Spring Data Redis Hashs

Spring Data Redis Hashes

Java Spring Framework를 사용한 레디스 해시(Hashes) 명령 사용법입니다.

Hashes 소스

Redis05_Hash.java

```
package com.redisgate.redis;

import lombok.extern.slf4j.Slf4j;
import org.springframework.data.redis.core.*;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.*;

@RestController
@Slf4j
public class Redis05_Hash {
   private final StringRedisTemplate stringRedisTemplate;
   private final HashOperations<String,String,String> hashOperations;

   public Redis05_Hash(StringRedisTemplate stringRedisTemplate) {
      this.stringRedisTemplate = stringRedisTemplate;
      this.hashOperations = stringRedisTemplate.opsForHash();
   }
   // 여기에 각 명령(메서드) 별 소스가 들어갑니다.
}
```

각 명령(메서드) 별 표시

HSET

```
// 예제 1) HSET: field 와 value를 저장
// HSET key field value [field value ...]
// http://localhost:8080/hset/myhash1
@GetMapping("/hset/{key}")
public String hset(@PathVariable("key") String key) {
    String msg = "예제 1) HSET(put) -> OK";
    // 하나 입력
    // void put(K key, HK hashKey, HV value)
    hashOperations.put(key, "field01", "value01");  // return void
    System.out.println(msg);

    // 맵을 구성해서 여러 개 입력
    msg = "예제 1) HSET(putAll) -> OK";
    Map<String, String> map = new HashMap<>();
    for (int i=2; i<=10; i++) {
        map.put("field"+String.format("%02d",i),"value"+String.format("%02d",i));
    }
    //     void putAll(K key, Map<? extends HK, ? extends HV> m)
    hashOperations.putAll(key, map);    // return void
    System.out.println(msg);
    return msg;
}
```

HDEL

```
// 예제 8) HDEL: field와 value를 삭제
// HDEL key field [field ...]
// http://localhost:8080/hdel/myhash1
@GetMapping("/hdel/{key}")
public String hdel(@PathVariable("key") String key) {
    String msg = "예제 8) HDEL(delete) -> ";
    long result = 0;
    // Long delete(K key, Object... hashKeys)
    result += hashOperations.delete(key,"field01");
    result += hashOperations.delete(key,"field02","field03");
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HGET

```
// 예제 2) HGET: field로 value를 조회
// HGET key field
//        http://localhost:8080/hget/myhash1:field01
@GetMapping("/hget/{key}")
public String hget(@PathVariable("key") String key) {
    String msg = "예제 2) HGET(get) -> ";
    String[] keyField = key.split(":");
    // HV get(K key, Object hashKey)
    String result = hashOperations.get(keyField[0],keyField[1]);
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HLEN

```
// 예제 4) HLEN: field의 개수를 조회
// HLEN key
// http://localhost:8080/hlen/myhash1
@GetMapping("/hlen/{key}")
public String hlen(@PathVariable("key") String key) {
    String msg = "예제 4) HLEN(size) -> ";
    Long result = hashOperations.size(key);
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HMGET

```
// 예제 3) HMGET: 여러 개의 value를 조회
// HMGET key field [field ...]
// http://localhost:8080/hmget/myhash1
@GetMapping("/hmget/{key}")
public String hmget(@PathVariable("key") String key) {
    String msg = "예제 3) HMGET(multiGet) -> ";
    // List<HV> multiGet(K key, Collection<HK> fields)
    List<String> result;
    // 입력: List 사용 -> 순서대로 나온다.
    List<String> fields1 = new ArrayList<>();
    fields1.add("field03");
    fields1.add("field04");
    fields1.add("field05");
    result = hashOperations.multiGet(key,fields1);
    System.out.println(msg+"Input List");
    result.forEach(System.out::println);

    // 입력: Set 사용 -> 순서없이 나온다.
    Set<String> fields2 = new HashSet<>();
    fields2.add("field03");
    fields2.add("field04");
    fields2.add("field05");
    System.out.println(msg+"Input Set");
    result = hashOperations.multiGet(key,fields2);
    result.forEach(System.out::println);
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HKEYS

```
// 예제 5) HKEYS: key에 속한 모든 field name을 조회
// HKEYS key
// http://localhost:8080/hkeys/myhash1
@GetMapping("/hkeys/{key}")
public String hkeys(@PathVariable("key") String key) {
    String msg = "예제 5) HKEYS(keys) -> ";
    // Set<HK> keys(K key)
    Set<String> result = hashOperations.keys(key);
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HVALS

```
// 예제 6) HVALS: key에 속한 모든 value를 조회
// HVALS key
// http://localhost:8080/hvals/myhash1
@GetMapping("/hvals/{key}")
public String hvals(@PathVariable("key") String key) {
    String msg = "예제 6) HVALS -> ";
    List<String> result = redisCommands.hvals(key);
    msg += result.toString();
    System.out.println(msg);
    return msg;
}
```

HGETALL

```
// 예제 7) HGETALL: key에 속한 모든 field와 value를 조회
// HGETALL key
// http://localhost:8080/hgetall/myhash1
@GetMapping("/hgetall/{key}")
public String hgetall(@PathVariable("key") String key) {
    String msg = "예제 7) HGETALL(entries) -> ";
    // Map<HK, HV> entries(K key)
    Map<String,String> result = hashOperations.entries(key);
    int i = 1;
    for ( Map.Entry<String,String> kv : result.entrySet()) {
        System.out.println((i++) + ") "+kv.getKey()+" "+kv.getValue());
    }
    return msg+result.toString();
}
```

HINCRBY

```
// 예제 11) HINCRBY: value를 increment 만큼 증가 또는 감소
// HINCRBY key field increment, HINCRBYFLOAT key field increment
// http://localhost:8080/hincrby/myhash1
@GetMapping("/hincrby/{key}")
public String hincrby(@PathVariable("key") String key) {
    String msg = "예제 11) HINCRBY(increment) -> ";
    Long result;
    // Long increment(K key, HK hashKey, long delta) -> 마이너스(-) 입력 가능
    result = hashOperations.increment(key,"field90",1);
    System.out.println(msg+result);
    result = hashOperations.increment(key,"field90",1);
    System.out.println(msg+result);
    result = hashOperations.increment(key,"field90",1);
    System.out.println(msg+result);
    return msg+result;
}
```

HEXISTS

```
// 예제 9) HEXISTS: field가 있는지 조회
// HEXISTS key field
// http://localhost:8080/hexists/myhash1
@GetMapping("/hexists/{key}")
public String hexists(@PathVariable("key") String key) {
    String msg = "예제 9) HEXISTS(hasKey) -> ";
    // Boolean hasKey(K key, Object hashKey)
    Boolean result = hashOperations.hasKey(key,"field01");
    System.out.println(msg+result);
    result = hashOperations.hasKey(key,"field04");
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HSETNX

```
// 예제 10) HSETNX: field가 기존에 없으면 저장
// HSETNX key field value
// http://localhost:8080/hsetnx/myhash1
@GetMapping("/hsetnx/{key}")
public String hsetnx(@PathVariable("key") String key) {
    String msg = "예제 10) HSETNX(putIfAbsent) -> ";
    // Boolean putIfAbsent(K key, HK hashKey, HV value)
    Boolean result = hashOperations.putIfAbsent(key,"field01","value01");
    System.out.println(msg+result);
    result = hashOperations.putIfAbsent(key,"field04","value04");
    msg += result;
    System.out.println(msg);
    return msg;
}
```

HSCAN

```
// 예제 12) HSCAN: field, member를 일정 단위 개수 만큼씩 조회
// HSCAN key cursor [MATCH pattern] [COUNT count]
// http://localhost:8080/hscan/myhash1
@GetMapping("/hscan/{key}")
public String hscan(@PathVariable("key") String key) {
   String msg = "예제 12) HSCAN(scan) -> ";
   // key와 pattern을 나눈다.
   String[] keyPattern = key.split(":");
   String key1 = null;
   String pattern = null;
   // ScanOptions -> COUNT(limit)와 MATCH(pattern) 설정
   ScanOptions scanOptions;
   if (keyPattern.length == 1) {   // pattern을 입력하지 않았을 경우를 고려
      key1 = keyPattern[0];
      scanOptions = ScanOptions.scanOptions().count(20).build();
   } else {
      key1 = keyPattern[0];
      pattern = keyPattern[1];
      scanOptions = ScanOptions.scanOptions().count(20).match(pattern).build();
   }
   // 테스트 데이터 입력 (키가 없을 경우에만 데이터를 입력)
   Long size = hashOperations.size(key1);
   if (size == 0) {
      for (int i = 1; i <= 100; i++) {
         String no = String.format("%02d",i);
         hashOperations.put(key1, "field"+no, "value"+no);
         hashOperations.put(key1, "user"+no, "value"+no);
      }
   }
   // HSCAN 시작
   System.out.println(msg+"0");
   // Cursor<Entry<HK, HV>> scan(K key, ScanOptions options)
   Cursor<Map.Entry<String, String>> result = hashOperations.scan(key1,scanOptions);
   long cursor = result.getCursorId();
   while (result.hasNext()) {
      if (cursor != result.getCursorId()) {
         System.out.println("Next cursor: "+result.getCursorId());
         cursor = result.getCursorId();
      }
      Map.Entry<String, String> member = result.next();
      System.out.println("  "+ result.getPosition() +") "+member.getKey()+" "+member.getValue());
   }
   return msg+result.getCursorId();
}
```

<< ZSets Hashes Streams >>

redisgate@gmail.com 02.503.2235
서울시 강남구 강남대로 342 역삼빌딩 5층 (역삼동) 우 06242