++ 2015 Spring Data | Евгений Борисов — Spring Data? Да, та!

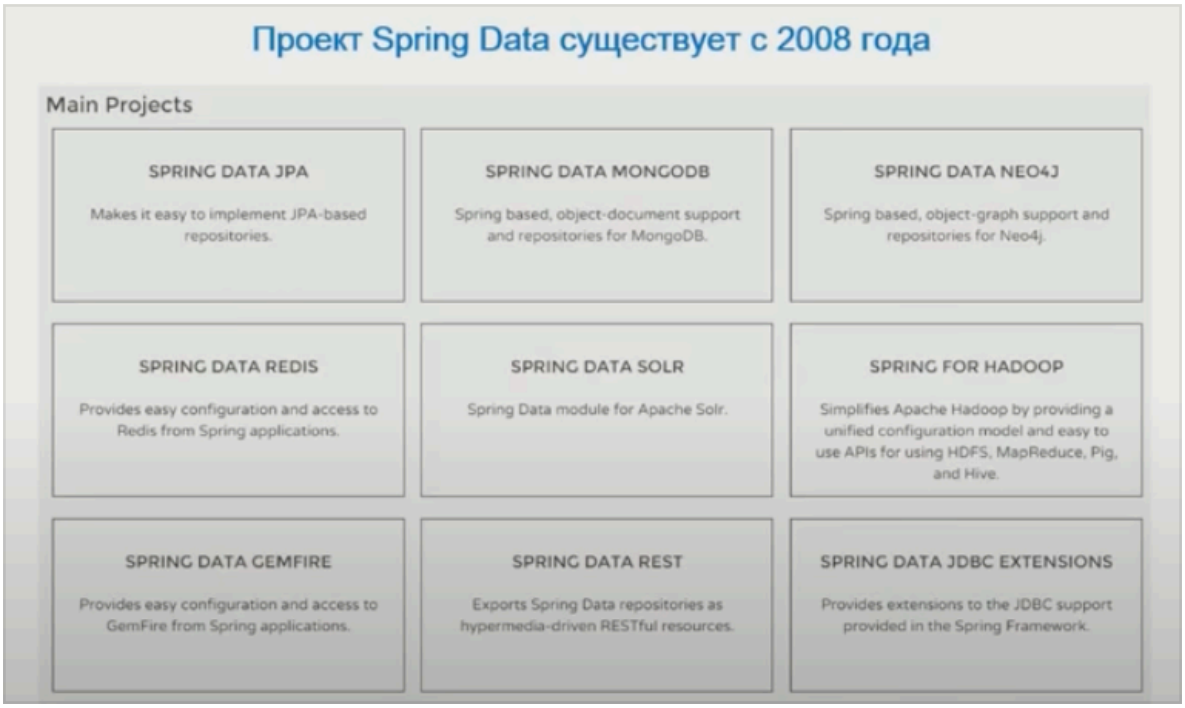
https://www.youtube.com/watch?v=nwM7A4TwU3M&ab_channel=JUG.ru

План работы с JDBC

1. Открываем Connection
2. Создаем Statement
3. Выполняем Statement
4. Проходим по ResultSet
5. Заботимся о всех Exception и Transaction
6. Закрываем Connection

(1) JPA и Hibernate может применяться только для реляционных баз данных, для нереляционных (MongoDB) графовых баз данных (Neo4j) JPA не применим.

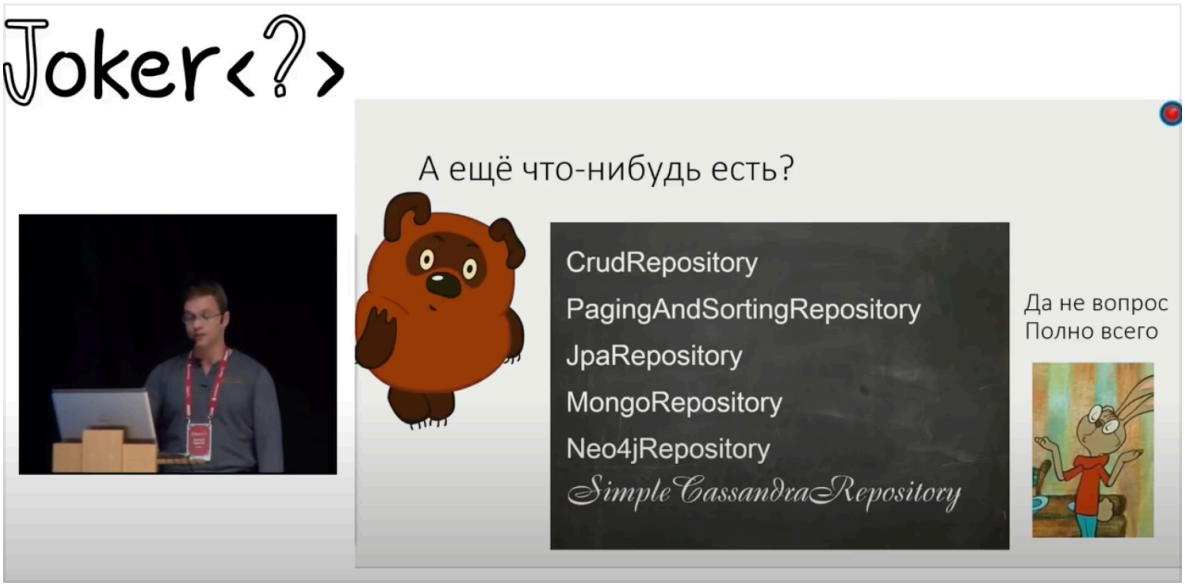
Эту проблему решает Spring Data, на изображении указаны ее модули



Разработчики Spring Data поняли, что когда обычный когда разработчик работает с Hibernate он пишет большое количество одинаковых запросов (findAll, save, findById...)

```java
@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();
}
```



Как только была указана аннотация @EnableJpaRepository Spring ищет все элементы наследующиеся от CrudRepository, создает из них объект, прописывает все методы указанные в основном интерфейсе и генерирует методы написанные руками, варианты

представлены ниже.

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)

В том случае, если все базовые методы не нужны, можно унаследоваться от интерфейса Repository, он пустой.

В том случае если нужны собственные методы в репозитории можно написать свой репозиторий, пометив его аннотацией @RepositoryDefinition.

```java
/**
 * Created by Jeka on 18/10/2014.
 */
@RepositoryDefinition(domainClass = Talk.class,idClass = Long.class)
public interface TalkRepository {
    List<Talk> findByTitleLikeIgnoreCase(String s);
}
```

Mongo

Конфигурация

```java
@Configuration
@EnableMongoRepositories
public class AppConfig {
    @Bean
    public MongoClient mongoClient() throws UnknownHostException {
        return new MongoClient();
    }

    @Bean
    public MongoTemplate mongoTemplate() throws UnknownHostException {
        return new MongoTemplate(mongoClient(),"db_mongo");
    }

}
```

MongoRepository позвоняет осуществлять поиска аналогично поиску в CrudRepository

```java
public interface SpeakerRepository extends MongoRepository<Speaker,Long> {


    Speaker findByName(String name);

    List<Speaker> findПожалуйстаByNameLike(String nameLike);

    List<Speaker> findByNameEndingWith(String suffix);

    List<Speaker> findByTalksTitleLikeIgnoreCase(String partOfTalkTitle);

    @Query(fields = "{talks.title}")
    List<Speaker> findByTalksWhenBetween(Date from, Date to);
}
```

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 } })

Neo4j

Конфигурация

```
@Configuration
@EnableNeo4jRepositories
public class AppConfig extends Neo4jConfiguration{
    public AppConfig() {
        setBasePackage("conference");
    }
    @Bean
    public GraphDatabaseService graphDatabaseService() {
        return new GraphDatabaseFactory().newEmbeddedDatabase("neo4j.db");
    }
}
```

В моделях графа важно правильно указывать связи

```
@NodeEntity
public class Speaker {
    @GraphId
    private Long speakerId;

    private String name;

    @RelatedTo(direction = Direction.BOTH)
    private Set<Talk> talks;
```