Painfree Object-Document Mapping for MongoDB

Philipp Krenn @xeraa

geecon


Vienna

ViennaDB
Papers We Love Vienna


ecosio

Who uses

JPA?

Who uses

MongoDB?

Who has heard of

Morphia?

Like JPA for MongoDB

...but better

```
@OneToMany(mappedBy = "destCustomerId")
@ManyToMany
@Fetch(FetchMode.SUBSELECT)
@JoinTable(name = "customer_dealer_map",
    joinColumns = {
       @JoinColumn(name = "customer_id",
       referencedColumnName = "id")},
    inverseJoinColumns = {
       @JoinColumn(name = "dealer_id",
       referencedColumnName = "id")})
private Collection<Client> dealers;
```


I'm okay.

Relations vs Objects

Ted Neward: ORM is "The Vietnam of Computer Science"

http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx

MongoDB

Table = Collection

Schemaless

Row = Document

JSON

```
{
    "name": "Philipp",
    "isAlive": true,
    "age": 30,
    "height_cm": 181.5,
    "address": {
       "city": "Vienna",
       "postalCode": "1190"
    },
    "phoneNumbers": [ {
       "type": "mobile",
       "number": "+43 123 4567890"
    } ]
}
```

MongoDB Java driver

```
List<BasicDBObject> phoneNumbers = new ArrayList<>();
phoneNumbers.add(
    new BasicDBObject("type", "mobile")
        .append("number", "+43 123 4567890"));

BasicDBObject document = new BasicDBObject("name", "Philipp")
    .append("isAlive", true)
    .append("age", 30)
    .append("height_cm", 181.5f)
    .append("address",
        new BasicDBObject("city", "Vienna")
            .append("postalCode", "1190"))
    .append("phoneNumbers", phoneNumbers);
```

Morphia

Object-Document Mapping

POJO + Annotations

Object-Relational Mapping

Features

Lightweight
Type safe & preserving

Required libraries

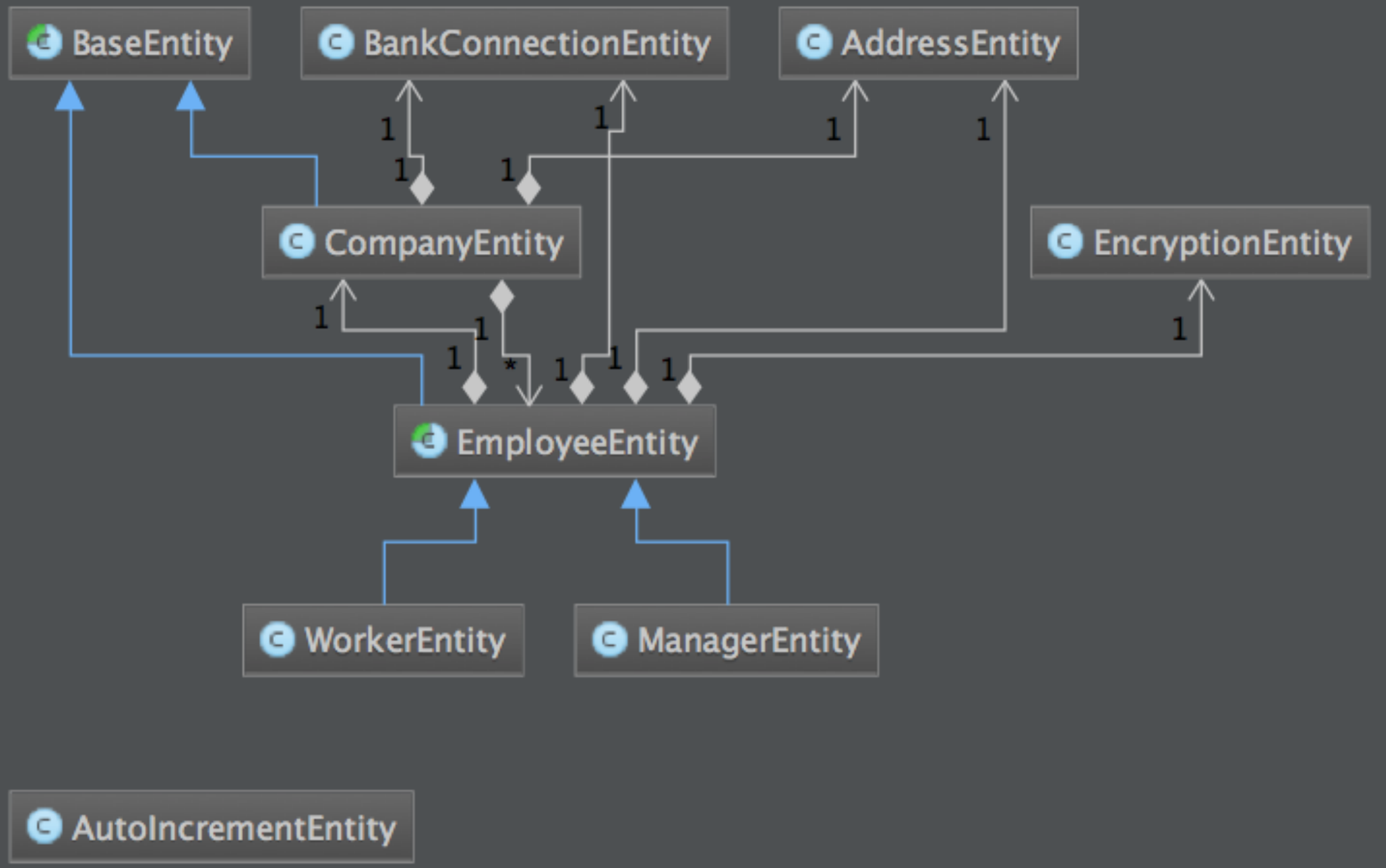
https://github.com/mongodb/mongo-java-driver (3.0.1)

+

https://github.com/mongodb/morphia (1.0.0-rc0)

Show me some code

https://github.com/xeraa/morphia-demo


BaseEntity
BankConnectionEntity
AddressEntity
CompanyEntity
EncryptionEntity
EmployeeEntity
WorkerEntity
ManagerEntity
AutoIncrementEntity
1
*

```
$ gradle test
```

Standalone or embedded MongoDB

Annotations

Collections

```
@Entity(
   value = "company",
   noClassnameStored = true
)
public class CompanyEntity {

    @Id
    protected ObjectId id;

    public CompanyEntity() { }
```

Do not

use dashes in the collection name

```
failbowl:~(master!?) $ jsc
> Array(16)
,,,,,,,,,,,,,,,
> Array(16).join("wat")
watwatwatwatwatwatwatwatwatwatwatwatwatwatwatwat
> Array(16).join("wat" + 1)
wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1wat1
wat1
> Array(16).join("wat" - 1) + " Batman!"
```

https://www.destroyallsoftware.com/talks/wat

Do not

copy paste the `value` attribute

Do not

use `@Id` as String (without reason)

Polymorphism

```
public abstract class EmployeeEntity {
    protected String name;
}

public class ManagerEntity extends EmployeeEntity {
    protected Boolean approveFunds;
}

public class WorkerEntity extends EmployeeEntity {
    protected Integer yearsExperience;
}
```

RDBMS

1. Union table with (many) `NULL` values

RDBMS

2. Concrete instances without common queries

RDBMS

3. Base table joined with all subtables


HELLO
my
name
is
BOB

```
@Entity(value = "employee", noClassnameStored = false)
public abstract class EmployeeEntity {
    @Id
    protected ObjectId id;

    protected String name;
}

public class ManagerEntity extends EmployeeEntity {
    protected Boolean approveFunds;
}

public class WorkerEntity extends EmployeeEntity {
    protected Integer yearsExperience;
}
```

```
{
  "_id": ObjectId("5461c8bf9e2acf32ed50c079"),
  "className": "net.xeraa.morphia_demo.entities.ManagerEntity",
  "name": "Peter",
  "approveFunds": true
}
{
  "_id": ObjectId("524d9fe7e4b0f8bd3031f84e"),
  "className": "net.xeraa.morphia_demo.entities.WorkerEntity",
  "name": "Philipp",
  "yearsExperience": 10
}
```

`className` and refactoring?

Properties

```
protected String firstname;

@AlsoLoad("lastname")
protected String surname;

protected Boolean approveFunds;

@Property("hire")
protected boolean managerCanApproveHires;

public EmployeeEntity setFirstname(String firstname) {
    this.firstname = firstname;
    return this;
}
```

Do

trim property names (MMAPv1)

Do

use object data types

Do

provide chainable setters

Indexes

```
@Entity(value = "employee", noClassnameStored = false)
@Indexes(@Index(name = "name", fields = {
            @Field(value = "surname"),
            @Field(value = "firstname")
}))
public class EmployeeEntity {

    protected String firstname;

    protected String surname;

    @Indexed(unique = true, sparse = false)
    protected String email;
```

Optimistic locking

vs pessimistic locking in RDBMS

```
@Version
private long version;
```

```
{
   ...
   "version": NumberLong("1")
}
```

Anti JOIN

```
public class EmployeeEntity {

    @Reference
    protected CompanyEntity company;

    @Embedded
    protected BankConnectionEntity bankConnection;
```

Queries

Save or upsert

```
public ObjectId persistCompanyEntity(CompanyEntity company) {
    mongoDatastore.save(company);
    return company.getId();
}

public ObjectId persistManagerEntity(ManagerEntity manager) {
    mongoDatastore.save(manager);
    return manager.getId();
}

public ObjectId persistWorkerEntity(WorkerEntity worker) {
    mongoDatastore.save(worker);
    return worker.getId();
}
```

Get data

```
public EmployeeEntity findByEmail(final String email) {
    return mongoDatastore.find(EmployeeEntity.class)
            .field("email").equal(email).get();
}

public List<EmployeeEntity> getAllEmployees() {
    return mongoDatastore.createQuery(EmployeeEntity.class).asList();
}

public List<ManagerEntity> getAllManagers() {
    return mongoDatastore.createQuery(ManagerEntity.class)
            .disableValidation()
            .field("className").equal(ManagerEntity.class.getName())
            .asList();
}
```

Watch out

`.equal() != .equals()`

Trust your compiler

Performance

Normalize fields

`email.toLowerCase()` **before saving**

More queries

Regular expressions

`.exists(), .doesNotExist()`

`.greaterThan(), .hasAnyOf(),...`

`.sort(), .skip(), .limit()`

More features

Capped collections
GridFS
Aggregation framework
Geo locations

Patterns

Base Class

```
public abstract class BaseEntity {

    @Id
    protected ObjectId id;

    protected Date creationDate;
    protected Date lastChange;

    @Version
    private long version;
```

```
public BaseEntity() {
    super();
}

// No setters
public ObjectId getId() {
    return id;
}
public Date getCreationDate() {
    return creationDate;
}
public Date getLastChange() {
    return lastChange;
}
```

```
    @PrePersist
    public void prePersist() {
        this.creationDate =
            (creationDate == null) ? new Date() : creationDate;
        this.lastChange =
            (lastChange == null) ? creationDate : new Date();
    }

    public abstract String toString();

}
```

```
public <E extends BaseEntity> ObjectId persist(E entity) {
    mongoDatastore.save(entity);
    return entity.getId();
}

public <E extends BaseEntity> long count(Class<E> clazz) {
    return mongoDatastore.find(clazz).countAll();
}

public <E extends BaseEntity> E get(Class<E> clazz, final ObjectId id) {
    return mongoDatastore.find(clazz).field("id").equal(id).get();
}
```

Converters

Option 1

Not readable or searchable in the database

```
@Serialized
protected BigDecimal bonus;
```

Option 2

Fugly

```
@Transient
protected BigDecimal salary;
protected String salaryString;
```

```java
@PrePersist
public void prePersist() {
    super.prePersist();
    if (salary != null) {
        this.salaryString = this.salary.toString();
    }
}

@PostLoad
public void postLoad() {
    if (salaryString != null) {
        this.salary = new BigDecimal(salaryString);
    } else {
        this.salary = null;
    }
}
```

Option 3

Yes!

```
@Converters({BigDecimalConverter.class})
public class WorkerEntity extends EmployeeEntity {
    protected BigDecimal dailyAllowance;
```

```java
public class BigDecimalConverter extends TypeConverter implements SimpleValueConverter {
    @Override
    public Object encode(Object value, MappedField optionalExtraInfo) {
        if (value == null) {
            return null;
        }
        return value.toString();
    }

    @Override
    public Object decode(Class targetClass, Object fromDBObject,
            MappedField optionalExtraInfo) throws MappingException {
        if (fromDBObject == null) {
            return null;
        }
        return new BigDecimal(fromDBObject.toString());
    }
}
```

Auto Increments

Approach

`AutoIncrementEntity` **keeps track of the current version for a class or a subset**

Approach

On persist get the current version from `AutoIncrementEntity` and increment it

Approach

Set the value in the target entity

```java
@Entity(value = "ids", noClassnameStored = true)
public class AutoIncrementEntity {

    @Id
    protected String key;

    protected long value = 1L;
```

```
protected AutoIncrementEntity() {
    super();
}

public AutoIncrementEntity(final String key) {
    this.key = key;
}

public AutoIncrementEntity(final String key, final long startValue) {
    this(key);
    value = startValue;
}

public Long getValue() {
    return value;
}
```

Example

Incrementing employee number within a company

Example

Saving the entity: Start counting at 1000 & get the current number

```
long employeeNumber =
    genericPersistence.generateAutoIncrement(
    CompanyEntity.class.getName() + "-" +
    company.getId(), 1000L);
```

```
public long generateAutoIncrement(final String key, final long minimumValue){

    final Query<AutoIncrementEntity> query = mongoDatastore
            .find(AutoIncrementEntity.class).field("_id").equal(key);

    final UpdateOperations<AutoIncrementEntity> update = mongoDatastore
            .createUpdateOperations(AutoIncrementEntity.class).inc("value");

    AutoIncrementEntity autoIncrement = mongoDatastore.
            findAndModify(query, update);

    if (autoIncrement == null) {
      autoIncrement = new AutoIncrementEntity(key, minimumValue);
      mongoDatastore.save(autoIncrement);
    }

    return autoIncrement.getValue();
}
```

More

https://github.com/fakemongo/fongo

https://github.com/evanchooly/critter

Critter

```
Query<Query> query = ds.createQuery(Query.class);
query.and(
  query.criteria("bookmark").equal(bookmark),
  query.criteria("database").equal(database)
);

QueryCriteria criteria = new QueryCriteria(datastore);
criteria.and(
  criteria.bookmark(bookmark),
  criteria.database(database)
);
Query query = criteria.query().get();
```

Thanks!
Questions?

Now, later today, or @xeraa