Spring Data MongoDB Webiner

Hakan Özler

ozler.hakan@gmail.com

/ozlerhakan

Kodcu.com

İçerik

- NoSQL & MongoDB
- Spring Data MongoDB (SDM) Giriş
- SDM Etkinleştirme
- SDM Konfigürasyonu
- SDM Nesne-Doküman Eşleşmesi
- SDM Sorgu Oluşturma
- SDM Mongo Depo Oluşturma
- Mongolastic

NoSQL & MongoDB

NoSQL

BigData

\+

Cassandra

mongoDB

Neo4j

NoSQL veritabanları

=

İşlenebilir veri

huMONGOus Veritabanı

- CreateReadUpdateDelete
- Yönetimsel Komutlar
- Performans
  - Storage Engine, Endekslemeler
- Deployment Seçenekleri

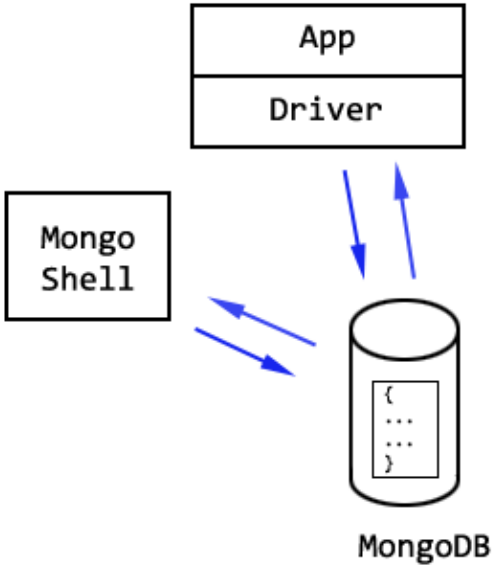
Standalone Deployment

Avantajlar:

- Sadelik
- Ucuz Maliyet

Dezavantajlar:

- Ölçekleme yok
- Süreklilik yok



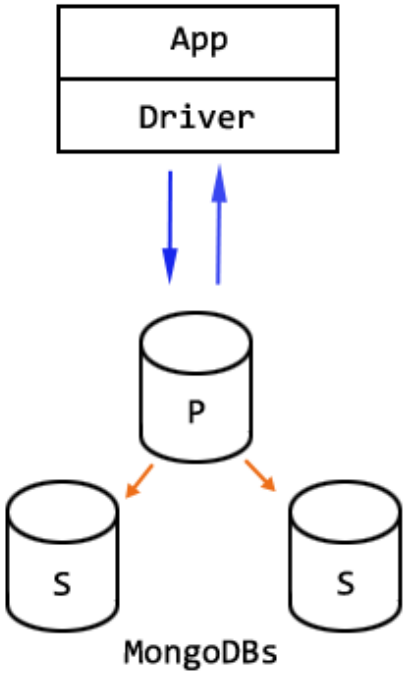
Replica Set Deployment

Avantajlar:

- Süreklilik
- Ulaşılabilirlik

Dezavantajlar:

- Ölçekleme yok
- Kompleks Yapı



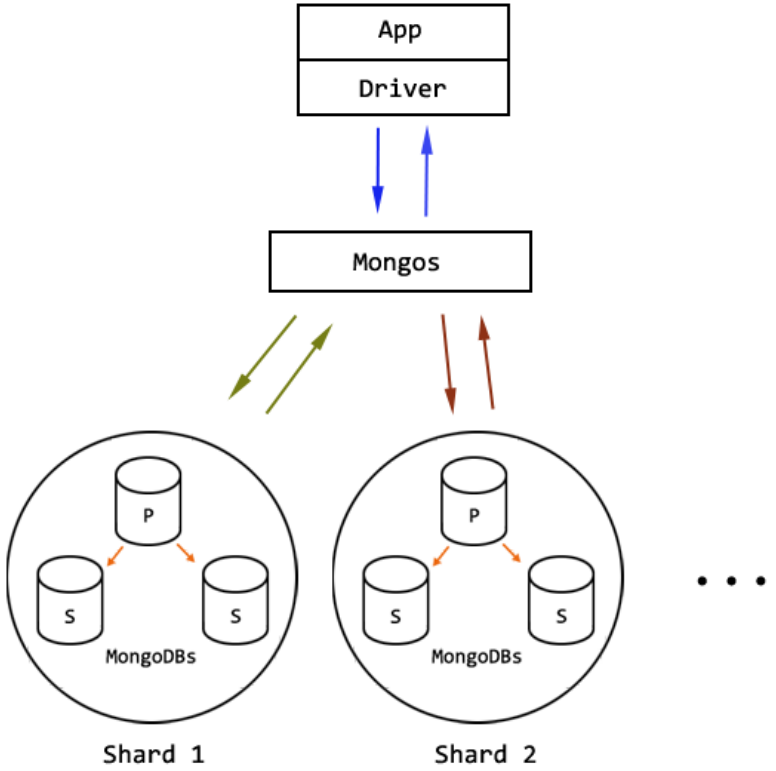
Sharded Cluster Deployment

Avantajlar:

- Ölçekleme
- Süreklilik
- Ulaşılabilirlik

Dezavantajlar:

- Maliyet
- Karmaşıklık



Spring Data MongoDB (SDM)
Giriş

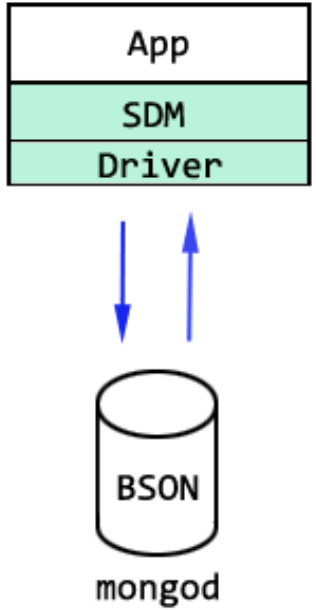
Spring Data Projesi

Giriş

- Şablon tabanlı veritabanı erişimi
- Nesne-doküman eşleşmesi
- Otomatik Mongo deposu

Giriş

- 1.7 versiyon
- >= MongoDB 2.6 Uyumluluk
- Mongo Java Driver 2/3-beta3
- JavaScript fonksiyonları yazılabilir
- Java 8 `Stream<T>` desteği



SDM Etkinleştirme

Etkinleştirme

- Xml tabanlı metadata
- Java anotasyon tabanlı metadata

Etkinleştirme

- com.mongodb.MongoClient
- org.springframework.data.mongodb.core.MongoClientFactoryBean
- org.springframework.data.document.mongodb.MongoTemplate

```
public MongoTemplate(Mongo mongo, String databaseName)
public MongoTemplate(Mongo mongo, String databaseName, UserCredentials userCredentials)
public MongoTemplate(MongoDbFactory mongoDbFactory)
public MongoTemplate(MongoDbFactory mongoDbFactory, MongoConverter mongoConverter)
```

```
public class MongoTemplate implements MongoOperations, ApplicationContextAware {
}
```

Etkinleştirme

MongoTemplate Özellikleri:

- Mongo Driver Api'den geçişi kolaylaştırma
- SDM yaklaşımının merkezi
- Thread-safe
- CRUD desteği
- MongoDB doküman – nesne eşleşmesi kullanmakta

SDM Konfigürasyonu

Java Tabanlı #1

```
@Configuration → konfigürasyon sınıfı tanımlanır
@ComponentScan(basePackages = "com.kodcu.mongodb.spring") → stereotype anotasyonlu sınıfları tara
class MongoConfigurationBeans {

    @Bean → com.mongodb.MongoClient API üretici bean
    public MongoClientFactoryBean mongo() {
        MongoClientFactoryBean mongo = new MongoClientFactoryBean();
        mongo.setHost("localhost");
        mongo.setPort(27017);
        return mongo;
    }
    @Bean → MongoTemplate üretici bean
    public MongoOperations mongoTemplate(Mongo mongo) {
        return new MongoTemplate(mongo, "twitter");
    }                    → dönüş tipine dikkat
}
```

Java Tabanlı #2

```
@ComponentScan(basePackages = "com.kodcu.mongodb.spring")
class MongoConfiguration extends AbstractMongoConfiguration {

    @Override
    protected String getDatabaseName() {
        return "twitter";
    }

    @Override
    public Mongo mongo() throws Exception {
        return new MongoClient(new ServerAddress("localhost", 27017));
    }
}

@Configuration
public abstract class AbstractMongoConfiguration {
...
}
```

mongoTemplate nesnesi için

veritabanı ismi belirleniyor

bir mongo client üretimi

tanımlı gelen konfigürasyon sınıfı

XML Tabanlı #1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mongo="http://www.springframework.org/schema/data/mongo"
       xsi:schemaLocation="http://www.springframework.org/schema/data/mongo
                           http://www.springframework.org/schema/data/mongo/spring-mongo.xsd
                           http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <mongo:mongo  id="client"
                  host="localhost"
                  port="27017"/>

    <bean id="mongoTemplate"
          class="org.springframework.data.mongodb.core.MongoTemplate">
        <constructor-arg ref="client" />
        <constructor-arg value="twitter" />
    </bean>
    <bean id="tweetDAOImpl" class="com.kodcu.mongodb.spring.TweetDAOImp">
        <constructor-arg ref="mongoTemplate"/>
    </bean>
</beans>
```

mongo namespace

MongoClient deklerasyonu

mongoTemplate bean üretimi

bağımlılık enjeksiyonu

SDM Nesne-Doküman Eşleşmesi

Nesne-Doküman Eşleşmesi

```
@Document(collection = "tweets", language = "turkish")
public class Tweet {

    @Id
    private String tid;
    private long id;
    private User user;

    @TextIndexed
    private String text;
    private TweetEntity entities;

    ...
    ... Tweet dokümanı için gerekli diğer alanlar gizlenmiştir.
    ...
}
```

tweet dokümanı

metin arama dili

_id alanı

koleksiyona bağlantı

gömülü kullanıcı nesnesi

metin arama fonksiyonuna özel

gömülü entity nesnesi

```
{
    "_id" : ...,
    "id" : ...,
    "text" : "...",
    ...
    "user" : {  },
    "entities" : {  },
    ...
    ...
}
```

Nesne-Doküman Eşleşmesi

```
class User {                                  gömülü kullanıcı nesnesi
    private Integer id;
    private String name;
    @Field("protected")
    private boolean userProtected;
                                  varsayılan alan ismi ezilmekte
    ...
    ...
}
                  gömülü entity nesnesi
class TweetEntity {
    private List<MediaEntity> media = new ArrayList<>();
    private List<URLEntity> urls = new ArrayList<>();
                                                        dizi tanımlamaları
    ...
    ...
}
```

```
{
    ...
    "user" : {
        ...,
        "screen_name" : "kodcucom",
        "protected" : false,
        "name" : "kodcu.com"
        ...
    },
    ...
    ...
}
```

```
{
    ...
    ...
    "entities" : {
        "hashtags" : [ {...} ],
        "symbols" : [ ... ],
        "user_mentions" : [ ... ],
        "urls" : [ ],
        "media" : [ ]
    },
    ...
    ...
}
```

SDM Sorgu Oluşturma

MongoOperations Arayüzü

- `org.springframework.data.mongodb.core.MongoOperations#`

```
void save (Object objectToSave);
void save(Object objectToSave, String collectionName);
void insert (Object objectToInsert);
void insert(Object objectToSave, String collectionName);
long count(Query query, Class<?> entityClass);
<O> AggregationResults<O> aggregate(Aggregation aggregation, String collectionName, Class<O> outputType);
<T> List<T> find(Query query, Class<T> entityClass);
<T> T findById(Object id, Class<T> entityClass);
<T> T findOne(Query query, Class<T> entityClass);
<T> List<T> findAllAndRemove(Query query, Class<T> entityClass);
String getCollectionName(Class<?> entityClass);
WriteResult updateFirst(Query query, Update update, Class<?> entityClass);
<T> void dropCollection(Class<T> entityClass);
<T> DBCollection createCollection(Class<T> entityClass);
<T> boolean collectionExists(Class<T> entityClass);
boolean exists(Query query, Class<?> entityClass, String collectionName);
```

Criteria Sınıfı

- Fluent API tasarımına sahip
- Sorgu oluşturmada merkezi sınıf

```
package org.springframework.data.mongodb.core.query;

public class Criteria implements CriteriaDefinition {
    ...                                  → criteria nesnesi başlangıcı
    public static Criteria where(String key) {
        return new Criteria(key);
    }
    ...
}
```

Criteria Sınıfı

- MongoDB operasyonlarının karşılığı olan metodları:

```
Criteria gte(Object o);          → $gte operatörü ile bir kriter oluşturma
Criteria is(Object o);           → $is operatörü ile bir kriter oluşturma
Criteria lt(Object o);           → $lt operatörü ile bir kriter oluşturma
Criteria gt(Object o);           → $gt operatörü ile bir kriter oluşturma
Criteria in(Collection<?> c);    → $in operatörü ile bir kriter oluşturma
Criteria exists(boolean b);      → $exists operatörü ile bir kriter oluşturma
Criteria not(Object value);      → $not operatörü ile bir kriter oluşturma
Criteria near(Point point);      → $near operatörü ile bir kriter oluşturma
```

Query Sınıfı

- Fluent API tasarımına sahip

```
package org.springframework.data.mongodb.core.query;

public class Query {
    ...
    public static Query query(CriteriaDefinition criteriaDefinition) {
        return new Query(criteriaDefinition);
    }
}
```

Query nesnesi için statik üretici metod

kriterlerin bulunduğu CriteriaDefinition nesnesi

TextQuery ve TextCriteria

```
package org.springframework.data.mongodb.core.query;

public class TextCriteria implements CriteriaDefinition {
}
```

tam metin aramanın kriterlerini oluşturur

```
package org.springframework.data.mongodb.core.query;

public class TextQuery extends Query {
}
```

tam metin arama sorgusu için gerekli sınıf

Aggregation Desteği

MongoDB kümeleme operasyonu için temsili sınıf

```
import static org.springframework.data.mongodb.core.aggregation.Aggregation.*;
                          → kümeleme işlemleri için gereken statik üretici metod
Aggregation agg = newAggregation(
    aggregationOperation1(),
    aggregationOperation2(),
    ...
    aggregationOperationN(),
);
                                                    çıktı eşleşmesi için gerekli bir sınıf
        → dönen değerler için gerekli genel konteyner
AggregationResults<OutputType> results = mongoTemplate.aggregate(agg, "koleksiyon ismi", OutputType.class);
```

sıralı listelenmiş MongoDB kümeleme operasyonları
mevcut operasyonlar: project, skip, limit, unwind, group, sort ve geoNear

Sorgu #1

```
import static org.springframework.data.mongodb.core.query.Criteria.where;
import static org.springframework.data.mongodb.core.query.Query.query;

public class TweetDAOImpl implements TweetDAO {

    private final MongoOperations mongos;

    public long countTweetsByFavoritesGreaterThanEqualTo(int favoriteCount) {
        Criteria criteria = where("favorite_count").gte(favoriteCount);
        Query query = query(criteria);
        return mongos.count(query, Tweet.class);
    }
}
```

↳ { "favorite_count" : { $gte: favoriteCount } }

↳ MongoDB persistence

Sorgu #2

```
import static org.springframework.data.mongodb.core.query.Criteria.where;
import static org.springframework.data.mongodb.core.query.Query.query;

public class TweetDAOImpl implements TweetDAO {

    private final MongoOperations mongos;

    public long countTweetsByTextSearch(String text) {
        TextCriteria criteria = TextCriteria.forDefaultLanguage()
                                            .matching(text);
        Query query = TextQuery.queryText(criteria);
        return mongos.count(query, Tweet.class);
    }
}
```

Türkçe karakterleri dikkate alan textCriteria nesnesi oluşur

"text" alanına girilen metin aranır

SDM Mongo Depo Oluşturma

MongoRepository

```
public interface TweetRepository extends MongoRepository<Tweet, String>, TweetDAO { }
public class TweetRepositoryDAO implements TweetDAO {
    private final MongoOperations mongos;
    ...
}
```

isimleri aynı olmalı

SDM deposu

özel orta TweetDAO arayüzü

özel sınıfın son eki DAO olmalı

```
public interface MongoRepository<T, ID extends Serializable> extends PagingAndSortingRepository<T, ID> { }

public interface PagingAndSortingRepository<T, ID extends Serializable> extends CrudRepository<T, ID> { }

public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> { }

public interface Repository<T, ID extends Serializable> { }
```

MongoRepository Etkinleştirme

```
@EnableMongoRepositories(basePackageClasses = TweetRepository.class, repositoryImplementationPostfix="DAO")
class MongoConfiguration extends AbstractMongoConfiguration {
    @Override
    protected String getDatabaseName() {
        return "twitter";
    }

    @Override
    public Mongo mongo() throws Exception {
        return new MongoClient(new ServerAddress("localhost", 27017));
    }
}
```

mongoRepository özelliğini etkinleştirme

paket yolu çözümlenir

özel oluşturulan depo sınıfın son eki "DAO" olanı tara

Repository Örnekleri

```
public interface TweetRepository extends MongoRepository<Tweet, String>, TweetDAO {

    @Query(value = "{'user.id_str': ?0}", count = true)
    Long countTweets(String userId);

    @Query(value = "{'user.name': ?0}", count = true)
    Long countTweetsByUserName(String userName);

    Long countTweetsByFavoritedIsFalse();
    Long countByFavoritedIsTrue();

    Stream<Tweet> findTweetsByFavorited(boolean b);
    Stream<Tweet> findByFavorited(boolean b);
    Stream<Tweet> getTweetsByFavorited(boolean b);
    Stream<Tweet> readTweetsByFavorited(boolean b);
    Stream<Tweet> streamTweetsByFavorited(boolean b);

    @Query(value = "{$text: {$search: ?0}}", count = true)
    Long countByTextSearch(String text);
    Long countAllBy(TextCriteria criteria);
}
```

depo metod için özel sorgular
Query ile yapılabilir

Özel sorgu depo metodları
ile sorgu oluşturma

hepsi aynı

full text search sorguları

Mongolastic

MongoDB koleksiyonları ElasticSearch ortamına kaydeder

https://github.com/ozlerhakan/mongolastic

Daha fazlası için

- Örnek demo projesi:
  - https://github.com/kodcu/spring-data-mongodb-webinar-project
- Spring Data MongoDB Projesi:
  - http://projects.spring.io/spring-data-mongodb/
- SDM Kaynak:
  - http://docs.spring.io/spring-data/mongodb/docs/current/reference/html/
  - http://kodcu.com/2015/05/spring-data-mongodb-ve-mongodb-java-driver-kullanarak-sorgular-olusturma/

Teşekkürler!

Hakan Özler

ozler.hakan@gmail.com

/ozlerhakan

Kodcu.com