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

Description

Being the sub-project of Spring Data, Spring Data MongoDB offers Data Access for Mongo DB compatible with document-oriented storage.

Notable MongoDB support Functions

- Offers Spring configuration (@Configuration, XML namespace)
- Offers MongoTemplate helper that helps basic processing.
- Offers Data Access Exception hierarchy conversion equivalent to what Spring offers.
- Offers Feature Rich Object Mapping integrated with the Conversion Service.
- Annotation-based mapping metadata.
- Java-based Query, Criteria, Update DSLs
- Offers Repository Inface of Spring Data
- QueryDSL, etc.

1. Getting Started

To make Spring Data MongoDB available, you'll need to add dependency as follows.

```
<dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-mongodb</artifactId>
        <version>1.5.1.RELEASE</version>
</dependency>
```

2. Configuration Related to Spring

Generating Mongo Instances

In Spring, you'll need to generate instances regarding Mongo 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:context="http://www.springframework.org/schema/context"
        xmlns:mongo="http://www.springframework.org/schema/data/mongo"
        xsi:schemaLocation=
        "http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.2.xsd
        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-3.2.xsd">

    <!-- Default bean name is 'mongo' -->
    <mongo:mongo host="localhost" port="27017"/>

</beans>
```

If MongoDB has been configured for replica, you can configure replica-set as follows:

```
<mongo:mongo id="replicaSetMongo" replica-set="127.0.0.1:27017,localhost:27018"/>
```

Generating MongoDBFactory

Next is MongoDBFactory that allows you to establish connection with Mongo instances. Configuration in XML environment is as follows:

```
<!-- Default bean name is 'mongoDbFactory' -->
<mongo:db-factory dbname="database" mongo-ref="mongo" />
```

Generating MongoTemplate

MongoTemplate is required for operations related to MongoDB.

Refer to the following generators that allows generation of MongoTemplate:

- MongoTemplate(Mongo mongo, String databaseName)
- MongoTemplate(Mongo mongo, String databaseName, UserCredentials userCredentials) : Adds user credentials
- MongoTemplate(MongoDbFactory mongoDbFactory) : Establishes connection via MogoDbFactory(com.mongodb.Mongo object, database name and login credentials)
- MongoTemplate(MongoDbFactory mongoDbFactory, MongoConverter mongoConverter) : Uses MongoConverter for mapping

In XML configuration:

```
<mongo:mongo host="localhost" port="27017"/>

<bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate">
      <constructor-arg ref="mongo"/>
      <constructor-arg name="databaseName" value="geospatial"/>
</bean>
```

In MongoDbFactory:

```
<!-- Default bean name is 'mongo' -->
<mongo:mongo host="localhost" port="27017" />

<!-- for Replica Sets -->
<!-- mongo:mongo id="replicaSetMongo" replica-set="127.0.0.1:27017,127.0.0.1:27018" /-->

<!-- Default bean name is 'mongoDbFactory' -->
<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>
```

Basic Operations

MongoTemplate allows you to save, edit, delete and map the objects.

Refer to the following example that assumes the object Person:

Class Person

```
public class Person {

   private String id;
   private String name;
   private int age;
```

```
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "Person [id=" + id + ", name=" + name + ", age=" + age + "]";
    }

}
```

Class Test

```
@Resource(name="mongoTemplate")
private    MongoTemplate mongoTemplate;

@Test
public void testBasicOperations() {

            Person person = new Person("Joe", 34);

            // Insert
            mongoTemplate.insert(person);
            LOGGER.info("Insert : " + person);

            // Find
            person = mongoTemplate.findOne(new Query(where("name").is("Joe")), Person.class);
            LOGGER.info("Found: " + person);

            assertEquals("Joe", person.getName());

            // Update
            mongoTemplate.updateFirst(query(where("name").is("Joe")), update("age", 35), Person.class);
            person = mongoTemplate.findOne(query(where("name").is("Joe")), Person.class);
            LOGGER.info("Updated: " + person);

            assertEquals(35, person.getAge());

            // Delete
            mongoTemplate.remove(person);

            // Check that deletion worked
            List<Person> people =    mongoTemplate.findAll(Person.class);
            LOGGER.info("Number of people = : " + people.size());
            assertEquals(0, people.size());

            mongoTemplate.dropCollection("person");

}
```

※ String and ObjectId are auto-converted by MongoConverter.

_id Field Operations

In MongoDB, documents must include the field '_id' or else the auto-generated ObjectId. When MongoMappingConverter is used, you can process the mapping for the field '_id'as follows:

- The designated property of @Id(org.springframework.data.annotation.Id) is mapped to '_id'.
- the attribute id is mapped to '_id'. (Refer to the foregoing example)

Document Reposition (save, insert)

A number of methods for reposition of objects are available in MongoTemplate.

One of the most simplest means of reposition is POJO, in which case the name of collection may or may not be designated when the reposition method is called, when the name of class is designated (not fully qualified).

Refer to the following repository methods for more information:

- void save(Object objectToSave) : Reposits in the default collection
- void save(Object objectToSave, String collectionName) : Reposits in the designated collection

- void insert(Object objectToSave) : Registers to the default collection
- void insert(Object objectToSave, String collectionName) : Registered to the designated collection

※ You can either 'save' or 'insert' for reposition in MongoTemplate. When you have no document previously registered, you are to 'insert' no matter what you choose to reposit. The file is overwritten when file exists. When you work 'insert' when an account ID exists, the error is thrown.

Document Update

You can either work the method updateFirst or updateMulti update the first document only or multiple documents, respectively.

Refer to the following example for how to add KRW 50 to SAVINGS accounts:

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

                ...

   WriteResult wr = mongoTemplate.updateMulti(new
Query(where("accounts.accountType").is(Account.Type.SAVINGS)),
                                                                    new Update().inc("accounts.$.balance",
50.00),
                                                                    Account.class);
```

Update Class Reposition Method

- Update addToSet (String key, Object value) Update using the $addToSet update modifier
- Update inc (String key, Number inc) Update using the $inc update modifier
- Update pop (String key, Update.Position pos) Update using the $pop update modifier
- Update pull (String key, Object value) Update using the $pull update modifier
- Update pullAll (String key, Object[] values) Update using the $pullAll update modifier
- Update push (String key, Object value) Update using the $push update modifier
- Update pushAll (String key, Object[] values) Update using the $pushAll update modifier
- Update rename (String oldName, String newName) Update using the $rename update modifier
- Update set (String key, Object value) Update using the $set update modifier

- Update unset (String key) Update using the $unset update modifier

Reference

See MongoDB Manual for MongoDB and Reference of Spring Data MongoDB for how to interwork with Spring.