http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte3.5:psl:data:mongodb:repositories

Description

Spring Data MongoDB supports the abstract Spring Data Repository . See Repository of Spring Data JPA Guideline for more information.

1. Getting Started

To use repository for MongoDB, you need to configure repositories for Mongo Scheme as follows:

```
<?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/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
				http://www.springframework.org/schema/data/mongo http://www.springframework.org/schema/data/mongo/spring-mongo.xsd">

	<mongo:mongo host="localhost" />

	<mongo:db-factory dbname="database" mongo-ref="mongo" />

	<bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate">
		<constructor-arg name="mongoDbFactory" ref="mongoDbFactory" />
	</bean>

	<!-- for Repository -->
	<mongo:repositories base-package="egovframework.rte.psl.data.mongodb.repository" />

</beans>
```

Generating Domain Object

You can use POJO to reposit the document in MongoDB as follows:

```
public class Person {
		@Id
		private String id;

		private String firstname;

		private String lastname;

		private Address address;

		private double[] location;

		...

		// getters/setters
```

Basic Repository Interface

```
public interface PersonRepository extends MongoRepository<Person, Long> {
```

```
    // additional custom finder methods go here
}
```

Paging Operations

The MongoRepository interface extends PagingAndSortingRepository to support paging operations, as follows:

```
@Autowired
private PersonRepository repository;

...

@Test
public void readsFirstPageCorrectly() {

    Page<Person> persons = repository.findAll(new PageRequest(0, 10));

    LOGGER.info("Persons Total elements : " + persons.getTotalElements());

    assertTrue(persons.isFirst());
}
```

2. Query Methods

The keyword-based query method for Spring Data is available for Repository.

ex:

```
public interface PersonRepository extends MongoRepository<Person, String> {

    List<Person> findByLastname(String lastname);

    Page<Person> findByFirstname(String firstname, Pageable pageable);

    Person findByShippingAddresses(Address address);

}
```

The following keywords are available:

Keyword	Sample	Logic result
GreaterThan	findByAgeGreaterThan(int age)	{“age” : {“$gt” : age}}
GreaterThanEqual	findByAgeGreaterThanEqual(int age)	{“age” : {“$gte” : age}}
LessThan	findByAgeLessThan(int age)	{“age” : {“$lt” : age}}
LessThanEqual	findByAgeLessThanEqual(int age)	{“age” : {“$lte” : age}}
Between	findByAgeBetween(int from, int to)	{“age” : {“$gt” : from, “$lt” : to}}
In	findByAgeIn(Collection ages)	{“age” : {“$in” : [ages…]}}
NotIn	findByAgeNotIn(Collection ages)	“age” : {“$nin” : [ages…]}}

IsNotNull, NotNull	findByFirstnameNotNull()	{“age” : {“$ne” : null}}
IsNull, Null	findByFirstnameNull()	{“age” : null}
Like	findByFirstnameLike(String name)	{“age” : age} (age as regex)
Regex	findByFirstnameRegex(String firstname)	{“firstname” : {“$regex” : firstname }}
(No keyword)	findByFirstname(String name)	{“age” : name}
Not	findByFirstnameNot(String name)	{“age” : {“$ne” : name}}
Near	findByLocationNear(Point point)	{“location” : {“$near” : [x,y]}}
Within	findByLocationWithin(Circle circle)	{“location” : {“$within” : {“$center” : [[x, y], distance]}}}
Within	findByLocationWithin(Box box)	{“location” : {“$within” : {“$box” : [[x1, y1], x2, y2]}}}
IsTrue, True	findByActiveIsTrue()	{“active” : true}
IsFalse, False	findByActiveIsFalse()	{“active” : false}
Exists	findByLocationExists(boolean exists)	{“location” : {“$exists” : exists }}

3. Repository Delete Queries

You can delete any of the foregoing keeywords by working “delete..By” or “remove..By”.

```
public interface PersonRepository extends MongoRepository<Person, String> {
    List <Person> deleteByLastname(String lastname);

    Long deletePersonByLastname(String lastname);
}
```

4. Geo-spatial Repository Queries

You can process the geo-spatial repository queries by using near keyword(s), as follows:

```
public interface PersonRepository extends MongoRepository<Person, String>

    // { 'location' : { '$near' : [point.x, point.y], '$maxDistance' : distance}}
    List<Person> findByLocationNear(Point location, Distance distance);
}
```

5. MongoDB JSON-based Query Methods

With org.springframework.data.mongodb.repository.Query anntation, you can manually assign the queries using JSON qurey as follows:

```
public interface PersonRepository extends MongoRepository<Person, String>

  @Query("{ 'firstname' : ?0 }")
  List<Person> findByThePersonsFirstname(String firstname);

}
```

http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte3:fdl:server_security:xmlschema