













## Performance Comparisons













### PRE GemFire 9.0.1 Testing Hash and Set testing

The following are the JUNIT test performance when ran against GemFire version 9.0.1 prior to introducing these changes.

- ▼  io.pivotal.gemfire9.HashesJUnitTest [Runner: JUnit 4] (12.549 s)
  - ✓  testHMSetHSetHLen (4.173 s)
  - ✓  testHkeys (2.117 s)
  - ✓  testHIncrBy (4.138 s)
  - ✓  testHMGetHDelHGetAllHVals (2.121 s)
- ▼  io.pivotal.gemfire9.SetsJUnitTest [Runner: JUnit 4] (31.507 s)
  - ✓  testSMove (4.160 s)
  - ✓  testSMembersIsMember (2.111 s)
  - ✓  testSDiffAndStore (8.450 s)
  - ✓  testSInterAndStore (6.281 s)
  - ✓  testSUnionAndStore (8.430 s)
  - ✓  testSAddScard (2.075 s)

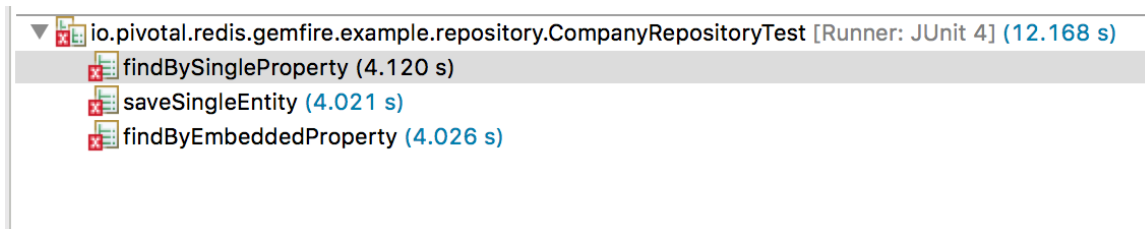
### AFTER Hash and Set Pull Request Changes

The following are the JUNIT test performance when ran against the GEODE-2469 changes.

- ▼  io.pivotal.gemfire9.HashesJUnitTest [Runner: JUnit 4] (0.022 s)
  - ✓  testHMSetHSetHLen (0.012 s)
  - ✓  testHkeys (0.002 s)
  - ✓  testHIncrBy (0.004 s)
  - ✓  testHMGetHDelHGetAllHVals (0.004 s)
- ▼  io.pivotal.gemfire9.SetsJUnitTest [Runner: JUnit 4] (0.036 s)
  - ✓  testSMove (0.015 s)
  - ✓  testSMembersIsMember (0.004 s)
  - ✓  testSDiffAndStore (0.006 s)
  - ✓  testSInterAndStore (0.004 s)
  - ✓  testSUnionAndStore (0.005 s)
  - ✓  testSAddScard (0.002 s)

## Spring Data Redis Fix

The initial Spring Data Redis test running against GemFire 9.0.1 Spring Data Redis Tests all failed.



See [https://github.com/Pivotal-Data-Engineering/gemfire9\\_examples/tree/person\\_example\\_sdg\\_Tracker139498217/redis/spring-data-redis-example/src/test/java/io/pivotal/redis/gemfire/example/repository](https://github.com/Pivotal-Data-Engineering/gemfire9_examples/tree/person_example_sdg_Tracker139498217/redis/spring-data-redis-example/src/test/java/io/pivotal/redis/gemfire/example/repository)

Exceptions related the keys having separated characters based on object:key format

See HASH section of the <https://redis.io/topics/data-types-intro> for more information on objects.

```
[Server error]
```

```
[fine 2017/02/10 16:04:33.289 EST server1 <Function  
Execution Processor2> tid=0x6a] Region names may only  
be alphanumeric and may contain hyphens or underscores:  
companies: 1000  
java.lang.IllegalArgumentException: Region names may  
only be alphanumeric and may contain hyphens or  
underscores: companies: 1000  
    at  
org.apache.geode.internal.cache.LocalRegion.validateReg  
ionName(LocalRegion.java:7618)  
    at
```

```
org.apache.geode.internal.cache.GemFireCacheImpl.create
VMRegion(GemFireCacheImpl.java:3201)
    at ...
```

While hashes are handy to represent *objects*, actually the number of fields you can put inside a hash has no practical limits (other than available memory), so you can use hashes in many different ways inside your application.

The command [HMSET](#) sets multiple fields of the hash, while [HGET](#) retrieves a single field. [HMGET](#) is similar to [HGET](#) but returns an array of values:

```
> hmget user:1000 username birthyear no-such-field 1)
```

```
"antirez" 2) "1977" 3) (nil)
```

There are commands that are able to perform operations on individual fields as well, like [HINCRBY](#):

```
> hincrby user:1000 birthyear 10 (integer) 1987 >
```

```
hincrby user:1000 birthyear 10 (integer) 1997
```

You can find the [full list of hash commands in the documentation](#).





It is worth noting that small hashes (i.e., a few elements with small values) are encoded in special way in memory that make them very memory efficient.

## Spring Data Redis Geode Fix Performance

The following is the performance from GEODE-2469

```
▼ io.pivotal.redis.gemfire.example.repository.CompanyRepositoryTest [Runner: JUnit 4] (0.222 s)
  ├── findBySingleProperty (0.142 s)
  ├── saveSingleEntity (0.022 s)
  └── findByEmbeddedProperty (0.058 s)
```

The following is how this is compared when point the Spring Data Redis code to a Redis version 3.2.7

- ▼  io.pivotal.redis.gemfire.example.repository.CompanyRepositoryTest [Runner: JUnit 4] (0.146 s)
  -  findBySingleProperty (0.113 s)
  -  saveSingleEntity (0.008 s)
  -  findByEmbeddedProperty (0.025 s)