

Grailsの
RDBシヤナイアレ

masaki@metabolics.co.jp

2010.12.09

非RDBデータストア

- いろんな種類がある (とそれぞれ主張している)
- メモリ
- キー・バリュー・ストア
- ドキュメント指向 (JSON, BSON, ...)
- マッシュブ・パラレル
- 融合型
-

(おおまかな) 特徴

- 動的
- スケーラブル
- シンプル
- 標準はない
- 緩めのデータ管理

G* 的可用性

- たいていは, もちろんJavaから使える
- ということは, Groovyからも使える
- 当然, Grailsからも使える
- でもドメイン・クラスをRDBと透過的に非RDBに永続化できると嬉しいかも

プラグイン

- Hadoop
- Cassandra
- MongoDB
- CouchDB
- Redis
-

inconsequential

- RDB/非RDBデータストアを統合的に扱うフレームワーク
- as SpringFramework and Grails
- by Graeme Rocher, et. al.
- github.com/grails/inconsequential

- spring-datastore-core
 - spring-datastore-xxx
 - grails-datastore-gorm-xxx
 - grails-plugins/xxx
- xxx = {appengine*, cassandra*, gemfire, jcr, mongo, redis, riak, simple*} (as of 2010.12.05)
- *はGORMサポートは現時点ではなし
- simpleはmockなどで使うオンメモリ・データストア

datastore-core

- org.springframework.datastore.mapping.core
- Datastore
- Session
- interface, abstract class
- データストアの種類ごとのサポート・クラス
- トランザクションやバリデーションの基礎クラス

datastore-xxx

- datastore-xxxのクラスの実体化
 - Datastore
 - Session
- 必要に応じて, Transaction, Query, Persister, Collectionなどの実体化

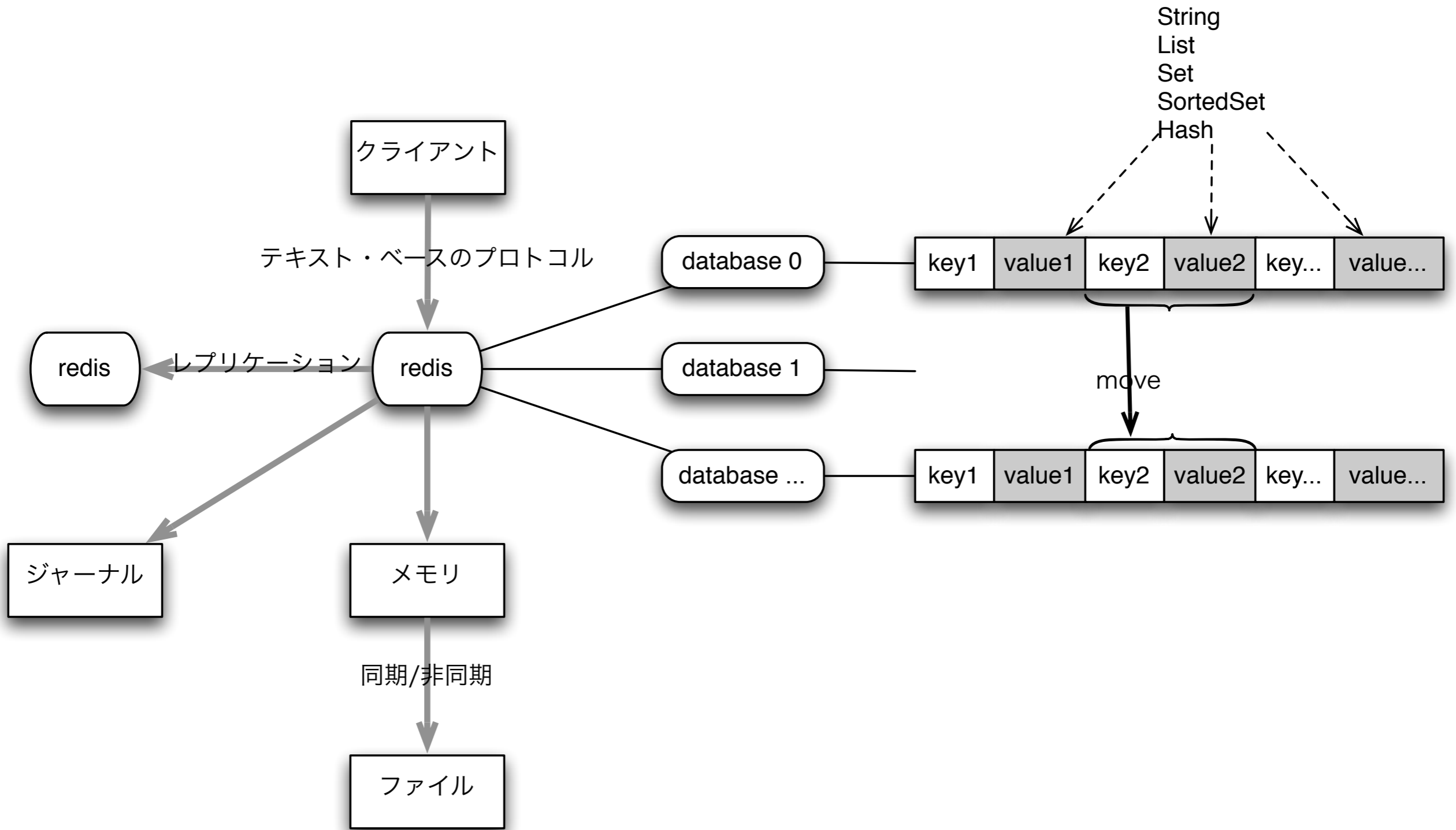
- **datastore-gorm-xxx**
 - GORM特有の実装
- **grails-plugins/xxx**
 - Grailsプラグインの実装
 - 通常は非常に薄い

GORM for Redis

redis

- vmwareがサポート
- KVSだが
- 多様なデータ型
 - String, List, Set, SortedSet, Hash
- replication, journaling, async. save

一目で分かるRedis



server

- <http://code.google.com/p/redis/>
- ソース・コードをダウンロードして
- `$ make; $./redis-server`

client

- `$ grails create-app xxx`
- `$ grails install-plugin redis`
- 適当にドメイン・クラスを作って
 - `static mapWith = "redis"`
- 適当にテスト・クラスを作って
 - `$ grails test-app`
- <http://dl.dropbox.com/u/265111/Redis-ja/guide/index.html>



Redis GORM - Reference Documentation

Authors: Graeme Rocher

Version: 1.0.0.M1

Translation into Japanese by: YAMADA Masaki masaki@metabolics.co.jp

目次

[1. はじめに](#)

[1.1 GORM for Hibernate との互換性](#)

[2. 使ってみよう](#)

[2.1 Redis を単独で使う](#)

[2.2 Hibernate と Redis を一緒に使う](#)

[3. オブジェクト・マッピング](#)

[4. クエリ](#)

[4.1 クエリの制限](#)

[5. トランザクション](#)

[6. Redis 固有の拡張](#)

[7. 低レベル API](#)

[8. テスト](#)

Grails to Redis mapping

```
package jp.co.metabolics.redistest
```

```
class Person {
```

```
    static mapWith = "redis"
```

```
    String surname
```

```
    String surnameKana
```

```
    String forename
```

```
    String forenameKana
```

```
    Date dateOfBirth
```

```
    Boolean gender
```

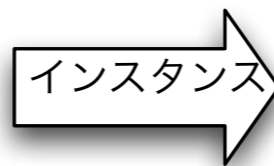
```
    static mapping = {
```

```
        surname index:true
```

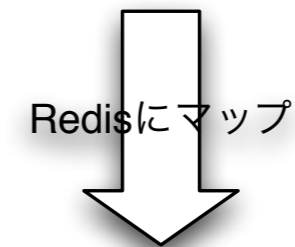
```
        forename index:true
```

```
    }
```

```
}
```



[surname:"山田", surnameKana:"やまだ", ...] as Person



jp.co.metabolics.redistest.Preson:1	[surname:"山田", surnameKana:"やまだ", ...]
jp.co.metabolics.redistest.Preson:id:1	["1"] as Set
jp.co.metabolics.redistest.Preson:surname:山田	["1"] as Set
jp.co.metabolics.redistest.Preson:forename:正樹	["1"] as Set

key

value

} インデクス

結果

- 一万個の単純なインスタンスをsave() (最後にflush:true) するのに20sec !
- その後にfindとかしようとすると200sec !!
- いちいちflush:trueすると, 230sec ... orz
- GroovyでJedis (RedisのJava API) を直接使
うと, 2sec ...
- あり得ん

GORM for MongoDB

MongoDB

- JSONオブジェクトをキーを付けてバイナリ形式 (BSON) で格納する
- KVS
- ドキュメント指向 (意味がよく分かん:-)
- SQLも使える
- indexの指定は不要
- sharding, replication, map/reduceなど多機能

server

- <http://www.mongodb.org/downloads>
- 自分のプラットフォームに合ったバイナリをダウンロードして,
- `$./bin/mongod --dbpath data/db`

client

- `$ grails create-app xxx`
- `$ grails install-plugin mongodb`
- 適当にドメイン・クラスを作って
 - `static mapWith = "mongo"`
- 適当にテスト・クラスを作って
 - `$ grails test-app`
- <http://dl.dropbox.com/u/265111/MongoDB-ja/guide/index.html>



GORM for Mongo - Reference Documentation

Authors: Graeme Rocher

Version: 1.0.0.M1

Translation into Japanese by: YAMADA Masaki masaki@metabolics.co.jp

目次

[1. はじめに](#)

[1.1 GORM for Hibernate との互換性](#)

[2. 使ってみよう](#)

[2.1 Mongo を単独で使う](#)

[2.2 Mongo と Hibernate を一緒に使う](#)

[2.3 より進んだ構成](#)

[3. ドメイン・クラスを Mongo の Collection にマッピングする](#)

[3.1 識別子の生成](#)

[3.2 問い合わせのインデキシング](#)

[3.3 WriteConcern をカスタマイズする](#)

[4. 低レベル API](#)

[5. トランザクション](#)

[6. 単体テスト](#)

結果

- 一万個の単純なインスタンスをsave() (最後にflush:true) するのに3sec
- ただしflushしていないエンティティに対しては, 操作 (find など) できない...
- いちいちsave(flush:true)すると34sec
- まあ普通
- ちなみにhsqldbだと20secくらい

まとめ

- inconsequential, 大丈夫か?
- どれだけ, RDBと異なる面白い使い方がGrails上でできるか?
- 今のGORMでいいのか?
- 具体的な実装実績が欲しいねえ
- g線路は続くよ, どこまでも
- <http://grailsgoeson.metabolics.co.jp/>