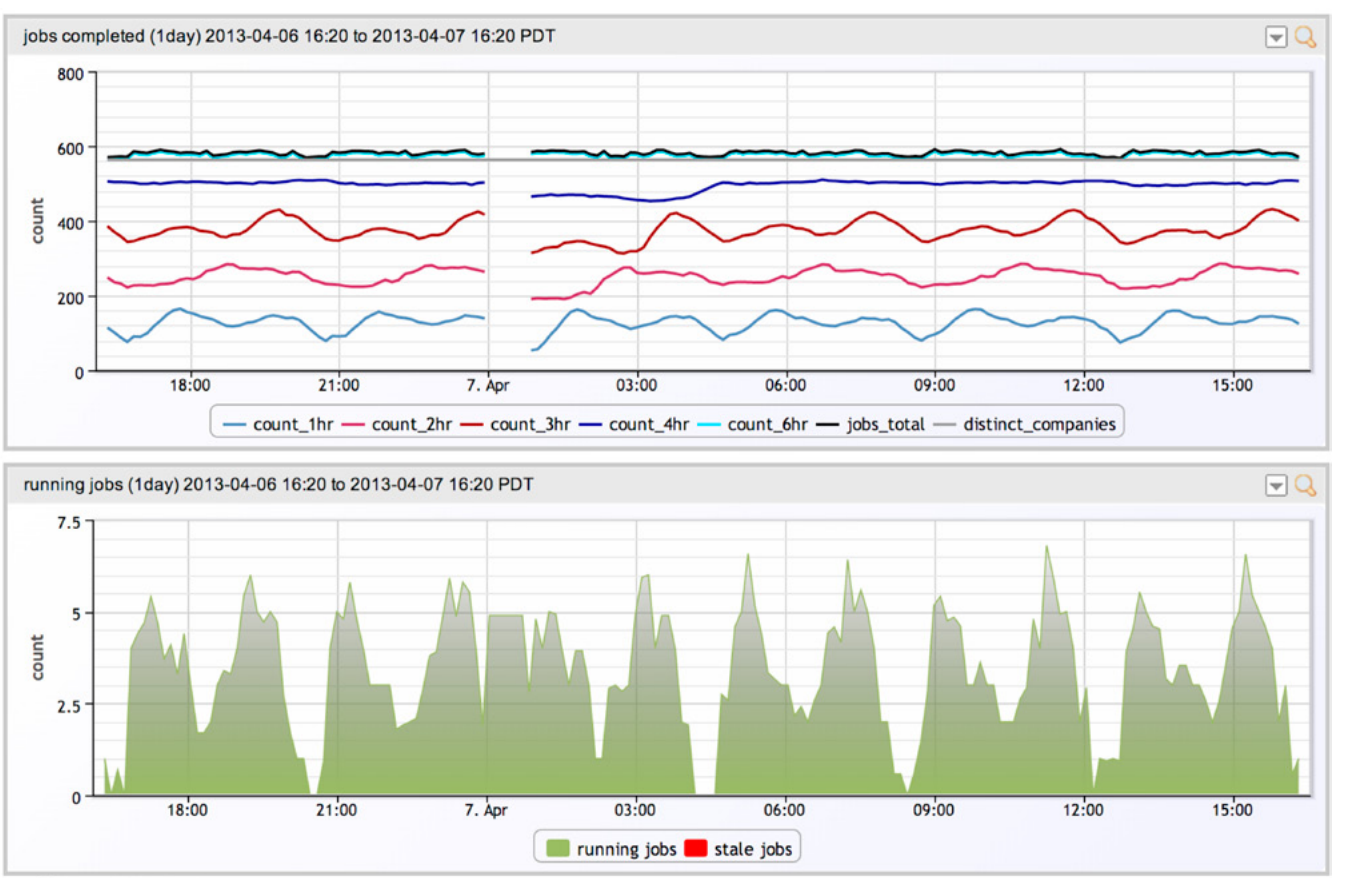


Figure 4. *Data*

easy for each server to act independently, but with full knowledge of the system. It's also very easy for us to instrument the backup system, so we can be confident relying on it without having to check that it is in fact working. We can obviously monitor the standard MongoDB metrics, including replication lag between replicas, queries, updates, disk flushing, etc – but we can also very easily monitor backup specific data. As all the data is in a standard MongoDB form, we can just as easily graph the output of specific MongoDB queries, showing things like the number of jobs in process; the number of jobs that backed up customers by the time of their last backup, and so forth (Figure 4).

We're alerted if customers are not backed-up as they should be, without having to "watch" the backup system. And the GridFS system exposes all the data it has for easy integration into other systems, such as data warehouse systems, as needed.

Adding additional datacenters into the backup system is easy – they can be members of the existing replica sets, or added as their own replica sets. In any case, no code changes are required for the backup system – add them as members to the MongoDB replica set, and MongoDB takes care of the synchronization.

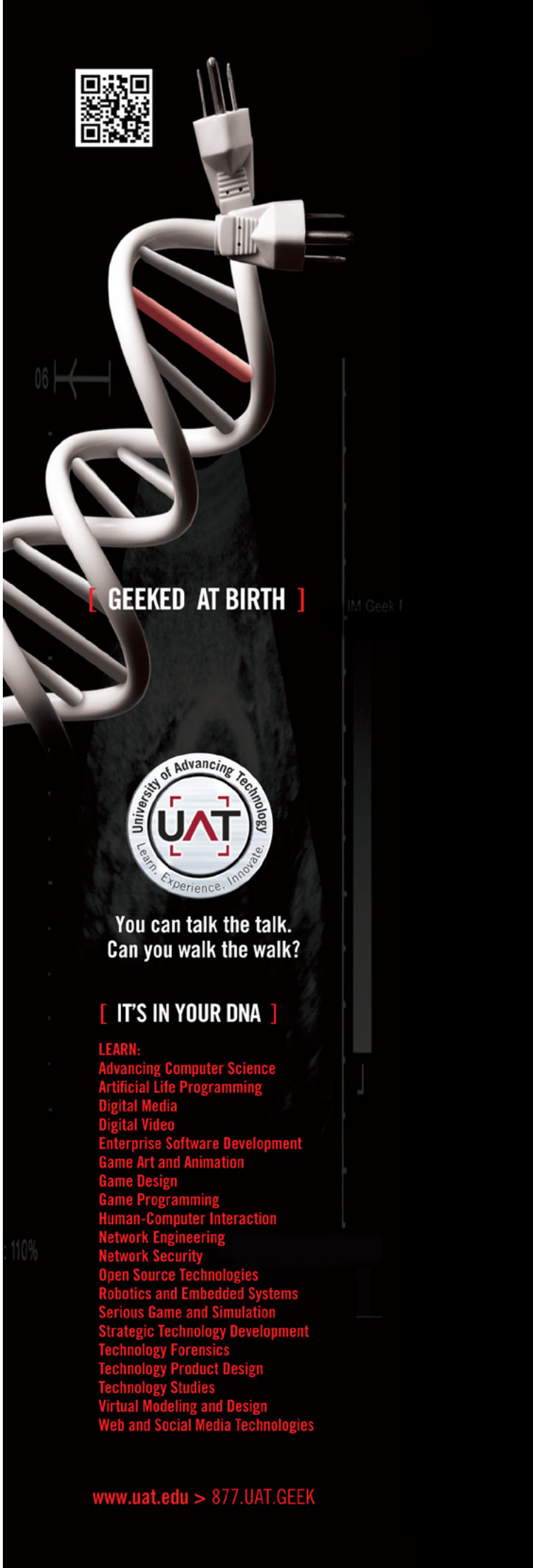
Recovering individual customers or entire nodes or datacenters are quick and easy operations. We've had to recover using this system several times – once because of a hardware failure, but also from customers requesting to be restored back to a point in time before they took some action they later regretted. The system has worked flawlessly.

Of all the backup systems I have worked with in 20 years in IT and systems administration, I've never seen an easier way of distributing data to multiple datacenters in near real-time, with great transparency of operations, than using GridFS on top of MongoDB.

STEVE FRANCIS

(@stevefrancisLM), Chief Product Officer, LogicMonitor Inc

Steve Francis founded LogicMonitor after running datacenters for SaaS and other companies for 20 years. A background in network and systems administration led him to believe that monitoring should not be the focus on sysadmins or ops staff – it should just work. Ironically, he started a company that focuses entirely on monitoring, so that others may be freed from it.



GridFS – Store and stream files with MongoDB

GridFS is a protocol for storing big files in MongoDB along with their metadata. Thanks to its Replica Set and Sharding features, MongoDB can be used as a fault-tolerant, high-availability distributed filesystem. With appropriate drivers like ReactiveMongo, one can also stream files from and into MongoDB.

MongoDB is a document-oriented database. The MongoDB documentation defines documents as "the default representation of most user accessible data structures in the database" (*http://docs.mongodb.org/manual/core/document/*). They are serialized using the BSON protocol (Binary JSON, *http://bsonspec.org*) and stored in groupings that are called "collections". BSON defines a binary data type; moreover, it has been designed to be very efficient for traversing. It is easy to skip or extract the data from a document. This makes MongoDB a good option to store binary blobs.

So, why not use MongoDB to store the files of our web application? Well, the problem is that BSON documents are limited to 16MB – if our application manages big files (like archives or movies), it is a showstopper.

That's where GridFS comes into play. GridFS is a protocol for storing files in a MongoDB database. The first implementation was written in early 2009; back then MongoDB documents could not be bigger than 4MB and there was no simple, recommended way to save bigger files.

How does it work?

GridFS divides a file content in many chunks of equal size, and stores them in documents in one collection called *chunks*. A chunk document holds three fields: *data* (the data of the chunk), *n* (the index of the chunk), and `files_id` (the id of the file itself). The other pieces of information about the file – like its name, its length, its type – are written in one document in a separate collection named *files*. We often refer to this document as "the file metadata".

To illustrate the way GridFS works, let's upload a file and inspect the documents that were created. We can achieve this using *mongofiles*, a small utility that ships with the default MongoDB distribution: Listing 1.

Now, let's connect to the database with the *mongo* console and inspect the documents *mongofiles* inserted (Listing 2).

Two collections were created: `fs.chunks` and `fs.files`. The *files* and *chunks* collection names are prefixed by the GridFS store name (by default *fs*), followed by a dot character (Listing 3).

As we can see, the metadata document contains interesting information, like the MD5 hash of the file, its length in bytes, and the size of the chunks. The default size of a chunk is 256kB – we will discuss this value later (Listing 4).

Apart from the data itself, a chunk document contains the id of the file to which it belongs, and its number.

Listing 1. *Upload a file with mongofiles*

```
$ ./bin/mongofiles -d gfstest -l ~/Downloads/mongodb-osx-x86_64-2.4.0-rc3.tgz put mongodb-osx-x86_64-2.4.0-rc3.
                    tgz
connected to: 127.0.0.1
added file: { _id: ObjectId('51425ec3a77d934fb2094ef7'), filename: "mongodb-osx-x86_64-2.4.0-rc3.tgz", chunkSize:
                    262144, uploadDate: new Date(1363304132007), md5: "e804fd138e2a43591d1f0f1061922025",
                    length: 101332608 }
```

Reading a file from GridFS seems obvious: we just have to get each chunk one by one in the natural order. For speeding up queries on the chunks collection, a compound index is created on `files_id` and `n`: Listing 5.

GridFS, a fault-tolerant, high availability distributed filesystem

So, what are the advantages using GridFS instead of a regular file system? Well, one of the killer features of MongoDB is its ability to be replicated and sharded easily. Thanks to GridFS, you get a distributed filesystem with redundancy and high availability with little effort. And you can use the power of the query engine. Moreover, if your application already uses MongoDB, you don't have to set up and maintain another component in your infrastructure.

Set up GridFS with sharding

Thanks to its sharding ability, MongoDB can manage a virtually unlimited amount of data – if your servers disks are full, you may simply add another shard.

Even if a GridFS stores millions of files, there is no need to shard the *files* collection since it only contains metadata – this collection will not take much space. On the contrary, the *chunks* collection may grow really fast and so is a good candidate to be sharded.

A very common issue with sharding is how to choose the shard key. The shard key must be chosen wisely to ensure that the data is evenly distributed across the shards. Failing to do so can lead to performance problems and makes sharding useless. Given the structure of the documents in the *chunks* collection, it may be tempting to choose `files_id` as the shard key so all the chunks will be on the same shard. There is a huge caveat though: the `files_id` field is often an *ObjectId*. *ObjectId* is a 12-bytes integer type that embeds a timestamp component and so it monotonically increases in time. Because of the way sharding works, all the new inserts will end up in one shard, creating a so-called "hotspot". The problem is well known and documented (*http://docs.mongodb.org/manual/core/sharded-cluster-internals/#write-scaling*). There are some options one may consider for sharding GridFS stores:

- with MongoDB 2.4+, one can use a hashed index as the shard key. The idea is to make a hashed index on the `files _ id` field, and use that index as the shard key. This is the best solution: all the chunks of a file are on the same shard, but chunks of different files tend to be evenly distributed across all the shards.
- use an evenly distributed value for `files _ id` (and so `_ id` in the *files* collection) instead of *ObjectIds* (like a random value or a MD5 hash). This solution is basically the same as the first one, except that one do not rely on MongoDB to generate the values for the shard key (*http://stackoverflow.com/questions/5344606/sharding-gridfs-on-mongodb/5346291#5346291*).
- use a compound shard key with the `files _ id` and n fields. If MongoDB 2.4+ cannot be used, or the type of `files _ id` (to avoid ObjectId) cannot be changed, this is a pretty good general solution: the chunks of a file tend to end up on the same shard, and the data is evenly distributed.

Listing 2. *Show the collections of a GridFS store*

```
> use gfstest
switched to db gfstest
> show collections
fs.chunks
fs.files
system.indexes
```

Listing 3. *The content of a file metadata document*

```
> db.fs.files.find().pretty()
{
    "_id" : ObjectId("51425ec3a77d934fb2094ef7"),
    "filename" : "mongodb-osx-x86_64-2.4.0-rc3.tgz",
    "chunkSize" : 262144,
    "uploadDate" : ISODate("2013-03-
                    14T23:35:32.007Z"),
    "md5" : "e804fd138e2a43591d1f0f1061922025",
    "length" : 101332608
}
```

Listing 4. *The content of a chunk (except the data field)*

```
> db.fs.chunks.find({}, {data: 0}).limit(1).pretty()
{
    "_id" : ObjectId("51425ec3ffd3b31f1be6bd5e"),
    "files_id" : ObjectId("51425ec3a77d934fb2094ef7"),
    "n" : 0
}
```

GridFS in practice

Inserts, updates and deletions

As soon as write operations are involved, one must take extra care of the errors. A common mistake is to write the metadata in the *files* collection before all the chunks are actually written. If an error happens, the content may be partial, yet the file may be downloaded because the metadata are available. The metadata should be saved at last, when everything was successful.

For the same reason, removing a file should be done in two steps: first, delete the metadata and then the chunks. It is easy to update the metadata of a file: this can be done with the regular `update()` function on the

Listing 5. *Show the collections of a GridFS store*

```
> db.system.indexes.find()
{ "v" : 1, "key" : { "_id" : 1 }, "ns" : "gfstest.fs.files", "name" : "_id_" }
{ "v" : 1, "key" : { "filename" : 1 }, "ns" : "gfstest.fs.files", "name" : "filename_1" }
{ "v" : 1, "key" : { "_id" : 1 }, "ns" : "gfstest.fs.chunks", "name" : "_id_" }
{ "v" : 1, "key" : { "files_id" : 1, "n" : 1 }, "ns" : "gfstest.fs.chunks", "name" : "files_id_1_n_1" }
```

Listing 6. *Writing a file into a GridFS store*

```
// metadata for the file we are going to put in the store
val metadata = DefaultFileToSave(
  "mongodb-osx-x86_64-2.4.0-rc3.tgz",
  contentType = Some("application/x-compressed"))

// enumerator of the file content
val enumerator = Enumerator.fromFile(new File("/Users/sgo/Downloads/mongodb-osx-x86_64-2.4.0-rc3.tgz"))

// ok, save the file and attach a callback when the operation is done
gridFS.save(enumerator, metadata).onComplete {
  case Success(file) =>
    println(s"successfully saved file of id ${file.id}")
  case Failure(e) =>
    println("exception while saving the file!")
    e.printStackTrace()
}
```

Listing 7. *Read a file from a GridFS store*

```
val query = BSONDocument("filename" -> "mongodb-osx-x86_64-2.4.0-rc3.tgz")
val maybeFile = gfs.find(query).headOption
maybeFile.onComplete {
  // we found a file
  case Success(Some(file)) =>
    println(s"fetching content of file '${file.filename}' (length ${file.length}b)...")
    val out = new FileOutputStream(file.filename)
    val writeOnFileSystemTask = gfs.readToOutputStream(file, out)
    writeOnFileSystemTask.onComplete {
      // we successfully wrote the file content on disk
      case Success(_) =>
        println("done...")
      // there was an error, let's print it out
      case Failure(e) =>
        println("file download failed")
        e.printStackTrace()
    }
  case Success(None) =>
    println("file not found")
  case Failure(error) =>
    println("got an error!")
    error.printStackTrace()
}
```

files collection. On the other hand, content updates should be done carefully. Indeed, concurrent updates should be taken in consideration: what happens if two applications (or two threads of the same application) are updating different chunks of the same file? This obviously leads to consistency problems. A good way is to remove the old version first, and then write the new one. This is generally well handled by MongoDB drivers. It is safe to perform such operations with them, but if for some reason you have to deal with the *files* and *chunks* collections directly, you should keep this in mind.

Chunk size

The GridFS specification limits chunk size to 256kB. The chunk size has no effect on space usage: a chunk takes up only what it needs. Although it is generally a good default value, it may be interesting to change it depending on the application that consumes the data.

Some implementations (like the official Java driver) fetch data chunk by chunk and do not use cursors. This means that the more chunks there are, the more roundtrips are needed. On the other side, if chunks are consumed on the stream, the chunk processing requires less memory – which is better when the application must serve a lot of files simultaneously.

Streaming with GridFS

The following examples are written in Scala using ReactiveMongo, a non-blocking driver for MongoDB. One of the main focuses of ReactiveMongo is to enable streaming data from and into MongoDB in a non-blocking way, thanks to the Iteratee/Enumerator pattern.

Listing 8. *Enable streaming data*

```
// Save the uploaded file as attachment of the article with the given id.
// We use the gridFSBodyParser method, that handles the file upload for
// us by saving the chunks directly in GridFS.  Then we save the uploaded
// file as attachment of the article with the given id
def saveAttachment(id: String) = Action(gridFSBodyParser(gridFS)) { request =>
  // here is the future file!
  val futureFile = request.body.files.head.ref
  // when the upload is complete, we add the article id to the file entry
  // (in order to find the attachments of the article)
  val futureUpdate = for {
    file <- futureFile
    // here, the file is completely uploaded, so time to update article
    updateResult <- {
      gridFS.files.update(
        BSONDocument("_id" -> file.id),
         BSONDocument("$set" -> BSONDocument("article" -> BSONObjectID(id))))
    }
  } yield updateResult
  // when update is done, send the response to the browser
  futureUpdate.map {
    // the update is successful, redirect to the article form
    case _ => Redirect(routes.Articles.showEditForm(id))
  }.recover {
    // there was an error, send back a 500 status code
    case e => InternalServerError(e.getMessage())
  }
}
// Stream the attachment to the browser
def getAttachment(id: String) = Action {
  Async {
    // find the matching attachment, if any, and stream it to the client
    val file = gridFS.find(BSONDocument("_id" -> new BSONObjectID(id)))
    serve(gridFS, file)
  }
}
```

A quick introduction to the Iteratee pattern

Iteratees are an immutable implementation of a Consumer/Producer pattern, in which producers generate data that may be processed by consumers at different speeds (*http://en.wikipedia.org/wiki/Producer-consumer_problem*). Iteratees are consumers and are "fed" data to do something with it. Enumerators are data producers that "feed" data to an iteratee. And a third component, Enumeratees, can be thought of as pipes or interfaces that connect Enumerators to Iteratees providing for transform capabilities adapting data from an Enumerator to conform to the needs of the Iteratee. This pattern has major assets:

- it is immutable, and therefore suitable for heavily concurrent processing
- it composes very well (it is very easy to define elementary Iteratees and Enumeratees and combine them for more complex usages)
- by design, it is perfect for stream processing: data is produced and consumed on the flow, without filling the memory unnecessarily

Using Iteratees and Enumerators to access a GridFS store

Given these properties, Iteratees are perfect for streaming files from MongoDB. Since ReactiveMongo is based on this pattern, it provides a `GridFS` class that deals with Iteratees and Enumerators. Among others, there are three useful methods on a `GridFS` instance:

- `find(selector: BSONDocument): Cursor[DefaultReadFile]` which returns the metadata documents in the files collection that mach the given selector
- `save(enumerator: Enumerator[Array[Byte]], metadata: DefaultFileToSave): Future[DefaultReadFile]` which saves the data prduced by the enumerator in the store
- `enumerate(file: DefaultReadFile): Enumerator[Array[Byte]]` which enumerates the content of the given file

For example, here is how we can use `save()` to store a file: Listing 6. Getting the content of a file follows the same logic: Listing 7.

Streaming files in a web application

Depending on the kind of control you want to have on your application files, you have several possibilities:

- if your application basically serves only static files, without any access policy, and does never add or update files in the store, the best option is to use a plugin for your reverse proxy (if available). There are plugins for Apache (*https://bitbucket.org/onyxmaster/mod_gridfs*) and Nginx (*https://github.com/mdirolf/nginx-gridfs*)
- if you make a more advanced usage of GridFS (like searching for files, saving new ones, updating metadata, …), you may consider relying on a driver and implementing those features yourself

If you go for the second option, then ReactiveMongo in combination with a modern web framework like Play2 may be a very good fit. Play2 has a built-in support for Iteratees and Enumerators. For example, one can give an `Enumerator[Array[Byte]]` as a result – Play will automatically stream the data produced by this Enumerator to the browser. Of course it works both ways, since you can supply an Iteratee that consumes the data sent by the browser. Basically, all you have to do to stream data from and to a GridFS store is to give Play the Enumerators and Iteratees returned by ReactiveMongo. To make things even smoother, there is a Play plugin that provides ready-to-use methods. Let's use it to write a simple application that stores attachments of articles: Listing 8.

On the Web

- *http://docs.mongodb.org/manual/applications/gridfs/* – GridFS specification
- *http://docs.mongodb.org/manual/sharding/* – Sharding documentation
- *http://docs.mongodb.org/manual/release-notes/2.4/#new-hashed-index-and-sharding-with-a-hashed-shard-key* – New Hashed Index support in MongoDB 2.4+
- *http://reactivemongo.org/* – ReactiveMongo, a non-blocking, reactive Scala driver for MongoDB
- *http://www.playframework.com* – Play2 Framework
- *http://mandubian.com/2012/08/27/understanding-play2-iteratees-for-normal-humans/* – A good article on Iteratees

Glossary

- Sharding: ability to partition data in multiple nodes
- Non-blocking IO: input/output operations that don't block the thread waiting for a reply
- Iteratee: a consumer of data, in the Iteratee/Enumerator
- Enumerator: a producer of chunks of data, in the Iteratee/Enumerator pattern
- Enumeratee: a transformer of chunks of data

Summary

GridFS provides a simple and elegant way to turn MongoDB into a distributed and horizontally scalable filesystem. Used along with ReactiveMongo, GridFS enables the ability to stream data in a non-blocking way, and makes MongoDB a database of choice for modern web applications.

STEPHANE GODBILLON

The author works has been working as a software architect at Zenexity for five years, designing and developing web applications. He is the creator of ReactiveMongo, a reactive Scala driver for MongoDB, and co-organizes the MongoDB User Group of Paris.

Real Time Big Data
for Business Users

Use Hadoop without complexities, in real time

Bring down the total cost of your Hadoop projects. Udichi provides an elegant web based interface to build your data solution in days.

Extend your existing Data Warehouse

Your existing data warehouse has its own value. Extend your current infrastructure with the power of Big Data but without spending big money.

It's easy and agile. No need for any costly and time consuming setup. Try different solutions, many times, from within the comfort of your browser.

UDiCHi
Unleash Power Of Data In Real Time

Contact us: **info@udichi.com**
www.udichi.com

Combining PHP with Java in the Same Application Using PHP/Java Bridge

As the two languages got stronger, they became highly appreciated by dedicated developers, who began to ask: What would happen if these two languages met? Could we combine their powers constructively?

What you will learn...

- Install and configure the PHP/Java Bridge,
- Use Java classes in PHP scripts,
- Use PHP scripts in Java classes.

What you should know...

- knowledge of Java SE (bean) and PHP5 core and how they interact using the PHP / Java Bridge.

Trying to answer these questions gave birth to the PHP/Java Bridge, which creates a communication channel between these two entities. Using the bridge, you can develop classes in Java and call their methods from PHP or you can use PHP scripts in your Java desktop/web applications.

How it works

As you probably know, PHP 4 supported an extension for combining PHP with Java, but from PHP 5 or PHP 6, to combine PHP with Java you should install PHP/Java Bridge (you can download it from *http://php-java-bridge.sourceforge.net/pjb/download.php*), whereof you can learn more from *http://php-java-bridge.sourceforge.net/pjb/.*

Using PHP/Java Bridge

The current distribution of the PHP/Java Bridge is available in *.zip* at *http://sourceforge.net/projects/php-java-bridge/* . The installation process depends on which Java platform will be interacting with PHP through this bridge.

For J2SE, installation is simple:

- Install J2SE 1.6 or above.
- Install PHP 5.1.4 or above.
- Extract the *php-java-bridge_5.2.2_j2ee.zip*.
- From the command prompt, navigate to this folder and type:
  `...>java -classpath` *JavaBridge.war* TestInstallation

In the current folder, you should see an ext folder that contains four *.jar* files. Copy *JavaBridge.jar* and *php-script.jar* to your J2SE ext directory `{JAVA_HOME}/jre/lib/ext`. Here's how to install the PHP/Java Bridge for J2EE:

- Copy *JavaBridge.war* archive into the auto-deploy folder of your J2EE server or servlet engine (Tomcat, Resin, etc.)
- Rename this *.war* archive whatever you're calling the new application and restart the J2EE server. Wait for the auto-deploying process to create the directory associated to this *.war*. In this example, the app is called *appName.war*.
- Test the new application from the browser like this: *http://localhost:8080/appName* (click on *test.php*).
- If you have your J2EE server running on a different host or listen on a different port, then modify these parameters accordingly.

To begin with, you must know that the PHP/Java Bridge comes with a set of functions (PHP classes)

Listing 1a. *tenisBean.jar*

```
package com.tenis;

import java.io.Serializable;

public class TenisBean implements Serializable {

    final String[] allPlayers = new String[]{"rafael.png", "nicolas.png", "roger.png", "novak.png", "andy.png",
                    "juan.png"};
    final String[] names = new String[]{"Rafael Nadal", "Nicolas Almagro", "Roger Federer", "Novak Djokovic",
                    "Andy Murray", "Juan Martin Del Potro"};
    final String[] countries = new String[]{"Spain", "Spain", "Switzerland", "Serbia", "United Kingdom",
                    "Argentina"};
    final String[] birthdates = new String[]{"3 June 1986", "21 August 1985", "8 August 1981", "22 May 1987", "15
                    May 1987", "23 September 1988"};
    final String[] birthplaces = new String[]{"Manacor, Mallorca, Spain", "Murcia, Spain", "Basel, Switzerland",
                    "Belgrade, Serbia", "Dunblane, Scotland", "Tandil, Argentina"};
    final String[] residences = new String[]{"Manacor, Mallorca, Spain", "Murcia, Spain", "Bottmingen,
                    Switzerland", "Monte Carlo, Monaco", "London, England", "Tandil, Argentina"};
    final String[] heights = new String[]{"6 ft. 1 in. ( 1.85 metres )", "6 ft. 0 in. ( 1.83 metres )", "6 ft. 1
                    in. ( 1.85 metres )", "6 ft. 2 in. ( 1.88 metres )", "6 ft. 3 in. ( 1.91 metres )", "6 ft. 6
                    in. ( 1.98 metres )"};
    final String[] weights = new String[]{"188 lbs. ( 85.5 kilos )", "179 lbs. ( 81.4 kilos )", "187 lbs. ( 85
                    kilos )", "176 lbs. ( 80 kilos )", "185 lbs. ( 84.1 kilos )", "182 lbs. ( 82.7 kilos )"};
    final String[] plays = new String[]{"Left", "Right", "Right", "Right", "Right", "Right"};
    final String[] pros = new String[]{"2001", "2003", "1998", "2003", "2005", "2005"};
    private String[] players = new String[]{"", "", ""};
    private String currentplayer = "";
    private String name = "";
    private String country = "";
    private String birthdate = "";
    private String birthplace = "";
    private String residence = "";
    private String height = "";
    private String weight = "";
    private String play = "";
    private String pro = "";

    public TenisBean() {
    }

    public String[] getPlayers() {

        if (players[0].trim().length() == 0) {
            int player = new java.util.Random().nextInt(4);
            for (int i = 0; i < 3; i++) {
                players[i] = allPlayers[player];
                player++;
            }
        }

        return players;
    }
```

Listing 1b. *tenisBean.jar*

```
    public void setPlayers(String[] players) {
        this.players = players;
    }

    public String getCurrentplayer() {
        return currentplayer;
    }

    public void setCurrentplayer(String currentplayer)
                    {
        this.currentplayer = currentplayer;

        if (currentplayer.trim().length() > 0) {
            for (int i = 0; i < allPlayers.length; i++)
                    {
                if (allPlayers[i].equals(currentplayer.
                    trim())) {
                    this.name = names[i];
                    this.country = countries[i];
                    this.birthdate = birthdates[i];
                    this.birthplace = birthplaces[i];
                    this.residence = residences[i];
                    this.height = heights[i];
                    this.weight = weights[i];
                    this.play = plays[i];
                    this.pro = pros[i];
                    break;
                }
            }
        }
    }

    public String getBirthdate() {
        return birthdate;
    }

    public void setBirthdate(String birthdate) {
        this.birthdate = birthdate;
    }

    public String getBirthplace() {
        return birthplace;
    }

    public void setBirthplace(String birthplace) {
        this.birthplace = birthplace;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getHeight() {
        return height;
    }

    public void setHeight(String height) {
        this.height = height;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPlay() {
        return play;
    }

    public void setPlay(String play) {
        this.play = play;
    }

    public String getPro() {
        return pro;
    }

    public void setPro(String pro) {
        this.pro = pro;
    }

    public String getResidence() {
        return residence;
    }

    public void setResidence(String residence) {
        this.residence = residence;
    }

    public String getWeight() {
        return weight;
    }

    public void setWeight(String weight) {
        this.weight = weight;
    }
}
```

that were especially created to integrate Java code into PHP scripts. Some of these functions are:

- `java`: This allows you to access the Java type with the given name. For example:
  `java("java.lang.System")->getProperties();`

- `java_autoload`: This allows you to load a set of Java libraries in the current PHP script. For example:
  `java_autoload("my_1.jar;my_2.jar");`

- `java_cast`: This allows you to convert a Java object into a PHP value. For example:
  `$mystr=new java("java.lang.String","9");$phpnr=java_cast($mystr,"integer");echo $phpnr;`

- `java_is_null`: This allows you to check if a value is null or not. For example:
  `java_is_null(java("java.lang.System")->;getProperty("my_prop"))`

- `java_session`: This allows you to return a session handle. For example:
  `$session = java_session();`

- `java_values`: This allows you to evaluate the object and fetch its content (only if this is possible). For example:

  ```
  $result = java_values($calcinstance>addAB($term_1,$term_2));
  ```

To make use of these functions, your PHP applications must contain the corresponding PHP classes. The most important class is *Java.inc*, but the complete list of classes can be seen in the `appName/java` directory.

PHP/Java application

Now, that you know the PHP/Java Bridge basics, it is time to develop your first PHP/Java application. The application listed next, is a PHP script that implements a Java session bean (*A session bean is the enterprise bean that directly interact with the user and contains the business logic of the enterprise application. A session bean represents a single client accessing the enterprise application deployed on the server by invoking its method.*) that will be the *engine* of the application, because the results will be provided by a Java class named *tenisBean*. In other words, Java behaves as a pure business logic component and PHP behaves as an interrogator component.

Listing 2. *atpPlayers.php*

```
<?php require_once("java/Java.inc");

$session = java_session();

if(java_is_null($t=$session->get("bean"))) {
  $session->put("bean", $t=new Java("com.tenis.
                    TenisBean"));
}

if(isset($_GET['cp'])){
  $t->setCurrentplayer($_GET['cp']);
}
?>

<html>
 <head>
  <title>ATP Best Players Ever</title>
  <style>
   .atpStyle1 {
     font-family: Helvetica,Arial,sans-serif;
     color: #000000;
     font-size: 11;
     font-weight: bold;
   }
   .atpStyle2 {
     font-family: Helvetica,Arial,sans-serif;
     color: #000000;
     font-size: 11;
     font-weight: bold;
   }
   .atpStyle3 {
     font-family: Helvetica,Arial,sans-serif;
     color: #cc0000;
     font-size: 15;
     font-weight: bold;
   }
   .atpStyle4 {
     font-family: Helvetica,Arial,sans-serif;
     color: #00000;
     font-size: 13;
     font-weight: bold;
   }
   .atpStyle5 {
     font-family: Helvetica,Arial,sans-serif;
     color: #00000;
     font-size: 24;
     font-weight: bold;
   }

  </style>
 </head>
 <body>
```

```
<?php $players = java_values($t->getPlayers()); ?>
<table align="center">
 <tr>
  <td colspan="3" align="center" class='atpStyle5'>THREE OF THE BEST EVER ATP PLAYERS</td>
 </tr>
 <tr>
  <?php
     foreach ($players as &$value) {
       echo("<td><img src='images/".$value."' alt=''></td>");
     }
  ?>
 </tr>
 <tr>
  <?php
     foreach ($players as &$value) {
       echo("<form name='atpForm_".$value."' action='".$PHP_SELF."'
             method='get'>");
       echo("<input type='hidden' name='cp' value='".$value."'>");
       echo("<td><input type='submit' value='Details'></td>");
       echo("</form>");
     }
  ?>
 </tr>
 <?php
    if(strlen(java_values($t->getCurrentplayer())) > 0){
    echo("<tr><td><hr></td><td><hr></td><td><hr></td></tr>");
    echo("<tr><td><img src='images/".java_values($t->
    getCurrentplayer())."' alt='' width='45'
    height='60'></td><td class='atpStyle1'>Name: </td>
    <td class='atpStyle3'>".java_values($t->getName())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Country: </td>
    <td class='atpStyle4'>".java_values($t->getCountry())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Birth Date: </td>
    <td class='atpStyle4'>".java_values($t->getBirthdate())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Birth Place: </td>
    <td class='atpStyle4'>".java_values($t->getBirthplace())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Residence: </td>
    <td class='atpStyle4'>".java_values($t->getResidence())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Weight: </td>
    <td class='atpStyle4'>".java_values($t->getWeight())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Plays: </td>
    <td class='atpStyle4'>".java_values($t->getPlay())."</td></tr>");
    echo("<tr><td></td><td class='atpStyle2'>Turned Pro: </td>
    <td class='atpStyle4'>".java_values($t->getPro())."</td></tr>");
    echo("<tr><td><hr></td><td><hr></td><td><hr></td></tr>");
    }
 ?>
 <tr>
  <td colspan="3" align="center" class='atpStyle5'>THREE OF THE BEST EVER ATP PLAYERS</td>
 </tr>
</table>
</body>
</html>
```

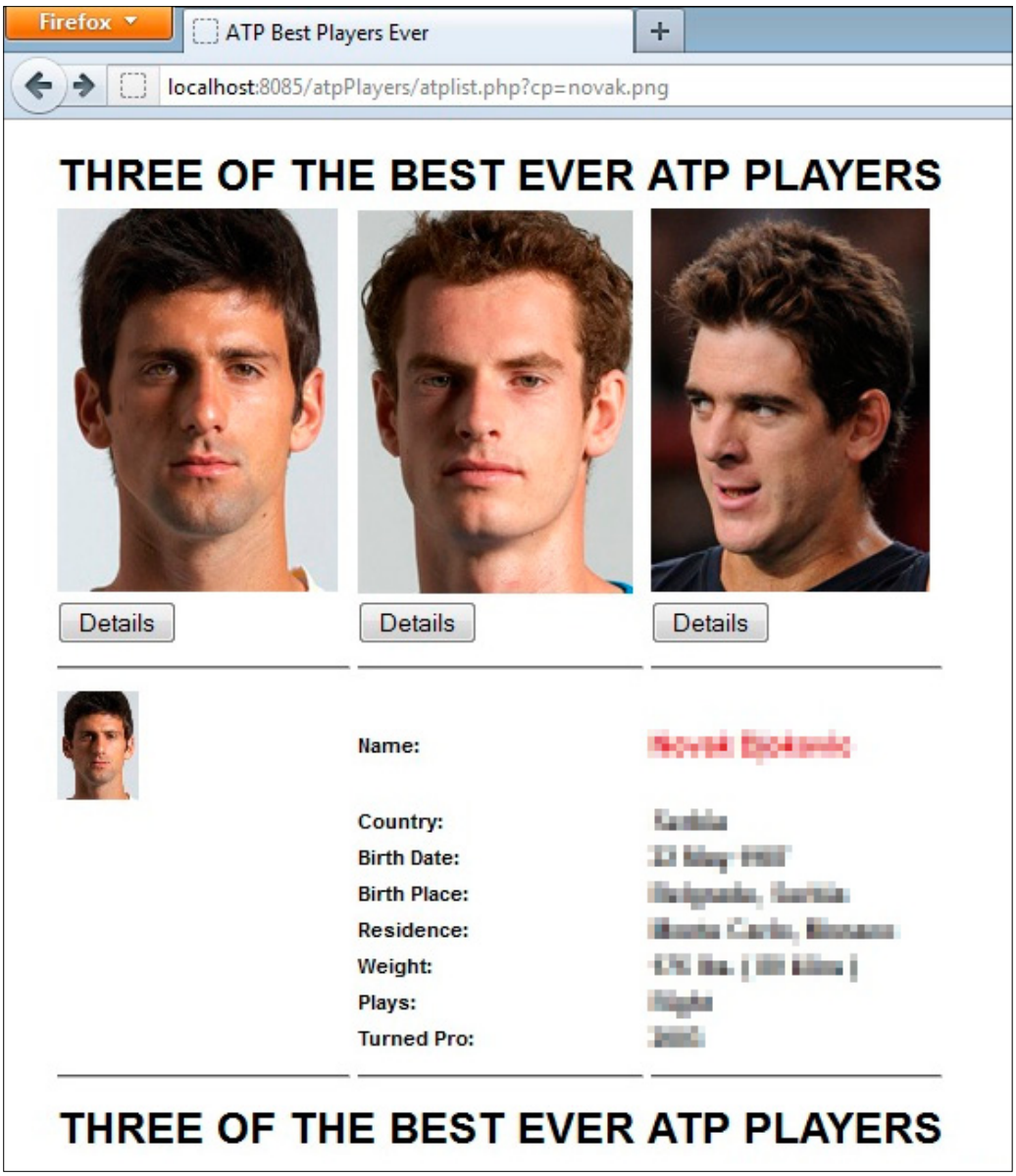


Figure 1. *Three of the best ever ATP players*

Practically, this application randomly displays three tennis players, from a list containing six players, as you can see in the *tenisBean* class, and their corresponding personal information. This will be accomplished by the PHP script, which interact with the *tenisBean.jar* through its set and get specific methods. The result is listed in the end part of this article.

After you compile this Java source, place it in a *.jar* archive named *tenisBean.jar*. Copy this file to the `atpPlayers/WEB-INF/lib` directory.

Now it's time to develop the PHP script that will call the above Java class. Using the methods described in the *Using PHP/Java bridge* section, you can write the *atpPlayers.php*, shown on Listing 2.

The output of mixing PHP with Java in the same application, can be seen writing in browser the *localhost:8085/atpPlayers/atplist.php* link.

Summary

This introduction to the PHP/Java Bridge hopefully taught you the basic mechanisms that allow the interaction between these two powerful programming languages. Developing these kinds of applications may be a delicate task, but if you think that you can get serious advantages from this symbiosis, then don't hesitate to use it!

OCTAVIA ANDREA ANGHEL

Octavia Andrea Anghel is a senior PHP developer currently working as a primary trainer for programming teams that participate at national and international software-development contests. She consults on developing educational projects at national and international level. She was coordinating in European Job and Career, International Comenius project along with Spain, Italy, Turkey and Czech Republic countries. She is a co-author of the book XML technologies—XML in Java and at the moment is working at an online course to teach PHP for Udemy website.

Localization Website with PHP and GETTEXT (L10N)

Nowadays, being such a globalized world that we live in, our website needs to be available in multiple languages to be accessible to the largest audience possible. Access new markets, make our business world available, are some of the benefits of the internationalization (i18n) and localization (l10n) of our website.

What you will learn...

- Planning the localization of our website
- Enabling dynamic extensions using php.ini file
- Using gettext extension to localize strings
- Using software to manPoEdit sage translations files

What you should know...

- PHP basic syntax

Well that's great! Let's do it right now!

Steady on, we should think about it a little bit. Working on a small website the content translation might sound pretty simple, we just have to copy our entire site as many times as the number of languages we have to work on, and then we make the changes wherever is necessary. But pay attention, because when the site becomes bigger, the translation is going to require more and more work, this work takes time, and as everyone knows, time means money.

Is there an easier way?

Of course! There is a widely used framework for localization (L10N): gettext that can be used with various programming languages, including PHP.

Working with poEdit is really simple, simply select the text you want to translate in the upper half of the window, and translate it in the other half.

At first sight seems to be complicated, but don't be afraid, finally you will see how easy is to get it to work. Let's get started.

First of all we need to enable the gettext extension, look in your *php.ini* the line

```
;extension=php_gettext.dll
```

and uncomment it, removing the leading semicolon (;). Restart your Apache or IIS server in order to reload the changes.

How it works

Using gettext to get translated strings couldn't be easier, simply call gettext("some text") and you get a localized version of "some text to get translated" if it's available or "some text" if it's not. Even simpler -lazy mode on- you can use _("some text") and you'll obtain the same result.

Listing 1. *How to use gettext*

```
<?php
echo _("some text");
?>
```

When this code executes you'll see "some text" in your screen.

The translation begins

Now we have created our initial script, we are going to create localized versions for different languages.

First you need to download a tool named poEdit, it can be obtained from *http://www.poedit.net/*. This tool will allow to easily creating the translation files that we need, you don't have to be a programmer to use it, so you can delegate the translation task to someone that have expertise on that field.

So download, install and launch poEdit, then create a new catalog (*File>New catalog*) and choose the language you want to translate the website to. In this example I'm using Spanish and UTF-8 as charset encoding (Figure 1).

Now in the paths tab we need to change "Base path" value to the directory of our php script, and add the "." as a path (Figure 2).

Before saving the catalog we need create some directory structure in our script location.

Create a folder "locale",

Inside create a folder named like "iso code language"_"iso code country", in this example es_AR (Spanish, Argentina),

Create a folder named "LC_MESSAGES" (Figure 3).

Now we are ready, press the OK button and save the file as messages.po inside the LC_MESSAGES directory.

After save the file, *poEdit* will scan all the paths that you have configured in the paths tab and extracts all the string that he founds in `gettext()` or `_()` `functions` (Figure 4).

Isn't that great? We are now ready to start the translation of our website!

Working with poEdit is really simple, simply select the text you want to translate in the upper half of the window, and translate it in the other half. After you finish the translation, save it, and the tool will generate a messages.mo file, this file is a compiled version of messages.po, and is the file that PHP will use later.

Mixing all together

Now it's time to see how to use this translation from our PHP script.

First we need to tell to gettext what locale we would like to use, Listing 2.

As you could see, our test script has some changes, I'll explain what it does line by line.

First line stands for set the language that we want to use to translate, remember `iso code language_ iso country code`, this is called *locale* also. The

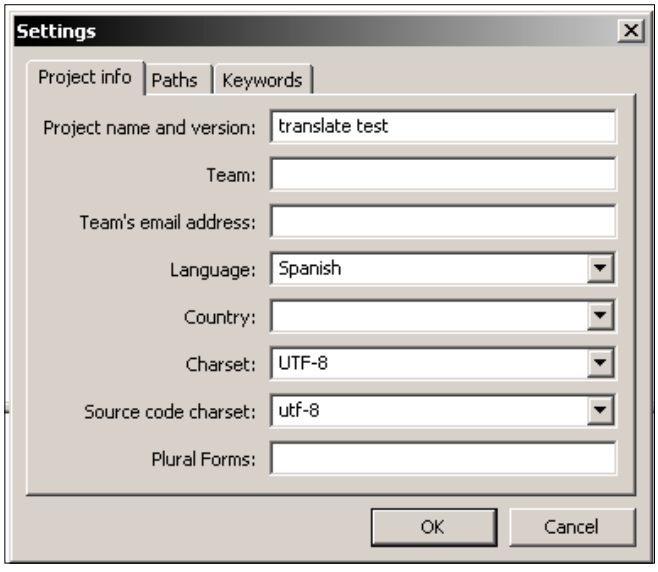


Figure 1. *Settings – information on project*

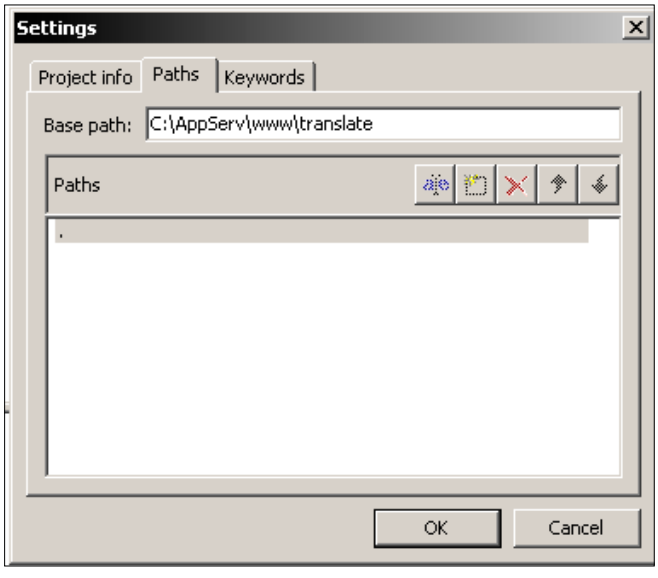


Figure 2. *Settings – paths*

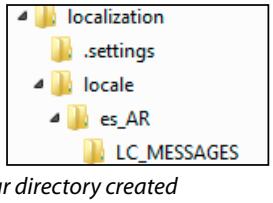


Figure 3. *Our directory created*

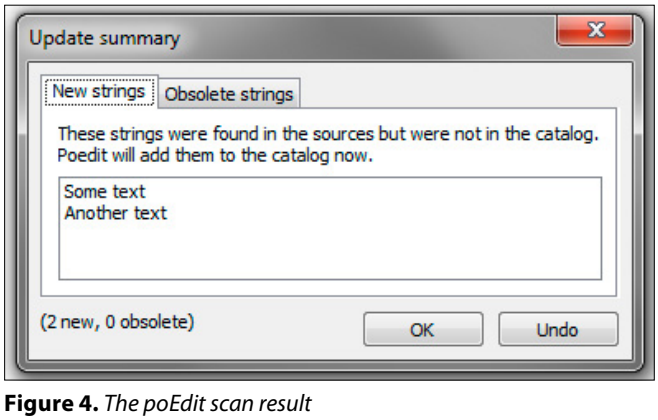


Figure 4. *The poEdit scan result*

Listing 2. *What locale we use*

```
<?php
    $l = "es_AR";

    putenv('LC_ALL='. $l);

    setlocale(LC_ALL, $l);

    bindtextdomain("messages", "./locale");

    textdomain("messages");


    echo "Some text translated: " . _("Some text");

    echo "<BR />";

    echo "Another text translated: " . _("Another text");
?>
```

next two lines actually set the current locale to the value that we specified.

The bindtextdomain call creates a text domain which will use our *messages.mo* file in our locale directory. The textdomain function selects that domain as the default domain. Yep, ok, but what is a text domain? A text domain is a set of translatable messages, it's sort of text catalog with messages that we can use (remember *poEdit*, when we were translating and I'm sure that you will figure it out). So far so good, let's back to our test PHP script.

We are ready now to execute it; we will obtain something like this:

```
Some text translated: un texto
Another text translated: otro texto
```

Notice that the text `inside _("")` was translated using the catalog that we made with poEdit. Works like a charm! Congratulations, you have created your first L10N compatible PHP script.

Some recommendations:

1. It's highly recommendable to put all the gettext initializing functions in another file, and then include it when necessary.
2. In order to add new text to the catalog, simply open the messages.po file with poEdit, the tool will scan again searching for new messages to translate.
3. Remember to restart your webserver after update your messages file.

Summary

This was just an introduction of what you can get from Gettext PHP extension and I sincerely hope you have found it usefull, you need to know that there are many other features to make L10N easy to implement. Remember that L10N it's not just about the language. Currencies, schedule, hours, and other aspects should be taken in account too, please keep reading and investigating and soon you'll become an expert!

CLAUDIO CORLATTI

Born in Argentina, the author has been in love with programming since his early schooldays. Has been working since 2006 in one of the largest retail company of his country as a programmer. Always passionate about new technologies, has been working in various personal and freelance projects too.
Now working in a new expenses tracking and saving web application integrated with banks information at www.al-centavo.com

Contact the author at: corlatti@gmail.com
or http://ar.linkedin.com/in/claudiocorlatti

Hit the books

- Gettext PHP extension – *http://php.net/manual/en/book.gettext.php*
- Textdomain – *http://linux.die.net/man/3/textdomain*
- Locale – *http://en.wikipedia.org/wiki/Locale*
- I12N and L10N – *http://en.wikipedia.org/wiki/Internationalisation_and_localisation*
- Language localization – *http://en.wikipedia.org/wiki/Language_localisation*
- poEdit – *http://www.poedit.net/*

What do all these have in common?

They all use Nipper Studio

to audit their firewalls, switches & routers

Nipper Studio is an award winning configuration auditing tool which analyses vulnerabilities and security weaknesses. You can use our point and click interface or automate using scripts. Reports show:

1) Severity of the Threat & Ease of Resolution

2) Configuration Change Tracking & Analysis

3) Potential Solutions including Command Line Fixes to resolve the Issue

Nipper Studio doesn't produce any network traffic, doesn't need to interact directly with devices and can be used in secure environments.

SME pricing from £650 scaling to enterprise level

evaluate for free at www.titania.com

2012 Computing Security Awards
WINNER
Enterprise Security Solution of the Year

2012 Computing Security Awards
WINNER
Network Security Solution of the Year

2012 Computing Security Awards
Runner-up
SME Security Solution of the Year

titania nipper STUDIO

CLOUD

www.titania.com
T: +44 (0) 1905 888785

IOS Software Development with Adobe AIR

Adobe AIR is a great tool for porting games to mobile devices or even to design directly for them. AIR technology allows running your flash game on IOS, Android and yes ... BlackBerry with small modifications in your code.

Adobe AIR (Adobe Integrated Runtime) is a cross-platform run-time system developed by Adobe Systems. To develop AIR application you can use one of the following technologies:

- Flash / Flex / ActionScript
- HTML / JavaScript / CSS / Ajax
- PDF can be used as well with both

Apps developed with AIR can be launched on IOS, Android, or BlackBerry Tablet OS. But today we will only discuss the IOS segment. AIR apps can be published as native phone apps (for IOS it is .ipa). Now AIR 3.7 Beta is available for developers. Adobe did alot for this version and we have an ability to use the new SDK.

So, first download the latest SDK on this link *http://labs.adobe.com/technologies/flashruntimes/air/*.

Screen Resolution

It's great to create one graphic and display it on different screens. We can detect the device screen resolution with flash.system.Capabilities. This class provides properties that describe the system that is hosting an application. Now, we need to call just three methods to see the screen resolution and screen dpi. Save these values to constants of integer type (don't use type Number). Then we have the resolution in global constants. Using them, the resizing is much easier; also it's possible to get access to them in other places within the same class. The graphics can be added to a scene according to resolution or by resizing one set. See the Listing 1.

Please notice, that the screen resolution will be different in retina and non-retina mode. You can change this mode in the descriptor file which is named your_app_name.xml (this file is created automatically). To make

Listing 1. *Detect screen resolution*

```
private const xRes:int = Capabilities.screenResolutionX; //returns 640 pixels in width on Iphone 4 Retina
private const yRes:int= Capabilities.screenResolutionY; //returns 960 pixels in height on Iphone 4 Retina
private const dpi:int = Capabilities.screenDPI;//returns screen dpi
```

Listing 2. *Requested display resolution*

```
<iPhone>
     <requestedDisplayResolution>high</requestedDisplayResolution>
  </iPhone>
```

Listing 3. *Requested display resolution exclude devices*

```
<iPhone>
     <requestedDisplayResolution excludeDevices="iPad3  iPad4">high</requestedDisplayResolution>
  </iPhone>
```

your app work with retina mode, you should find tag `<requestedDisplayResolution>` and include value of that mode you are needing (high – for retina mode, standard – for non-retina mode; see Listing 2).

In AIR 3.6 was added new tag as child of element `<iPhone>`. This tag helps to switch different modes on different devices. In our example we have "high" resolution and have iPad3 and iPad4, so our app will be working on those devices in standard resolution mode (Listing 3).

Touch Input

We have an application and we need to be able to control it on devices which have no control buttons. For this issue, AIR allows you to listen and handle multitouch events. AIR can convert mouse events to touch events by default (for example mouse click, or dragging) which avoids rewriting code if there is no other necessery functionality. But AIR also has special multitouch events for handling any types of finger taps and gestures. So the first thing we need to do is to specify which types of finger taps will be handled. Multitouch.inputMode can take one of three properties:

- GESTURE – handles five events: `GestureEvent.GESTURE_TWO_FINGER_TAP`, `GesturePressAndTap.GESTURE_PRESS_AND_TAP`, `TransformGestureEvent.GESTURE_ROTATE`, `TransformGestureEvent.GESTURE_SWIPE`, `TransformGestureEvent.GESTURE_ZOOM`. `TOUCH_POINT` – handles eight events (of one type TouchEvent): `TOUCH_BEGIN`, `TOUCH_END`, `TOUCH_MOVE`, `TOUCH_OVER`, `TOUCH_OUT`, `TOUCH_ROLL_OVER`, `TOUCH_ROLL_OUT`, `TOUCH_TAP`.
- NONE – AIR just handle mouse events.

There are also methods for enabling touch dragging (Listing 4):

```
stopTouchDrag();
startTouchDrag();
```

Listing 4. *Using touch events*

```
package sdj
{
   import flash.display.Sprite;
   import flash.events.TouchEvent;
   import flash.ui.Multitouch;
   import flash.ui.MultitouchInputMode;
   public class Main extends Sprite
   {
      private const xRes:int = Capabilities.screenResolutionX;
      private const yRes:int = Capabilities.screenResolutionY;
      private const dpi:int = Capabilities.screenDPI
      public function Main():void
      {
         Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
         drawRect();
      }
      private function drawRect():void
      {
         var rect:Sprite = new Sprite();
         rect.graphics.beginFill(0x00000, 1);
         rect.graphics.drawRect((yRes -100)/2, (xRes -100)/2, 100, 100);
         rect.graphics.endFill();
         addChild(rect);
         rect.addEventListener(TouchEvent.TOUCH_BEGIN, drag);
         rect.addEventListener(TouchEvent.TOUCH_END, stop);
      }
      private function drag(e:TouchEvent):void
      {
         e.target.startTouchDrag(e.touchPointID);
      }
      private function stop(e:TouchEvent):void
      {
         e.target.stopTouchDrag(e.touchPointID);
      }
```

Listing 5. *Set render mode*

```
<initialWindow>
    <renderMode>cpu</renderMode>
  </initialWindow>
```

As you can see, both drag methods use touchPointID values as parameters. What is touchPointID? This property is responsible for correct event handling. Because the devices have multitouch, we have to know when react. This property is more critical for `TouchEvent.TOUCH _ MOVE` and drag methods.

Render Mode

This is probably the most important issue, since performance is dependant on the correct render mode in different cases. And when should each mode be used? CPU is used when an application mainly uses vector graphics. It can cause performance decrising when bitmap graphics are used in this mode.

If the application does not contain a lot of vector graphics you can safely use this mode. But if the application contains a lot of graphics and animation, it is better to convert it to bitmap "on the fly" and use the GPU mode.

You can certainly prepare a raster graphics in advance for different screens, but this method is less streamlined and requires more work, and also the. apk size will significantly increase.

For object optimization, cacheAsBitmap property can be used. This property needs to be set as TRUE just for display objects that have no: transformation, alpha changing, or nested animation. Those objects can only move on the stage. So be careful when using this property otherwise your application will be very slow. For objects that are transformed or have changed transparency, you can use the following: `my_mc.cacheAsBitmapMatrix = my_mc.transform.cocatenatedMatrix; my_mc.cacheAsBitmap = true;`. But this approach will only work in the GPU mode. You need to set the render mode in the application descriptor file (Listing 5). AIR 3.7 allows to force CPU render mode on required devices. You just need to include them between tags `<forceCPURenderModeForDevices>` (Listing 6).

Orientation

Changing screen orientation according to device orientation is important for usability. Please, take a look at the Figure 1 and remember all these orientations.

To determine when the device is changed, orientation AIR has flash.events.StageOrientationEvent. There are two ways we can re-orient stage objects: first – allow AIR do it itself; second – change objects size and orientation with Action Script 3.0. Let see the first technique (we will work with the app descriptor file). There are few cases are shown Listing 7-11. So, instead of using the last case I recommend to change the stage object properties programaticlly. Take a look at the code which handles all device orientations (Listing 12).

One app for Both iPhone and iPad

App can be deployed for iPhone (iPod Touch), iPad or for all those devices. Open the app descriptor file, find

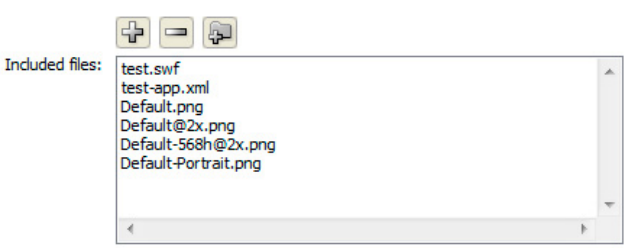


Figure 2. *Adding default images*

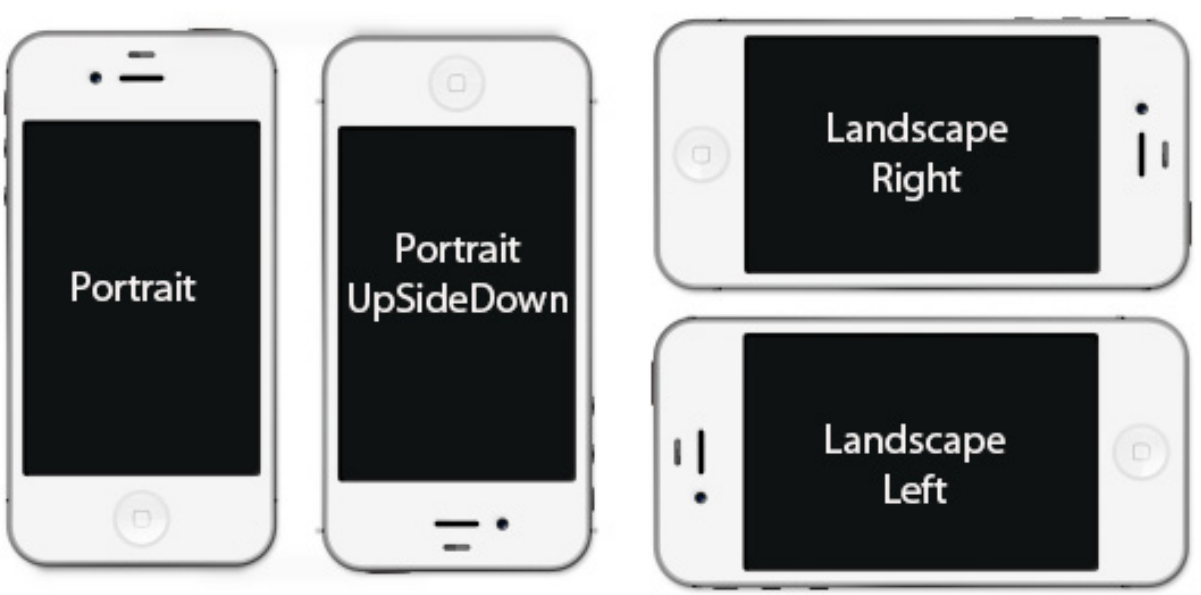


Figure 1. *Device orientations*

Listing 6. *Force CPU mode*

```
<renderMode>GPU</renderMode>
...
<iPhone>
   <forceCPURenderModeForDevices>iPad1,1 </
                    forceCPURenderModeForDevices>
</iPhone>
```

Listing 7. *Constantly oriented to LandscapeLeft*

```
<initialWindow>
    <autoOrients>false</autoOrients>
    <aspectRatio>landscape</aspectRatio>
  </initialWindow>
```

Listing 8. *Constantly oriented to Portrait*

```
<initialWindow>
    <autoOrients>false</autoOrients>
    <aspectRatio>portrait</aspectRatio>
  </initialWindow>
```

Listing 9. *Can be re-oriented to LandscapeLeft or LandscapeRight*

```
<initialWindow>
    <autoOrients>true</autoOrients>
    <aspectRatio>landscape</aspectRatio>
  </initialWindow>
```

Listing 10. *Can be re-oriented to Portrait or PortraitUpsideDown*

```
<initialWindow>
    <autoOrients>true</autoOrients>
    <aspectRatio>landscape</aspectRatio>
  </initialWindow>
```

Listing 11. *Can be re-oriented to all modes (not recommended)*

```
<initialWindow>
    <autoOrients>true</autoOrients>
    <aspectRatio>auto</aspectRatio>
  </initialWindow>
```

Listing 12. *Detecting device orientation changing*

```
package sdj
{
   import flash.events.StageOrientationEvent;
   public class Main extends Sprite
   {
      public function Main():void
      {
                stage.addEventListener(StageOr
                  ientationEvent.ORIENTATION_
                  CHANGE,OrientationChange);
      }

      private function OrientationChange(event:StageOr
                  ientationEvent):void
      {
         switch (stage.deviceOrientation)
         {
            case StageOrientation.DEFAULT:
            {
            // Default (Portrait) orientation.
            break;
            }
            case StageOrientation.ROTATED_RIGHT:
            {
            // LandscapeRight orientation.
            break;
            }
            case StageOrientation.ROTATED_LEFT:
            {
            //LandscapeLeft orientation.
            break;
            }
            case StageOrientation.UPSIDE_DOWN:
            {
            // PortraitUpsideDown orientation.
            break;
            }
         }
      }
   }
}
```

Listing 13. *Set device family property*

```
<iPhone>
    <InfoAdditions>
<![CDATA[<key>UIDeviceFamily</key>
<array>
<string>1</string>
<string>2</string>
</array>
<key>UIApplicationExitsOnSuspend</key><false/>]]>
    </InfoAdditions>
  </iPhone>
```

Listing 14. *Closing an application*

```
<key>UIApplicationExitsOnSuspend</key> <true/>
```

Listing 15. *Accelerometer example*

```
package sdj
{
   import flash.display.Sprite;
   import flash.events.AccelerometerEvent;
   import flash.sensors.Accelerometer;
   public class Main extends Sprite
   {
      private var accel:Accelerometer;
      public function Main():void
      {
         if (Accelerometer.isSupported)
         {
            accel= new Accelerometer();
            accel.addEventListener(AccelerometerEv
                  ent.UPDATE, accelUpdate);
         }
      }
      private function accelUpdate
                  (e:AccelerometerEvent):void
      {
         outputField.text = new String(e.accelerationX);
         outputField.text = new String(e.accelerationY);
         outputField.text = new String(e.accelerationZ);
      }
   }
}
```

tag `<iPhone>` then `<InfoAdditions>`. There you can see CDATA tag. For supporting iPhone (iPod Touch) family there should be `<string>1</string>`, for iPad – `<string>2</string>`. We have both lines, so our app will work on all devices (Listing 13).

Exiting Application

When the Home button is pressed on an IOS device, the app frame rate drops to 0 (paused). If you need to completely close your app, set the `UIApplicationExitsOnSuspend` property to true (Listing 14).

Launch Image

While the app is loading the user will see a black screen. To avoid this, you can load an image that will displayed until your app is completely loaded. Figure 2 shows how to add launch images to .apk using Adobe Flash Froffesional CS 5.5. Open the AIR IOS Settings window and include images.

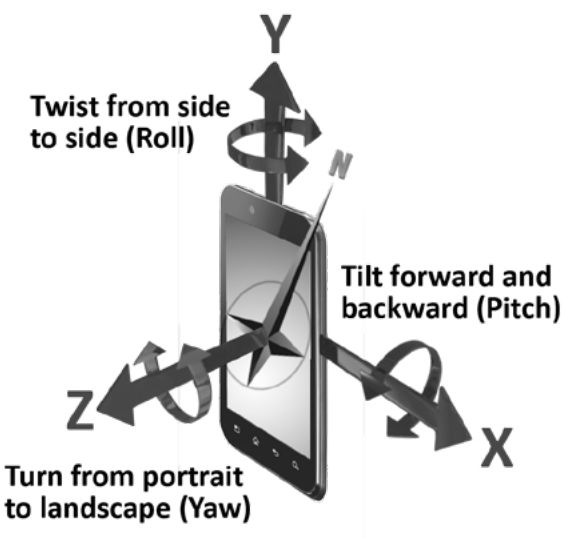


Figure 3. *Accelerometer demonstration*

Source links

- *http://help.adobe.com/en_US/air/build/WS901d38e593cd1bac1e63e3d129907d2886-8000.html#WS901d38e593cd1bac58d08f9112e26606ea8-8000*
- *http://spb-global.com/uncategorized/invensense-introduces-the-worlds-first-motionapps-platform-for-embedded-system-developers/*

Accelerometer

Accelerometer is widely used for game control. So AIR also has Accelerometer API. To work with it, we need two classes: flash.sensors.Accelerometer and flash.events.AccelerometerEvent. Accelerometer axis as shown Figure 3 and Listing 15.

Summary

So, we have examined the basics of AIR technology. These are not all the features this platform provides. For more detailed information, visit the Adobe website. I will not say this approach is better than using Objective C, because it would not be true. But this technology has advantages for some tasks and in certain situations. If you have your own game written in Flash, you can try to run it on IOS. It does not take much time and you can immediately see the result. Have fun!

SOSENYUK OLEG

The author is from Lutsk, Urkaine. Oleg works independently, as a Freelance Flash Developer.

Table 1. *Sizes of launch images*

File name	Image size	Usage
Default.png	320 x 480	iPhone, standard resolution
Default@2x.png	640 x 960	iPhone, high resolution
Default-568h@2x.png	640 x 1136	iPhone, high resolution
Default-Portrait.png	768 x 1004 (AIR 3.3 and earlier) 768 x 1024 (AIR 3.4 and higher)	iPad, portrait orientation
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 and earlier) 1536 x 2048 (AIR 3.4 and higher)	iPad, high resolution, portrait orientation
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 and earlier)768 x 1024 (AIR 3.4 and higher)	iPad, upside down portrait orientation
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 and earlier) 1536 x 2048 (AIR 3.4 and higher)	iPad, high resolution, upside down portrait orientation
Default-Landscape.png	1024 x 768	iPad, left landscape orientation
Default-LandscapeLeft@2x.png	2048 x 1536	iPad, high resolution, left landscape orientation
Default-LandscapeRight.png	1024 x 768	iPad, right landscape orientation
Default-LandscapeRight@2x.png	2048 x 1536	iPad, high resolution, right landscape orientation

BUILDING BIG DATA SOLUTIONS WITH THE BEST OPEN SOURCE HAS TO OFFER

Open source technologies MongoDB and Hadoop offer completely customizable and flexible platforms to meet your business' big data needs. Appnovation's big data solutions can process massive amounts of information and are built to be secure, stable and reliable in your IT enviroment. Appnovation delivers all our big data solutions in conjunction with our partners 10gen and Hortonworks.

In partnership with :

10gen

Hortonworks

Experts in Open Source Technologies

Creating Custom Solutions for Web, Mobile, Document Management, Big Data, Business Intelligence, E-commerce, Middleware and much much more!

Drupal · Alfresco · Sproutcore · MuleSoft · PhoneGap · Sencha · MongoDB · Hadoop

www.appnovation.com
ATLANTA · LONDON · VANCOUVER

APPNOVATION
TECHNOLOGIES

RFC for a New Init Mechanism

(Another Way to Initialize Objects in Objective C)

Writing one's own initialization methods in an Objective-C program while respecting the Apple – originally NeXT – official writing rules can be a tedious task.

This article aims at offering another, simpler and more generic approach, inspired from scripting languages like Python or Ruby.

Let's start with a short reminder of the current and historical systems. For initialization methods to be added to one of our classes, three rules should be complied with:

- Redefining the designated initialization method of the superclass is mandatory only if the class defines a new one (with a different prototype).
- If a variant of an initialization method differs from the original only by the number of expected parameters (and not by their nature), then the method that accepts a 'low' number of parameters has to call one that accepts just a little more.
  (Implication: all initialization methods have to call directly or indirectly the designated initialization method of their own class.)
- The designated initialization method of a class must call the designated initialization method of its superclass.

For those familiar with initialization issues, remember how boring it can be to have to choose between two exclusive methods which can both apply to becoming the designated initialization method?

Example:

- `(id) initProbeName: (NSString) *aName temperatureInDegreesFahrenheit: (double) aTemperature`
- `(id) initProbeName: (NSString) *aName temperatureInDegreesCelsius: (double) aTemperature`

Now imagine there is a way to write your initialization method only once, then use it in all your subclass hierarchies. The principle is simple and understated: it consists in providing one unique method to perform all initializations, whatever the properties you want to define. Furthermore, you don't have to create new initialization methods, again and again, every time you add a new class.

Implementation

First idea:

```
MyObject (subclass of NSObject)
```

Intuitively, the first idea that comes to the mind of an object-oriented developer who wants to add behaviors to a class is to 'subclass'.

In this particular case, we would have to subclass NSObject. That would probably work but that method is both inelegant and 'heavy':

- we are inserting a new class into an already complex hierarchy, and we must keep in mind from now on that we must prefer our own root class (rather than the official one, namely NSObject). Much ado about a tiny method…
- ii order to remain consistent, we would have to edit some of our parent class declarations for projects we want to maintain and in which we want to add our new mechanism.

This issue is conveniently solved by the concept of Category. Categories play at least three roles, the most prominent of which – and that is the heart of the issue at hand – is to enable developers to expand the list of

a class' methods (no new properties) without requiring that the developer own the source code. So this is the choice we retained. Better integration, NSObject+MyInit (category): For "not so modern" Objective C: Listing 1 and listing 2. Only for "modern" Objective C: Listing 3 and Listing 4. Few lines but a compendium of concepts (some recents and other more historical):

- Category (to augment a class whose, in this case, we do not have the source code) – simply with the syntax NewClassName (CategoryName),
- fast iteration – with for / in construction (I would prefer @for / in to show it's an Objective C exclusiveness),
- KVC (Key-Value Coding) – by using here the "setValue:forKey:" method,
- "block" – availability of blocks is not really "modern" but it is always interesting to show usage of this feature,
- exceptions handling – with @try / @catch construction and a NSException instance, => a lot of controversy about NSException use on iOS (discussion in *http://stackoverflow.com/questions/4310560/usage-of-nsexception-in-iphone-apps* with a link on a full article: *http://club15cc.com/code/objective-c/dispelling-nsexception-myths-in-ios-can-we-use-try-catch-finally*)
- modern dictionary manipulation syntaxes (incidentally of NSArray lists): initialization (container literals), access (subscripting),
- pseudo type instancetype (which is an efficient alternative of id for this kind of methods),
- more below…

Listing 1. *file NSObject+InitWithDictionary.h*

```
@interface NSObject (InitWithDictionary)

- (id) initWithDictionary: (NSDictionary *) aDict;

@end
```

Listing 2. *file NSObject+InitWithDictionary.m*

```
#import "NSObject+InitWithDictionary.h"

@implementation NSObject (InitWithDictionary)

- (id) initWithDictionary: (NSDictionary *) aDict
{
    if (self = [self init])
    {
        [aDict enumerateKeysAndObjectsUsingBlock: ^(id
                    iVarName, id initValue, BOOL *stop)
        {
            @try
            {
                [self setValue: initValue forKey:
                    iVarName];
            }
            @catch (NSException *exception)
            {
                NSLog(@"initWithDictionary assign
                    error (%@) value %@", [exception
                    reason], initValue);
            }
        } ];
    }
    return self;
}

@end
```

Listing 3. *file NSObject+InitWithDictionary.h*

```
@interface NSObject (InitWithDictionary)

- (instancetype) initWithDictionary: (NSDictionary *) aDict;

@end
```

Listing 4. *file NSObject+InitWithDictionary.m*

```
#import "NSObject+InitWithDictionary.h"

@implementation NSObject (InitWithDictionary)

- (instancetype) initWithDictionary: (NSDictionary *) aDict
{
    if (self = [self init])
    {
        for (NSString *iVarName in aDict)
        {
            @try
            {
                [self setValue: aDict[iVarName]
                    forKey: iVarName];
            }
            @catch (NSException *exception)
            {
                NSLog(@"initWithDictionary assign
                    error (%@) value %@", [exception
                    reason], aDict[iVarName]);
            }
        }
    }
    return self;
}

@end
```

I detect the begining of a controversy. With such a mechanism the user (the consumer) will reap the error messages (resulting from the exceptions).

That is why it is important to use wisely the NSString constants (which carry our property names). This will never be a panacea, but prevent a significant proportion of inevitable typographical errors.

Question: how-to assign default values to properties during object instanciation?

Probably 2 ways. I don't know at that time which one is the best (if there is one). Start redefining initWithDictionary: which begins to send message [super initWithDictionary: aDict] then continue giving initial values to properties.

The other way you have to assign initial values for your objects is to redefine your own init method which – of course – starts with the message [super init]. It's possible simply because the main method, initWithDictionary:, send message to self.

In fact I thinks both solutions are not really equivalent and meet two different needs. (Up to you to discover which one.)

And if it really is unavoidable, I do not see any cons-indication to mix the two principles (historical initialization

Listing 5. *file Person.h*

```
#import "NSObject+InitWithDictionary.h"

@interface Person : NSObject

@property (copy, nonatomic) NSString *lastname;
FOUNDATION_EXPORT NSString * const PROPNAME_LASTNAME;
@property (copy, nonatomic) NSString *firstname;
FOUNDATION_EXPORT NSString * const PROPNAME_FIRSTNAME;
@property (assign, nonatomic) char gender; //
                    'F'emale, 'M'ale, 'U'nknown
FOUNDATION_EXPORT NSString * const PROPNAME_GENDER;
@property (assign, nonatomic) int age;
FOUNDATION_EXPORT NSString * const PROPNAME_AGE;

- (id) init;
- (void) setAge: (int) newAge;

@end
```

Listing 6. *file Person.m*

```
#import "Person.h"

@implementation Person

@synthesize lastname = _lastname, firstname = _firstname,
                    gender = _gender, age = _age;

NSString * const PROPNAME_LASTNAME = @"lastname";
NSString * const PROPNAME_FIRSTNAME = @"firstname";
NSString * const PROPNAME_GENDER = @"gender";
NSString * const PROPNAME_AGE = @"age";

- (id) init
{
    if (self = [super init])
    {
        self.gender = 'U';
        self.age = -1;   // -1 for unknown age
    }
    return self;
}

- (void) setAge: (int) newAge
{
    if ((newAge == -1) || ((newAge >= 0) && (newAge <
                    150)))
    {
        _age = newAge;
    }
    else
    {
        NSLog(@"Age off limit (%i)", newAge);
    }
}

@end
```

Listing 7. *file main.m*

```
#import "Person.h"

int main (int argc, const char * argv[])
{
    @autoreleasepool {
        Person *yannick = [[Person alloc]
                    initWithDictionary: [NSDictionary
                    dictionaryWithObjectsAndKeys:
                    @"Yannick", PROPNAME_FIRSTNAME,
                    @"Cadin", PROPNAME_LASTNAME,
                    [NSNumber numberWithInt: 45],
                    PROPNAME_AGE, nil]];

        NSLog(@"Welcome to %@ %@ who is %d years
                    old", [yannick firstname], [yannick
                    lastname], [yannick age]);
    }
    return 0;
}
```

method prototype and call to the method described here). Such a situation could arise in the context of a Protocol.

If it requires the definition of a method, for example `initProbeName: tempatureInDegreesCelsius`, then why not write:

```
- (instancetype) initProbeName: (NSString)
*aName temperatureInDegreesCelsius: (double) aTemperature
{
... optional checking (like if (aTemperature > 1000) ...)
return [self initWithDictionary: @{ @"name" : aName,
                    @"temperature" : @(aTemperature) }];
}
```

The syntax @(count) is a good example of what the last release of Objective C admits, it's called boxing.

Usage

For "classic" Objective C: Listing 5-7. The idea offered here became even more interesting with the recent changes in Objective-C. In particular literals and container literals which permit a very short wrtiing of many expressions, especially NSDictionary instances. The same for NSArray, NSNumber, boolean constants, etc. *http://clang.llvm.org/docs/ObjectiveCLiterals.html*. For "modern" Objective C: Listing 8-10.

Listing 8. *file Person.h*

```
The same as for "classic" Objective C, only replace
(id) init;
by
(instancetype) init;
```

Listing 9. *file Person.m*

```
The same as for "classic" Objective C, only replace
(id) init
by
(instancetype) init
AND… You can remove all the "@synthesize" lines.
```

Listing 10. *file main.m*

```
#import "Person.h"

int main (int argc, const char * argv[])
{
    @autoreleasepool {
        Person *yannick = [[Person alloc]
initWithDictionary: @{ PROPNAME_FIRSTNAME :
@"Yannick", PROPNAME_LASTNAME : @"Cadin", PROPNAME_
                    AGE : @45 } ];

        NSLog(@"Welcome to %@ %@ who is %d years
                    old", yannick.firstname, yannick.
                    lastname, yannick.age);
    }
    return 0;
}
```

End of concept overviews:

- dot notation (not so recent).
- automatic (implicit) synthesize.
- boxing (in an example above).
- modern syntaxes (literals) of constants writing (NSNumber, booleans, etc).

Depending the release of Objective C you want to write for, you have to use the first syntax or you can prefer the second (for the latest releases). My own opinion (thanks to add yours ;-)

Pros

- simplicity.
- the KVC mechanism allow keeping custom initialization of each variable. It's not a short circuit.

Cons

(performance) slowness (but this needs to be qualified because it probably depends on ratio: class hierarchy depth – given the potentially large number of messages – from the number of values to allocate / initialize) the main reasons for the poor performances: NSExceptions management and use of the KVC mechanism.

For interested readers: *https://github.com/nicklockwood/BaseModel*. For discussion about const in Objective C: *http://stackoverflow.com/questions/538996/constants-in-objective-c*.

YANNICK CADIN

Yannick Cadin is 45 years old, including 27 devoted mainly to computer industry (hardware and software). Programming of UBI-SOFT's very first game as freelance developer. Many salary jobs in small or medium companies: video game publisher, professional software editor, textile CAM software editor, Decision Tools software editor, computer resellers or distributors, consulting companies, value-added resellers (VAR), mapping company and even in the public sector with the title of Chief Operating Officer at the Louvre Museum. Manager of MICRO REPONSE specialized in providing computers running NEXTSTEP / OpenStep. Positions held: most of them, sales engineer, developer, support technician, trainer, ... Freelance writer on a more or less regular basis for a dozen magazines for the last 26 years. Casual proofreader and speaker. Red Hat Linux, Ubuntu, LPI, * BSD and Apple certified. Currently Manager of Diablotin.

Dr.WEB®
20 years

Dr.Web SpIDer is 8-legged!

New Version 8.0

Security Space and Dr.Web Antivirus for Windows

Get your free 60-day license under https://www.drweb.com/press/
to protect your PC and your smartphone with Dr.Web!
Your promo code: ***Hakin9***

Protect your mobile device free of charge!
https://support.drweb.com/free_mobile/

Doctor Web is a Russian anti-virus vendor with a software development record dating back to 1992.
www.drweb.com

© Doctor Web
2003 — 2013

nicky in the clouds

scalable creativity

Get your Projects Off the Ground !

www.nickyintheclouds.com/sdj2013

LIKE US ON facebook

DATACENTRE
TRANSFORMATION
CONFERENCE

9 July 2013
aql Conference Auditorium,
Salem Church, Leeds

Excellent Networking Opportunities!

Find your way through the data centre maze

The modern datacentre is transforming, impacted by efficiency demands, new design concepts, legislation and IT innovation, solutions that worked yesterday can soon be rendered uncompetitive or even obsolete.

The Datacentre Transformation Conference sets out to help steer you through the maze of requirements and developments.

Register today to secure your place

www.dtconference.com

Sponsored by:

3M adapt Airedale APPLIED THERMAL INNOVATION

APC by Schneider Electric ARISTA EMERSON Network Power

Gigamon RITTAL Server Technology Quality Rack Power Solutions

In Association with:

datacentrealliance The Data Centre Industry Association

UNIVERSITY OF LEEDS

datacentre solutions DCS EUROPE

AN ANGEL EVENT