Spring data

Евгений Борисов

bsevgeny@gmail.com

@jekaborisov

2 Слова о себе

Пишу курсы

Пишу код для JFrog-а

Синглтоны – не пишу, пью

Страдаю от аллергию на весну,

Но люблю спринг

Терминология

- Аппликация = приложение
- **А**йбернет = хибернет
- Штрудель =Собака
- Компонент – использую с любым ударением
- Пар**а**метр = Парам**е**тр
- Список пополняется…

Agenda

Времена JDBC

Времена JPA

Сегодня - Spring Data

Agenda

- Spring Data JPA (Hibernate)
- Spring Data Mongo
- Spring Data Neo4J
- Кассандра – только в виде фонта

Когда мы были молодые и с нами был JDBC

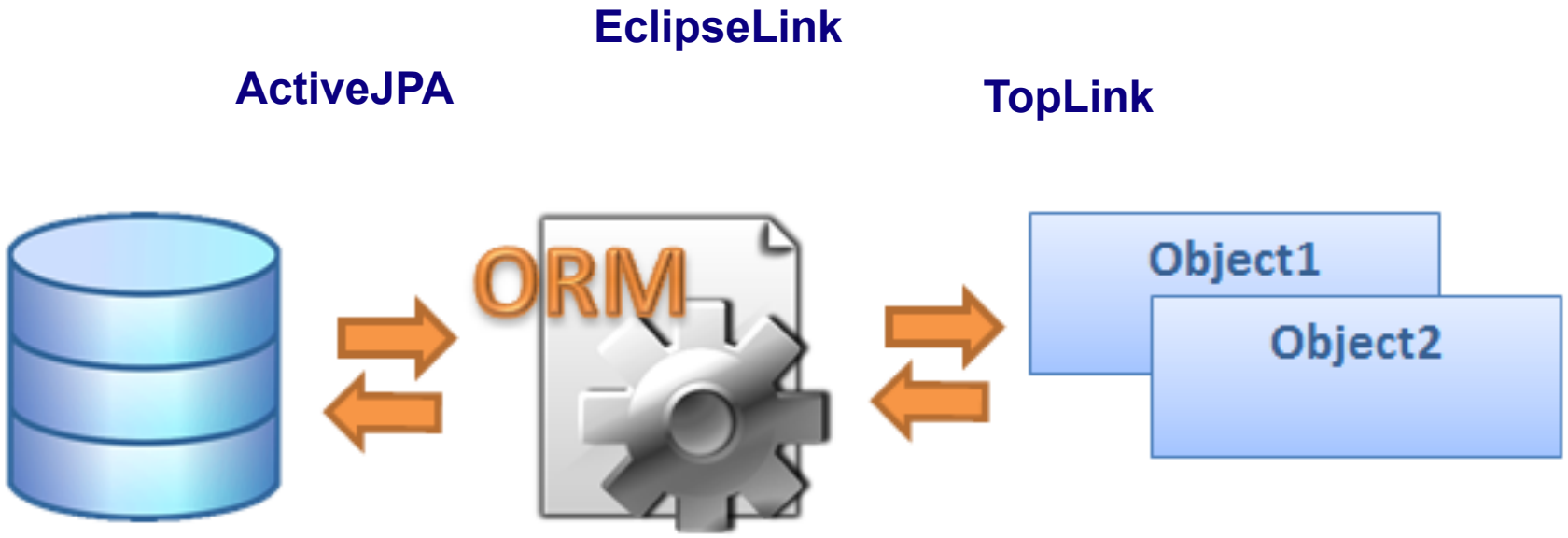
План работы:

1. Open the connection.
2. Create the statement.
3. Execute the statement.
4. Loop through the results.
5. Treat Exceptions.
6. Handle Transactions.
7. Close the connection.

Боль

```
try {
    connection = DriverManager.getConnection("jdbc:…");
    PreparedStatement pStm = connection.prepareStatement(
    "Select TITLE from BOOKS where price < ?");
    pStm.setInt(1, 100);
    ResultSet rs = pStm.executeQuery();
    while (rs.next()) {
        System.out.println(rs.getString("TITLE"));
    }
} catch (SQLException e) {
        // А что тут писать??? Кидаем дальше.
} finally {
    if (connection != null) {
        try {
                connection.close();
        } catch (SQLException e) {
                // Вообще нечего писать
        }
    }
}
```

Когда уходит детство приходит JPA…



А что сегодня в моде?

Document database

JSON documents

JSON queries

MongoDB

```
Mongo mongo = new Mongo(…);
DB db = mongo.getDB("myDatabase");
Collection collection = db.getCollection("myCollection");
DBObject query = new BasicDBObject();
query.put("address.city", "Kiev");
DBObject cursor = collection.find(query);
for (DBObject element : cursor) {
   // Map data onto object
}
```

Graph database

Nodes / Relationships

Traversals / Cypher / Gremlin

Neo4j

```
GraphDatabaseService database = new EmbeddedGraphDatabase(…);
Transaction tx = database.beginTx();
try {
    Node mrRodionov = database.createNode();
    mrRodionov.setProperty("name", "Andrey Rodionov");
    Node baruch = database.createNode();
    baruch.setProperty("name", "Baruch");
    Relationship friendship = mrRodionov.createRelationshipTo(baruch, FriendTypes.KNOWS);
    tx.success();
} finally {
    tx.finish();
}
```

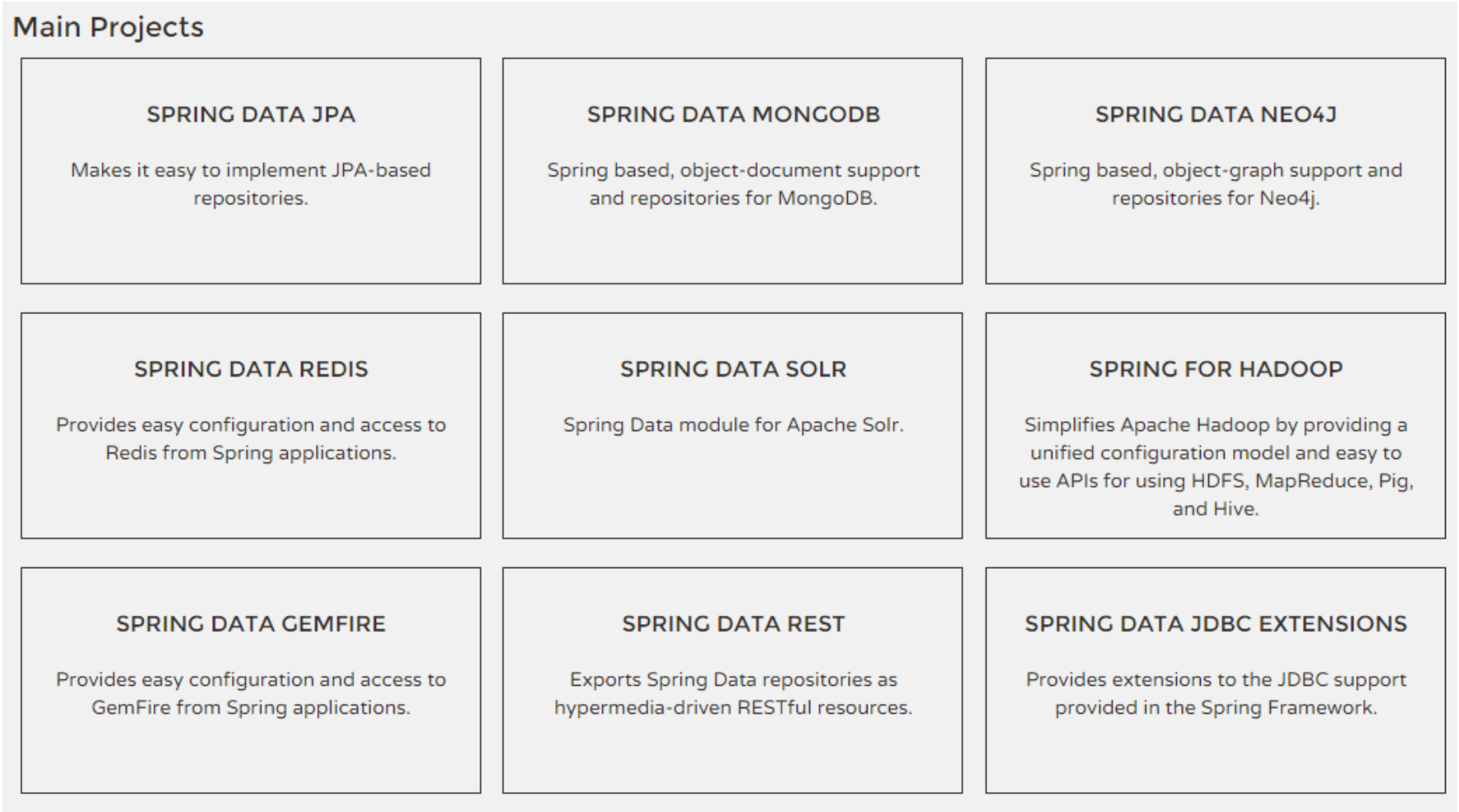
Можно ли тут применить JPA?

Вывод:

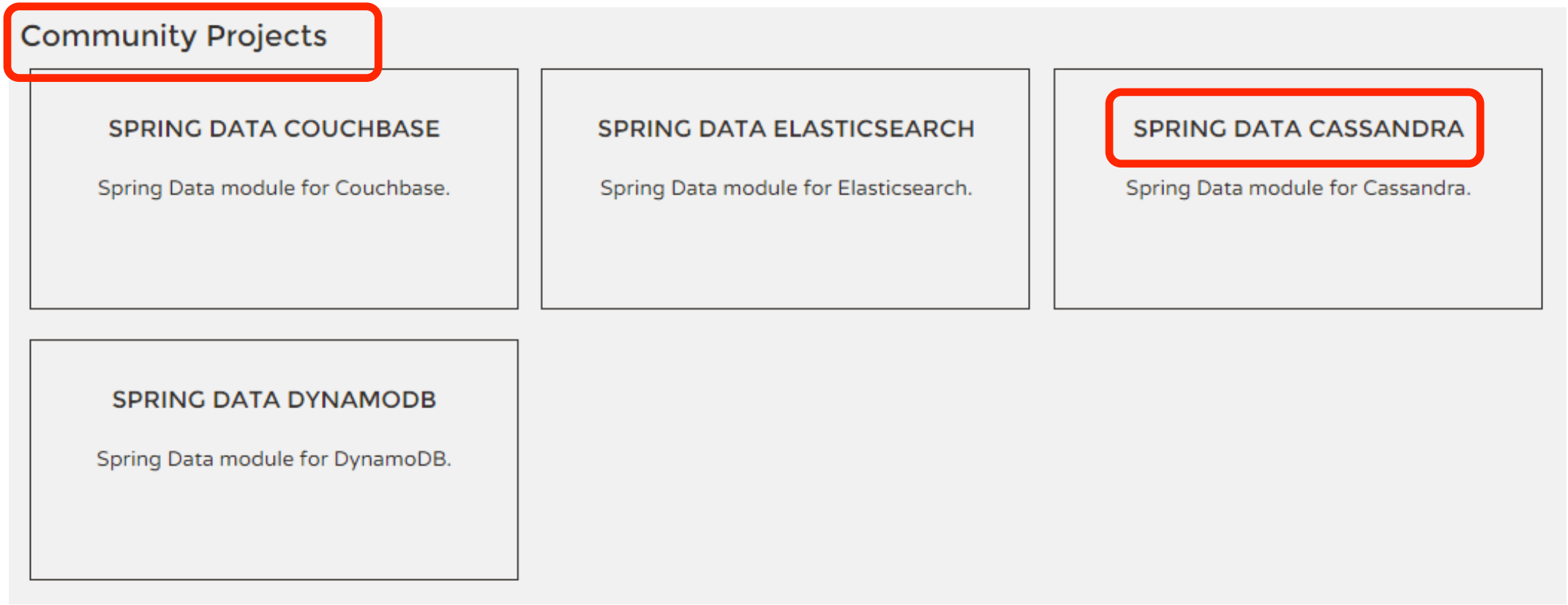
Пришло иное время, достал всех Хибернет Spring Data нам поможет ему ответить НЕТ

http://projects.spring.io/spring-data/

Проект Spring Data существует с 2008 года



А Кассандра?


Community Projects
SPRING DATA COUCHBASE
Spring Data module for Couchbase.
SPRING DATA ELASTICSEARCH
Spring Data module for Elasticsearch.
SPRING DATA CASSANDRA
Spring Data module for Cassandra.
SPRING DATA DYNAMODB
Spring Data module for DynamoDB.

Поехали…

А теперь Spring Data JPA

Релиз 1.0.0 вышел в 2011 году

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
    <version>1.7.0.RELEASE</version>
</dependency>
```

Загадочный CrudRepository

```
@NoRepositoryBean
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> {

  <S extends T> S save(S entity);
  <S extends T> Iterable<S> save(Iterable<S> entities);
  T findOne(ID id);
  boolean exists(ID id);
  Iterable<T> findAll();
  Iterable<T> findAll(Iterable<ID> ids);
  long count();
  void delete(ID id);
  void delete(T entity);
  void delete(Iterable<? extends T> entities);
  void deleteAll();
}
```

А ещё что-нибудь есть?

CrudRepository

PagingAndSortingRepository

JpaRepository

MongoRepository

Neo4jRepository

SimpleCassandraRepository

Да не вопрос
Полно всего

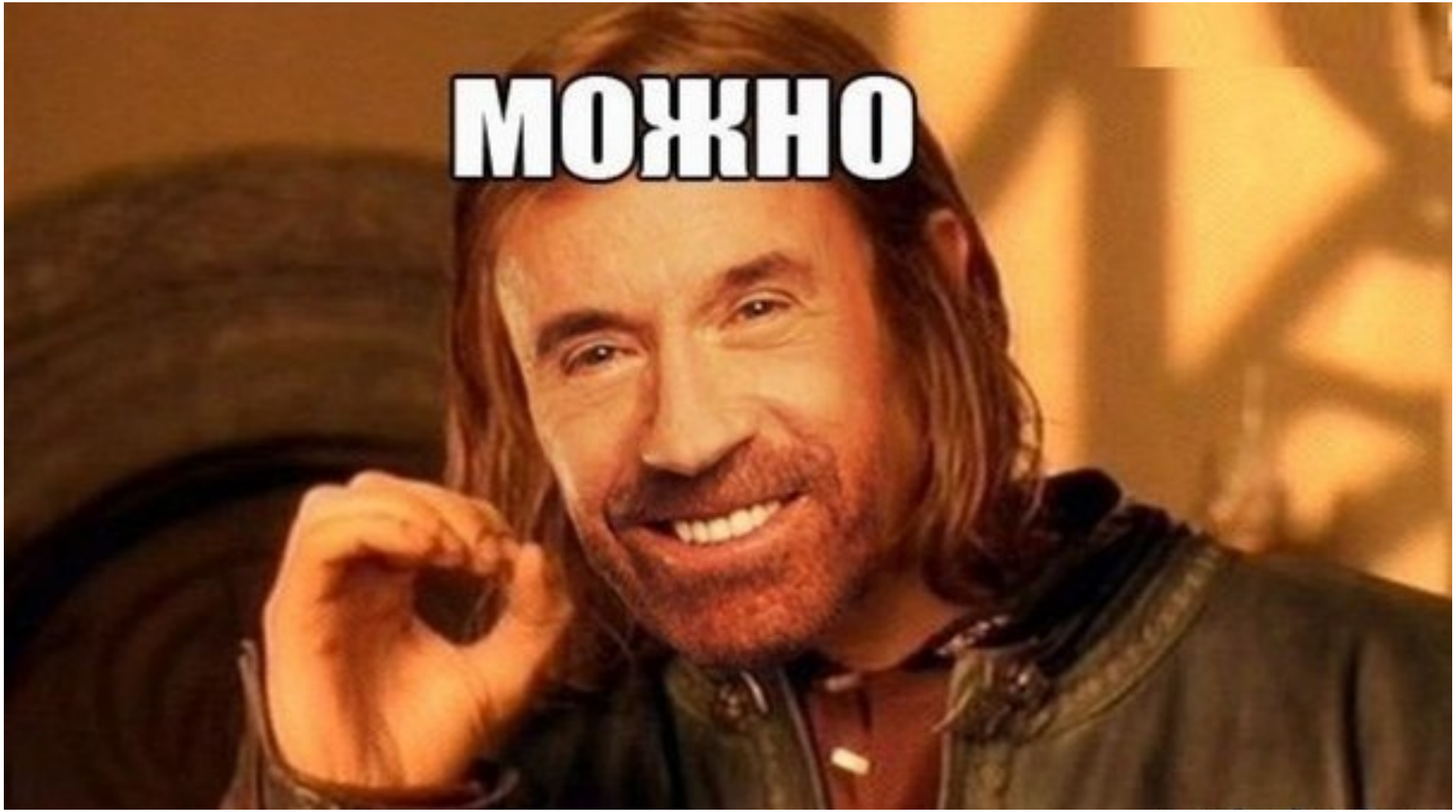
PagingAndSortingRepository

- **Наследует от CrudRepository**
- **Добавляет ещё 2 метода**

```
Iterable<T> findAll(Sort sort);

Page<T> findAll(Pageable pageable);
```

А нельзя ли без магии?



Вот все заклинания и их расшифровки

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs,findByFirstnameEquals	where x.firstname = 1?
Between	findByStartDateBetween	where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	where x.age ⇐ ?1
GreaterThan	findByAgeGreaterThan	where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	where x.age >= ?1
After	findByStartDateAfter	where x.startDate > ?1
Before	findByStartDateBefore	where x.startDate < ?1
IsNull	findByAgeIsNull	where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	where x.age not null
Like	findByFirstnameLike	where x.firstname like ?1
NotLike	findByFirstnameNotLike	where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	where x.age not in ?1
True	findByActiveTrue()	where x.active = true
False	findByActiveFalse()	where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	where UPPER(x.firstame) = UPPER(?1)

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	where x.lastname = ?1 or x.firstname = ?2
Between	findByStartDateBetween	where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	where x.age ⇐ ?1
GreaterThan	findByAgeGreaterThan	where x.age > ?1
After	findByStartDateAfter	where x.startDate > ?1
Before	findByStartDateBefore	where x.startDate < ?1

IsNull	findByAgeIsNull	where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	where x.age not null
Like	findByFirstnameLike	where x.firstname like ?1
NotLike	findByFirstnameNotLike	where x.firstname not like ?1
Containing	findByFirstnameContaining	where x.firstname like ? 1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	where x.age not in ?1
IgnoreCase	findByFirstnameIgnoreCase	where UPPER(x.firstame) = UPPER(?1)

Мминуточку,
а я если я не хочу все базовые методы?

Тогда наследуем от интерфейса Repository Он пустой.

Мы хотим независимость,
Даже от Спринга!


Ну тогда пишите свои интерфейсы и аннотируйте их @RepositoryDefinition

Давайте перейдём на MongoDB

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-mongodb</artifactId>
    <version>1.6.0.RELEASE</version>
</dependency>
```

Я всю жизнь писал только SQL
Как же мне писать под Mongo?

Как писать запросы для Mongo если я знаю только SQL?

SQL SELECT Statements	MongoDB find() Statements
SELECT * **FROM** users	**db.users.find**()
SELECT id, user_id, status **FROM** users	**db.users.find**({ }, { **user_id**: 1, **status**: 1 })
SELECT user_id, status **FROM** users	**db.users.find**({ }, { **user_id**: 1, **status**: 1, **_id**: 0 })
SELECT * **FROM** users **WHERE** status != "A"	**db.users.find**({ **status**: { **$ne**: "A" } })
SELECT * **FROM** users **WHERE** status = "A" **AND** age = 50	**db.users.find**({ **status**: "A", **age**: 50 })
SELECT * **FROM** users **WHERE** status = "A" **OR** age = 50	**db.users.find**({ **$or**: [{ **status**: "A" } , { **age**: 50 }] })
SELECT * **FROM** users **WHERE** age > 25	**db.users.find**({ **age**: { **$gt**: 25 } })
SELECT * **FROM** users **WHERE** age < 25	**db.users.find**({ **age**: { **$lt**: 25 } })
SELECT * **FROM** users **WHERE** age > 25 **AND** age <= 50	**db.users.find**({ **age**: { **$gt**: 25, **$lte**: 50 } })

А теперь с Neo4j

```
compile 'org.springframework.data:spring-data-neo4j:3.2.0.RELEASE'
```

Выводы…

Раньше

Сегодня

Всякое, полезное

- http://docs.spring.io/spring-data/jpa/docs/current/reference/html/
- http://projects.spring.io/spring-data-neo4j/
- http://docs.spring.io/spring-data/mongodb/docs/current/reference/html/
- http://docs.mongodb.org/manual/reference/sql-comparison