Nama	Dwinanda Alfauzan S Raefaldhi Amartya Junior	Mata Kuliah	APPL – PR
NIM	181524006 181524026	Dosen	
Kelas	2A/D4	Tanggal	

Spring-data MongoDB

Pada spring, terapat suatu abstraksi yang dapat mempermudah pengelolaan data untuk database MongoDB yaitu Spring Data Repository. Interface utama dari abstraksi ini berada pada *Repository*. Spring menawarkan kemudahan sebagai berikut:

1. Anotasi untuk memetakan objek kepada dokumen
2. Akses database template-based dengan MongoTemplate
3. Runtime repository generation

Untuk menginterasikan spring-data dengan project, dibutuhkan bean mongoClient yang didefinisikan sebagai method dalam class beranotasi @Configuration. Sejak versi 2.1 terapat AbstractMongoClientConfiguration yang dapat digunakan untuk membuat kelas konfigurasi khusus untuk MongoDB, dua method yang wajib diimplementasi yaitu mongoClient dan getDatabaseName. mongoClient membutuhkan instance dari MongoClient yang didapat dari factory MongoClients.create. Untuk konfigurasi sederhana dan tanpa memperhatikan cluster, method tersebut dapat diisi oleh ConnectionString yang mengandung informasi tentang protokol yang digunakan, hostname, dsb.

```
mongodb://root:example@mongo/?authSource=admin
```

Suatu dokumen MongoDB dapat direpresentasikan sebagai POJO hanya dengan kebutuhan untuk mendefinisikan field yang akan digunakan sebagai _id bernotasi @Id. Anotasi lain terdapat pada tabel berikut.

Annotation	Description
`@Document`	Identifies a domain object to be mapped to a MongoDB document
`@Id`	Indicates that a field is the ID field
`@DbRef`	Indicates that a field is intended to reference another document, possibly in another database
`@Field`	Defines custom metadata for a document field
`@Version`	Identifies a property to be used as a version field

Nama	Dwinanda Alfauzan S Raefaldhi Amartya Junior	Mata Kuliah	APPL – PR
NIM	181524006 181524026	Dosen	
Kelas	2A/D4	Tanggal	

Untuk mendefnisikan query, digunakan suatu interface inheritance dari MongoRepository. Method dalam interface tersebut tidak perlu memiliki interface karena Spring akan membuatkan implementasi dari setiap method yang mengacu pada return type dan signature dari setiap method. Oleh karena itu, nama dari setiap method yang didefinisikan dalam interface tersebut harus mengandung keyword sebagaimana dicontohkan dalam tabel dibawah ini.

Keyword	Sample	Logical result
After	findByBirthdateAfter(Date date)	{"birthdate" : {"$gt" : date}}
GreaterThan	findByAgeGreaterThan(int age)	{"age" : {"$gt" : age}}
GreaterThanEqual	findByAgeGreaterThanEqual(int age)	{"age" : {"$gte" : age}}
Before	findByBirthdateBefore(Date date)	{"birthdate" : {"$lt" : date}}
LessThan	findByAgeLessThan(int age)	{"age" : {"$lt" : age}}
LessThanEqual	findByAgeLessThanEqual(int age)	{"age" : {"$lte" : age}}
Between	findByAgeBetween(int from, int to) findByAgeBetween(Range<Integer> range)	{"age" : {"$gt" : from, "$lt" : to}} lower / upper bounds ($gt / $gte & $lt / $lte) according to Range
In	findByAgeIn(Collection ages)	{"age" : {"$in" : [ages…]}}
NotIn	findByAgeNotIn(Collection ages)	{"age" : {"$nin" : [ages…]}}
IsNotNull, NotNull	findByFirstnameNotNull()	{"firstname" : {"$ne" : null}}
IsNull, Null	findByFirstnameNull()	{"firstname" : null}
Like, StartingWith, EndingWith	findByFirstnameLike(String name)	{"firstname" : name} (name as regex)
NotLike, IsNotLike	findByFirstnameNotLike(String name)	{"firstname" : { "$not" : name }} (name as regex
Containing on String	findByFirstnameContaining(String name)	{"firstname" : name} (name as regex)
NotContaining on String	findByFirstnameNotContaining(String name)	{"firstname" : { "$not" : name}} (name as regex)
Containing on Collection	findByAddressesContaining(Address address)	{"addresses" : { "$in" : address}}
NotContaining on Collection	findByAddressesNotContaining(Address address)	{"addresses" : { "$not" : { "$in" : address}}}
Regex	findByFirstnameRegex(String firstname)	{"firstname" : {"$regex" : firstname }}

Nama	Dwinanda Alfauzan S Raefaldhi Amartya Junior	Mata Kuliah	APPL – PR
NIM	181524006 181524026	Dosen	
Kelas	2A/D4	Tanggal	

(No keyword)	findByFirstname(String name)	{"firstname" : name}
Not	findByFirstnameNot(String name)	{"firstname" : {"$ne" : name}}
Near	findByLocationNear(Point point)	{"location" : {"$near" : [x,y]}}
Near	findByLocationNear(Point point, Distance max)	{"location" : {"$near" : [x,y], "$maxDistance" : max}}
Near	findByLocationNear(Point point, Distance min, Distance max)	{"location" : {"$near" : [x,y], "$minDistance" : min, "$maxDistance" : max}}
Within	findByLocationWithin(Circle circle)	{"location" : {"$geoWithin" : {"$center" : [[x, y], distance]}}}
Within	findByLocationWithin(Box box)	{"location" : {"$geoWithin" : {"$box" : [[x1, y1], x2, y2]}}}
IsTrue, True	findByActiveIsTrue()	{"active" : true}
IsFalse, False	findByActiveIsFalse()	{"active" : false}
Exists	findByLocationExists(boolean exists)	{"location" : {"$exists" : exists }}

Spring Aspect Oriented Programming

Aspect Memoularisasi potongan potongan kode di beberapa class. Pada Spring AOP dapat dilakukan dengan class reguler (pendekatan schema-based) atau dengan menggunakan anotasi @Aspect (gaya dari @AspectJ).

Join Point Suatu titik dimana method di eksekusi, seperti setelah program dieksekusi atau saat mengatasi exception.

Advice Aksi yang diambil oleh sebuah aspek pada suatu Joint Point tertentu. Perbedaan antar advice antaralain “Around”, “Before”, “After”.

Pointcut Predikat yang sesuai dengan join point. Advice diasosiasikan dengan suatu ekspresi pointcut dan akan dijalankan pada setiap join point yang sesuai dengan pointcut.

Berikut contoh dari ekspresi pointcut:

Nama	Dwinanda Alfauzan S Raefaldhi Amartya Junior	Mata Kuliah	APPL – PR
NIM	181524006 181524026	Dosen	
Kelas	2A/D4	Tanggal	

- The execution of any public method:

```
execution(public * *(..))
```

- The execution of any method with a name that begins with `set`:

```
execution(* set*(..))
```

- The execution of any method defined by the `AccountService` interface:

```
execution(* com.xyz.service.AccountService.*(..))
```

- The execution of any method defined in the `service` package:

```
execution(* com.xyz.service.*.*(..))
```

- The execution of any method defined in the service package or one of its sub-packages:

```
execution(* com.xyz.service..*.*(..))
```

- Any join point (method execution only in Spring AOP) within the service package:

```
within(com.xyz.service.*)
```

- Any join point (method execution only in Spring AOP) within the service package or one of its sub-packages:

```
within(com.xyz.service..*)
```

- Any join point (method execution only in Spring AOP) where the proxy implements the `AccountService` interface:

```
this(com.xyz.service.AccountService)
```

Setiap pointcut dapat dikombinasikan dengan menggunakan operator && (AND) atau || (OR). Setiap advice dapat langsung mendeklarasikan method pointcut.

Perlu diperhatikan bahwa penggunaan @Aspect dalam Spring harus dibarengi dengan penggunaan @Component untuk class biasa. @Component akan membantu proses Classpath Scanning sehingga aspect dapat dikenali.