Spring.repository

InformAtelier

Relational JPA

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity // This tells Hibernate to make a table out of this class
public class User {
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private Integer id;
...
```

Repository Interface

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

    <S extends T> S save(S entity);
    T findOne(ID primaryKey);
    Iterable<T> findAll();
    Long count();
    void delete(T entity);
    boolean exists(ID primaryKey);
    // … more functionality omitted.
}
```

InformAtelier

Paging Sorting Interface

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


  Iterable<T> findAll(Sort sort);
  Page<T> findAll(Pageable pageable);
}
```

Custom Repository

`src/main/java/hello/UserRepository.java`

```
package hello;

import org.springframework.data.repository.CrudRepository;

import hello.Person;

public interface PersonRepository extends CrudRepository<Person, Integer> {
}
```

Usage

```
@Controller
@RequestMapping("/users")
public class PersonController {

 @Autowired PersonRepository repository;

 @RequestMapping
 public String showUsers(Model model, Pageable pageable) {

  model.addAttribute("users", repository.findAll(pageable));
  return "users";
 }
}
```

Query Methods

```
List<Person> persons =
personRepository.findByFirstNameLike("J%");
```

InformAtelier

Custom Methods

```
interface PersonRepository extends Repository<User, Long> {

  List<Person> findByEmailAddressAndLastname(EmailAddress emailAddress, String lastname);

  // Enables the distinct flag for the query
  List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String firstname);
  List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String firstname);

  // Enabling ignoring case for an individual property
  List<Person> findByLastnameIgnoreCase(String lastname);
  // Enabling ignoring case for all suitable properties
  List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String firstname);

  // Enabling static ORDER BY for a query
  List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
  List<Person> findByLastnameOrderByFirstnameDesc(String lastname);
}
```

@Query

```
public final static String FIND_BY_ADDRESS_QUERY = "SELECT p " +
                                                   "FROM Person p LEFT JOIN p.addresses a " +
                                                   "WHERE a.address = :address";
 @Query(FIND_BY_ADDRESS_QUERY)
  public List<Person> findByAddress(@Param("address") String address);
```

Typed SQL

```
public interface QuerydslPredicateExecutor<T> {

  Optional<T> findById(Predicate predicate);

  Iterable<T> findAll(Predicate predicate);

  long count(Predicate predicate);

  boolean exists(Predicate predicate);

  // … more functionality omitted.
}
```

InformAtelier

Typed SQL Usage

```
Predicate predicate = user.firstname.equalsIgnoreCase("dave")
                                    .and(user.lastname.startsWithIgnoreCase("mathews"));

userRepository.findAll(predicate);
```

InformAtelier

Document

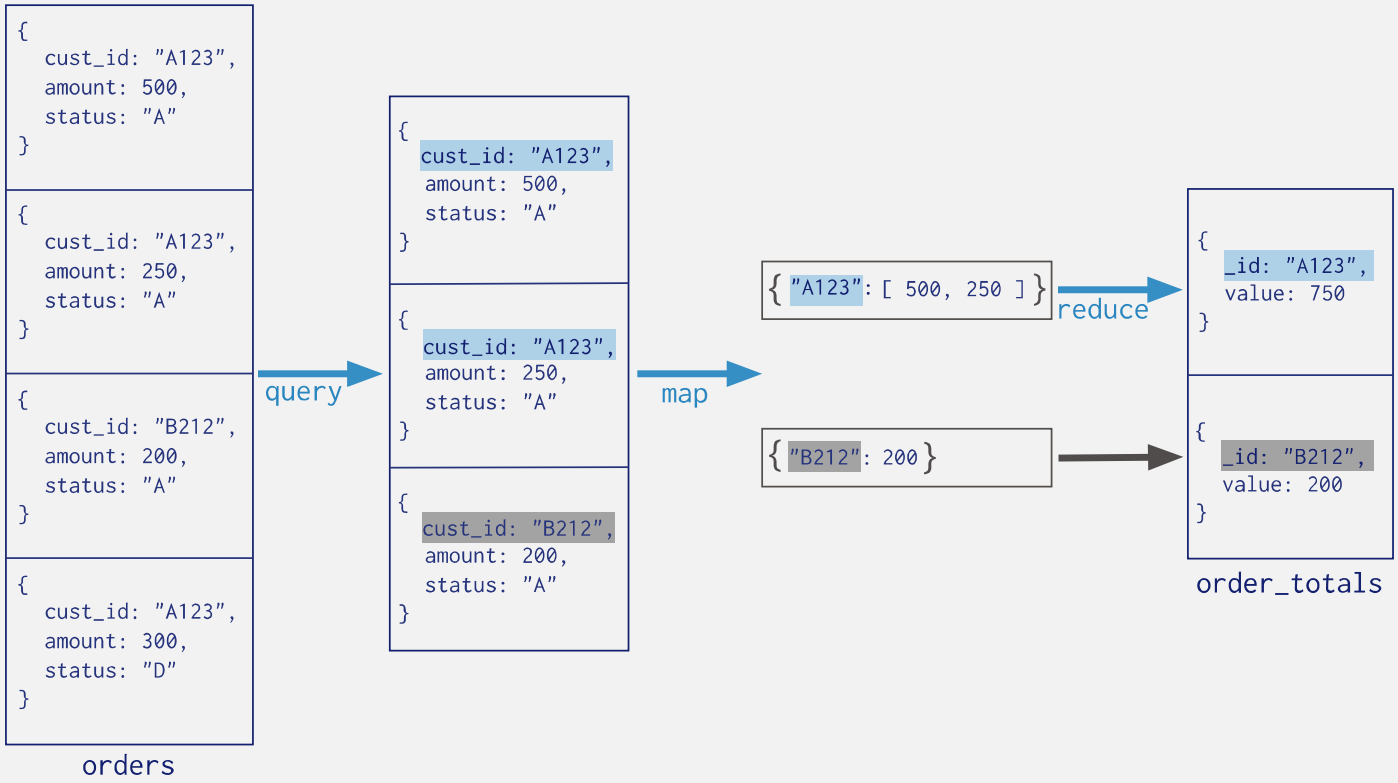
```
import static org.springframework.data.mongodb.core.query.Criteria.where;


   MongoOperations mongoOps
                        = new MongoTemplate(new MongoClient(), "database");
   mongoOps.insert(new Person("Joe", 34));


log.info(mongoOps.findOne(new Query(where("name").is("Joe")), Person.class));
mongoOps.dropCollection("person");
```

Collection

```
db.orders.mapReduce(
          map      →  function() { emit( this.cust_id, this.amount ); },
          reduce   →  function(key, values) { return Array.sum( values ) },
                      {
          query    →     query: { status: "A" },
          output   →     out: "order_totals"
                      }
                   )
```



Map Reduce

```
MapReduceResults<ValueObject> results = mongoOperations.mapReduce(
                "jmr1", "classpath:map.js", "classpath:reduce.js", ValueObject.class);


for (ValueObject valueObject : results) {
  System.out.println(valueObject);
}
```

GeoSpatial

```
Circle circle = new Circle(-73.99171, 40.738868, 0.01);

List<Venue> venues =

    template.find(new Query(Criteria.where("location").within(circle)), Venue.class);
```

Riferimenti

- https://spring.io/guides/gs/relational-data-access/
- https://spring.io/guides/gs/accessing-data-mysql/
- https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/
- https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/#repository-query-keywords
- http://www.querydsl.com/
- https://docs.mongodb.com/manual/core/map-reduce/
- https://github.com/spring-projects/spring-data-book

InformAtelier