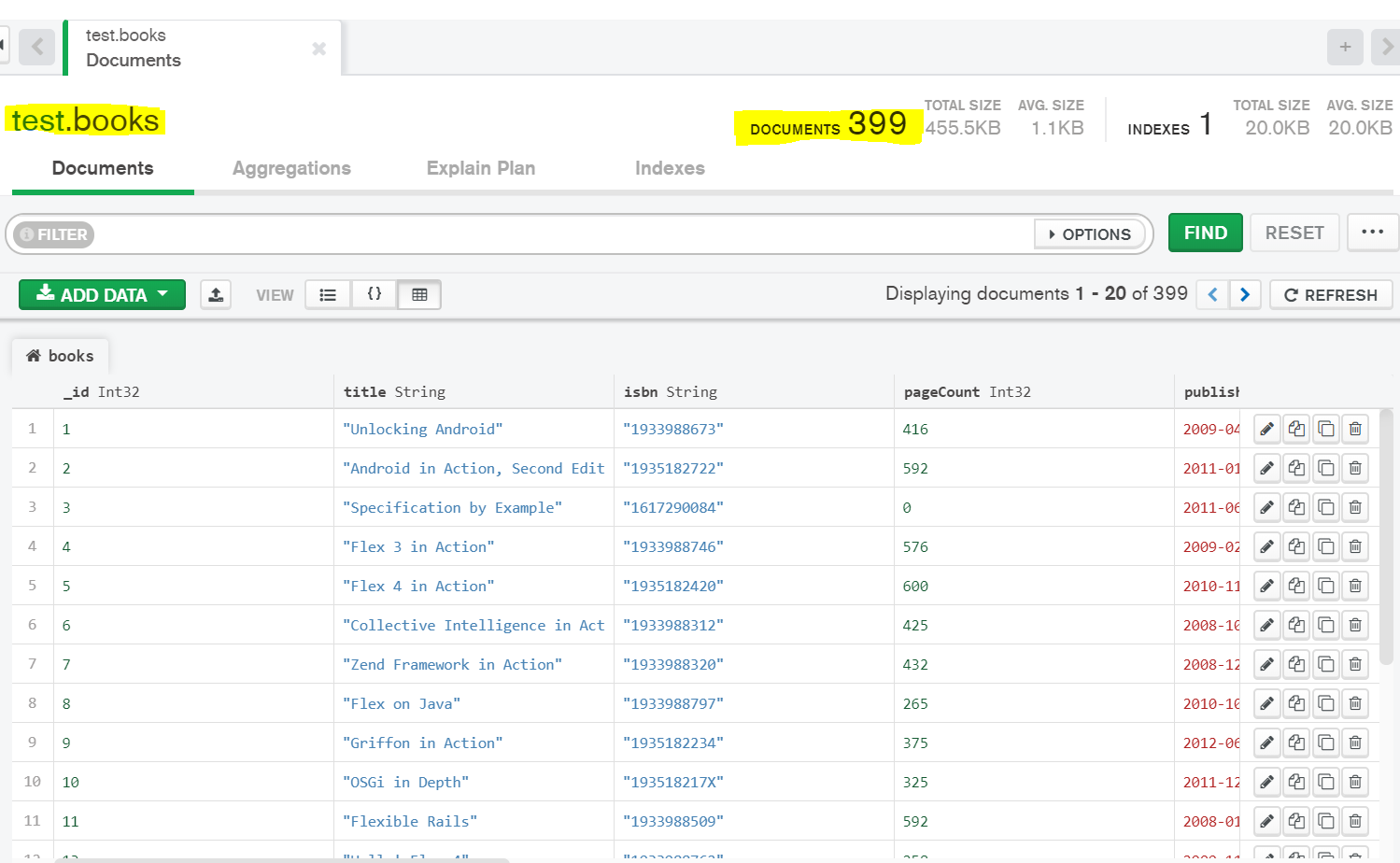
**Mongo Collection**

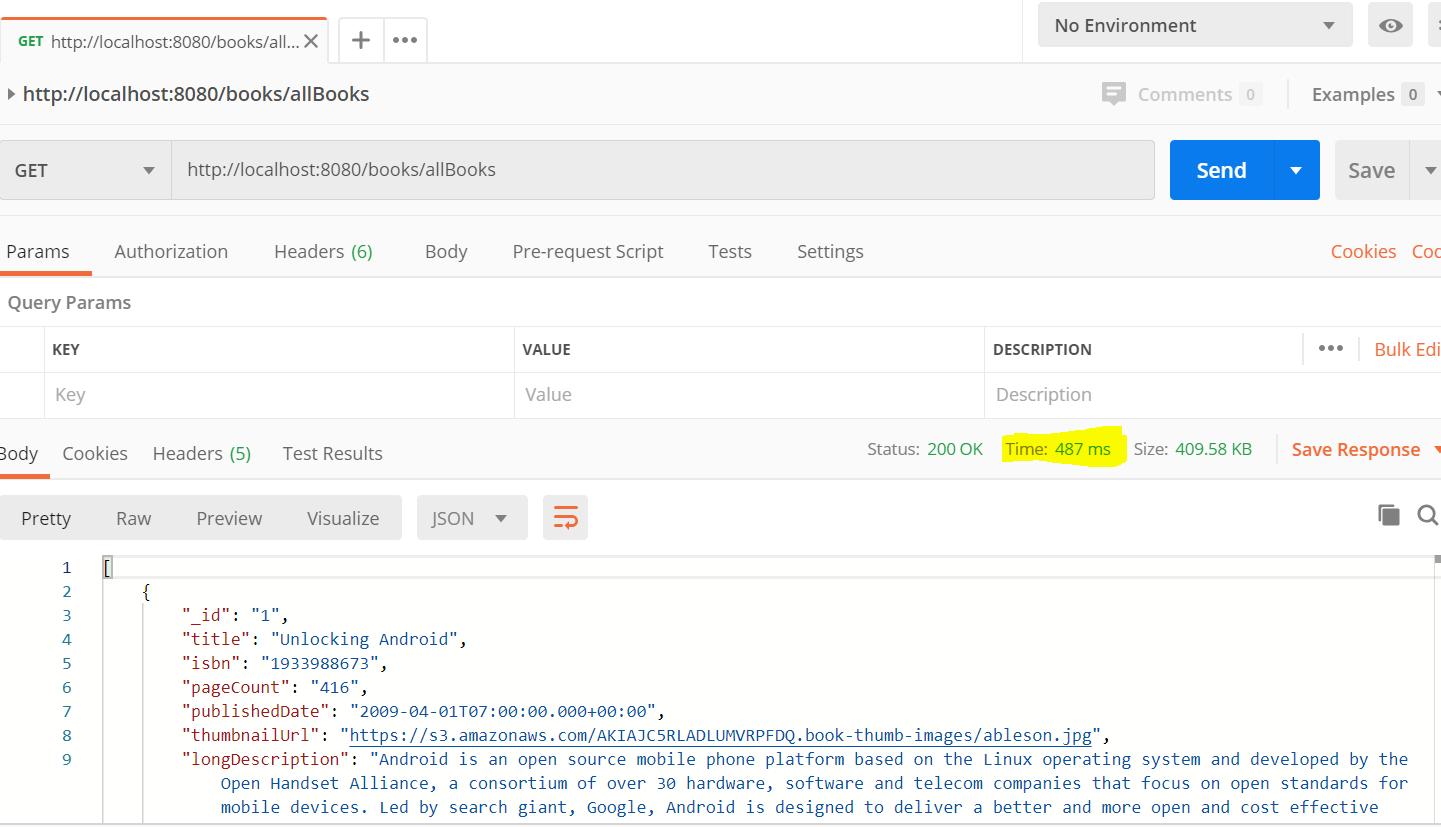
**Collection name:** books

**Total documents:** 399

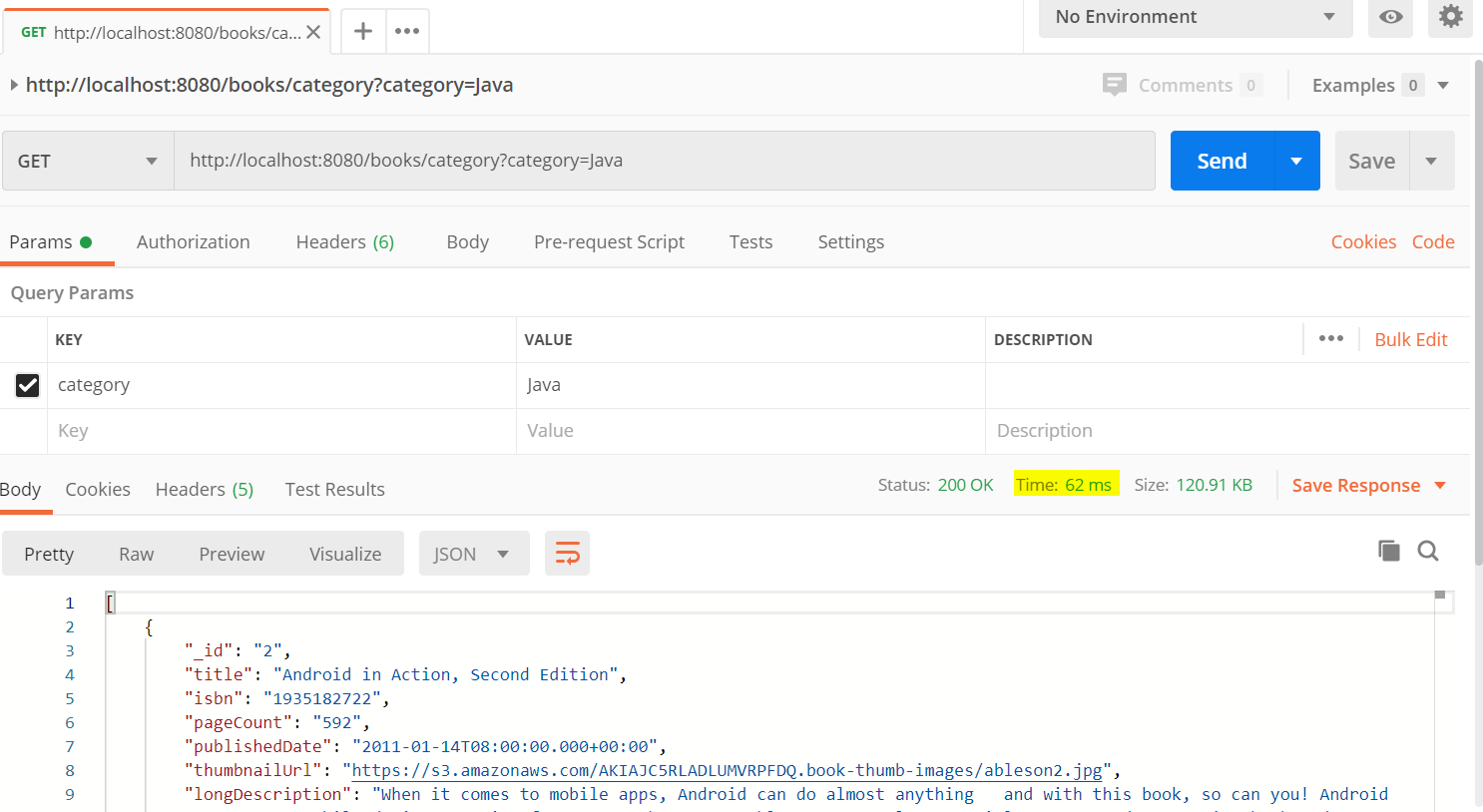


**Without Redis Cache- Response Time**

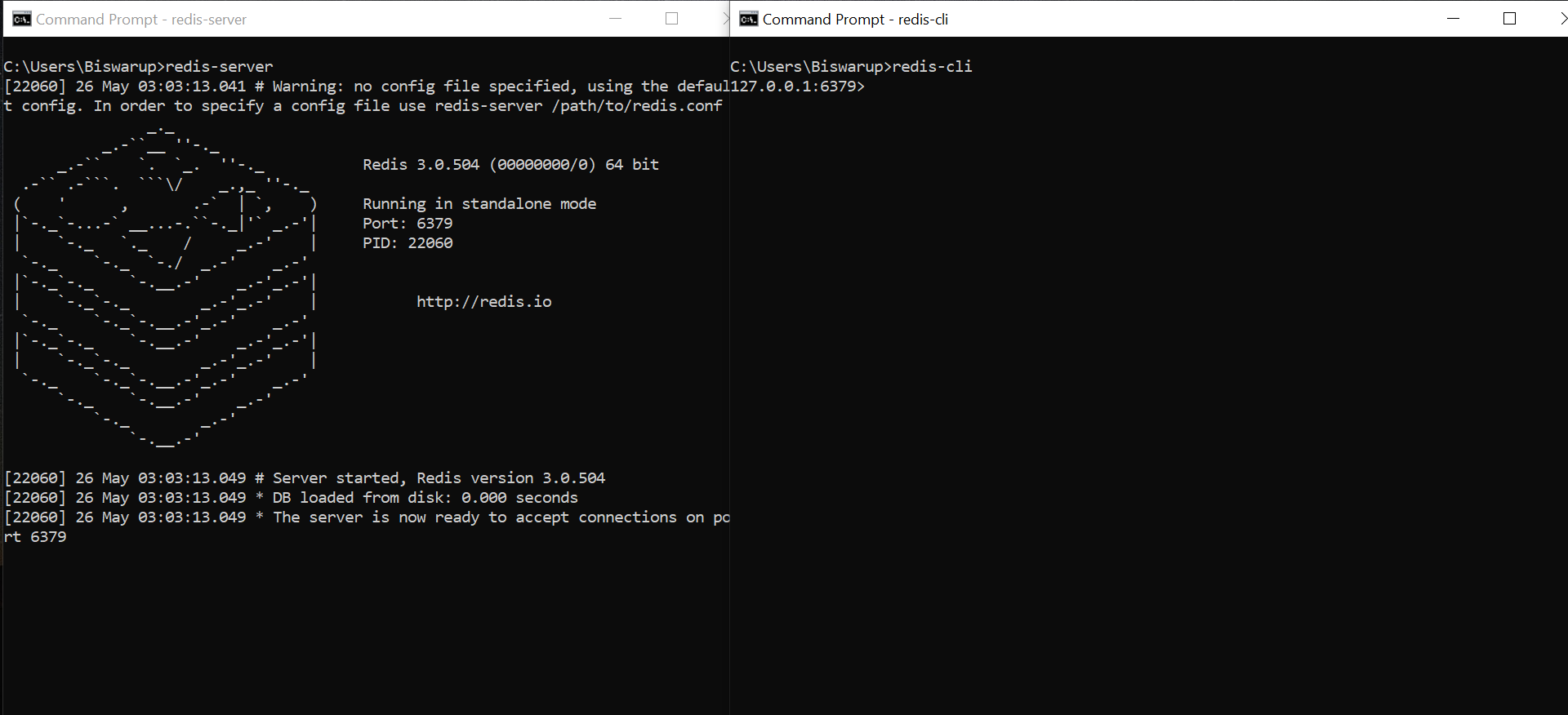
1. All Books



2. Books By category:

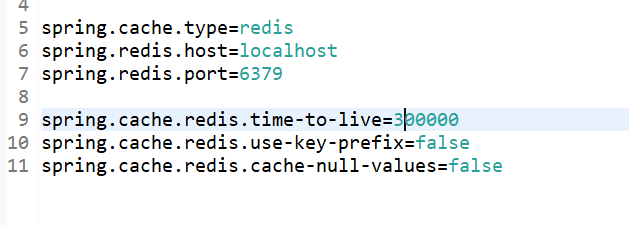


**Start Redis server and cli**



**Properties**

* Redis host details to connect to this server:
* And few other optional properties like cache ttl which is set as 5 mins here.

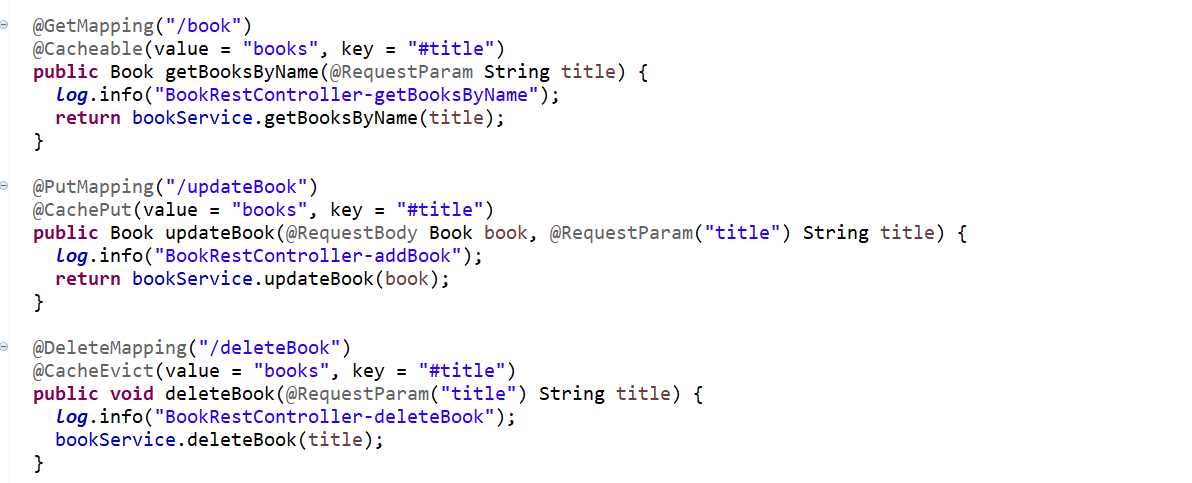


**Cacheable Methods:**

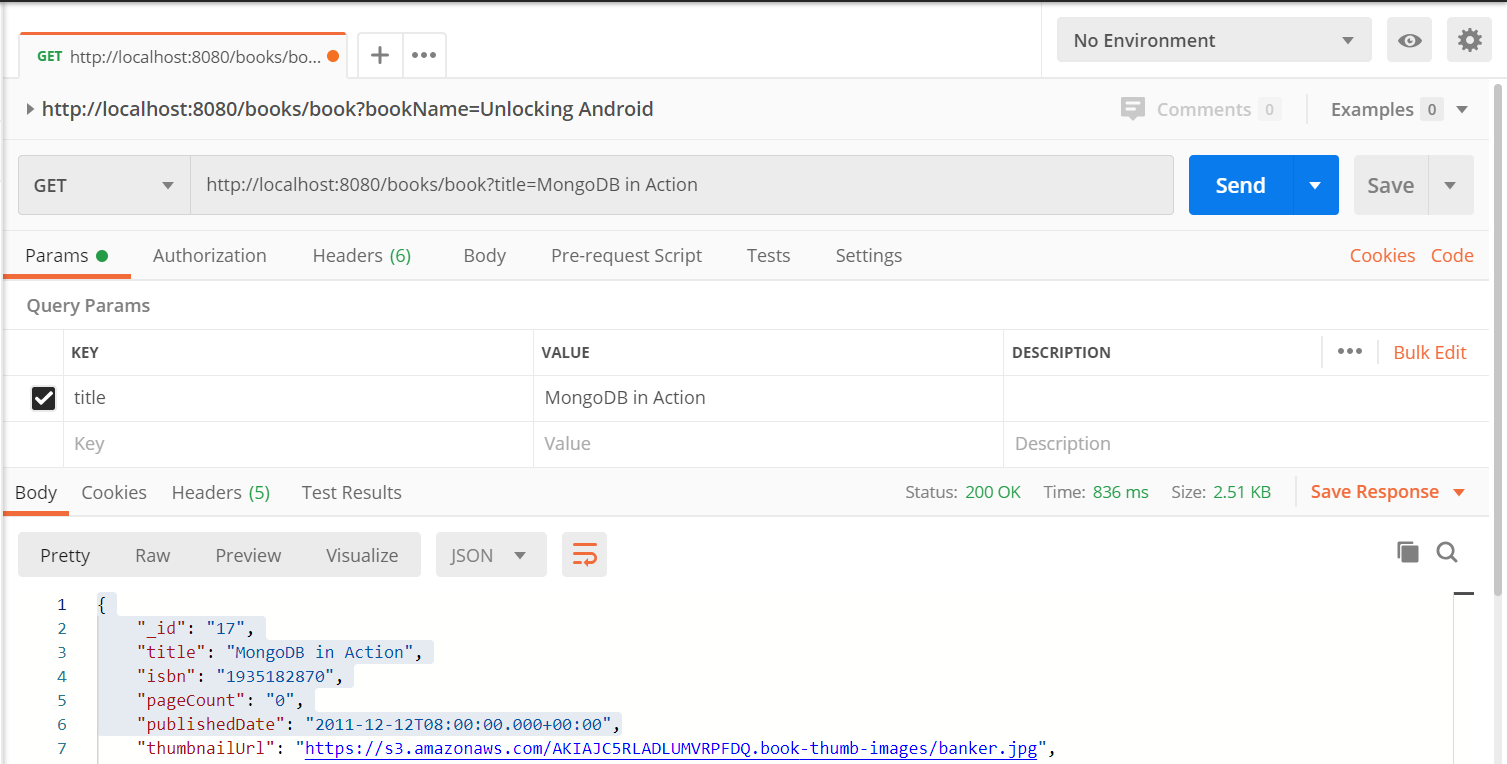
1.

* For these 3 methods Key name is the Book Title which is passed as parameter.
* If we update the content using put method, get method will get the content automatically from cache without hitting db again.

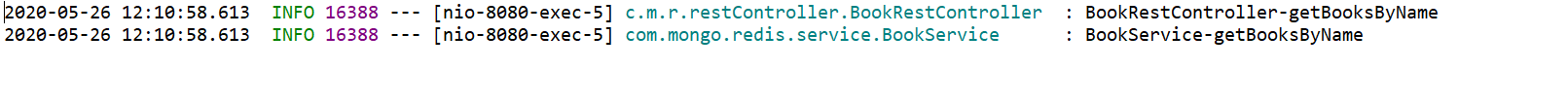
Controller:



**First time hit**:



Logs:

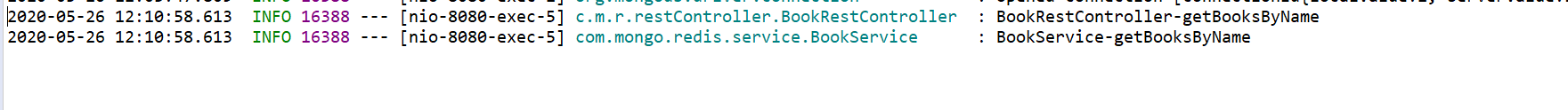


Cache:

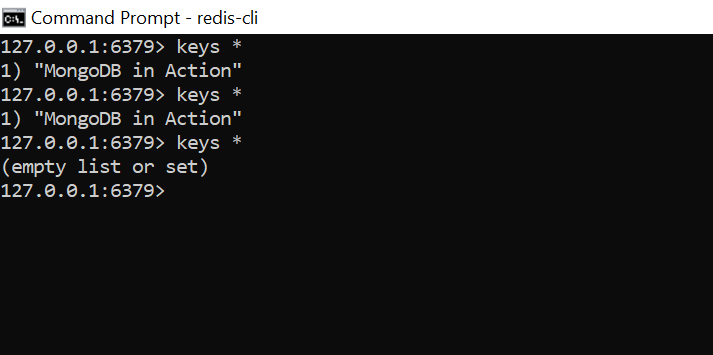


**Hit Second time:**

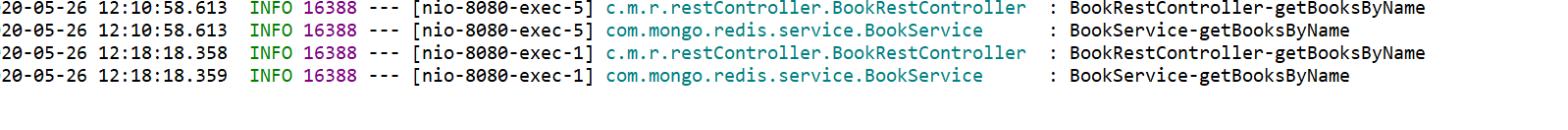
No logs from controller and service class:



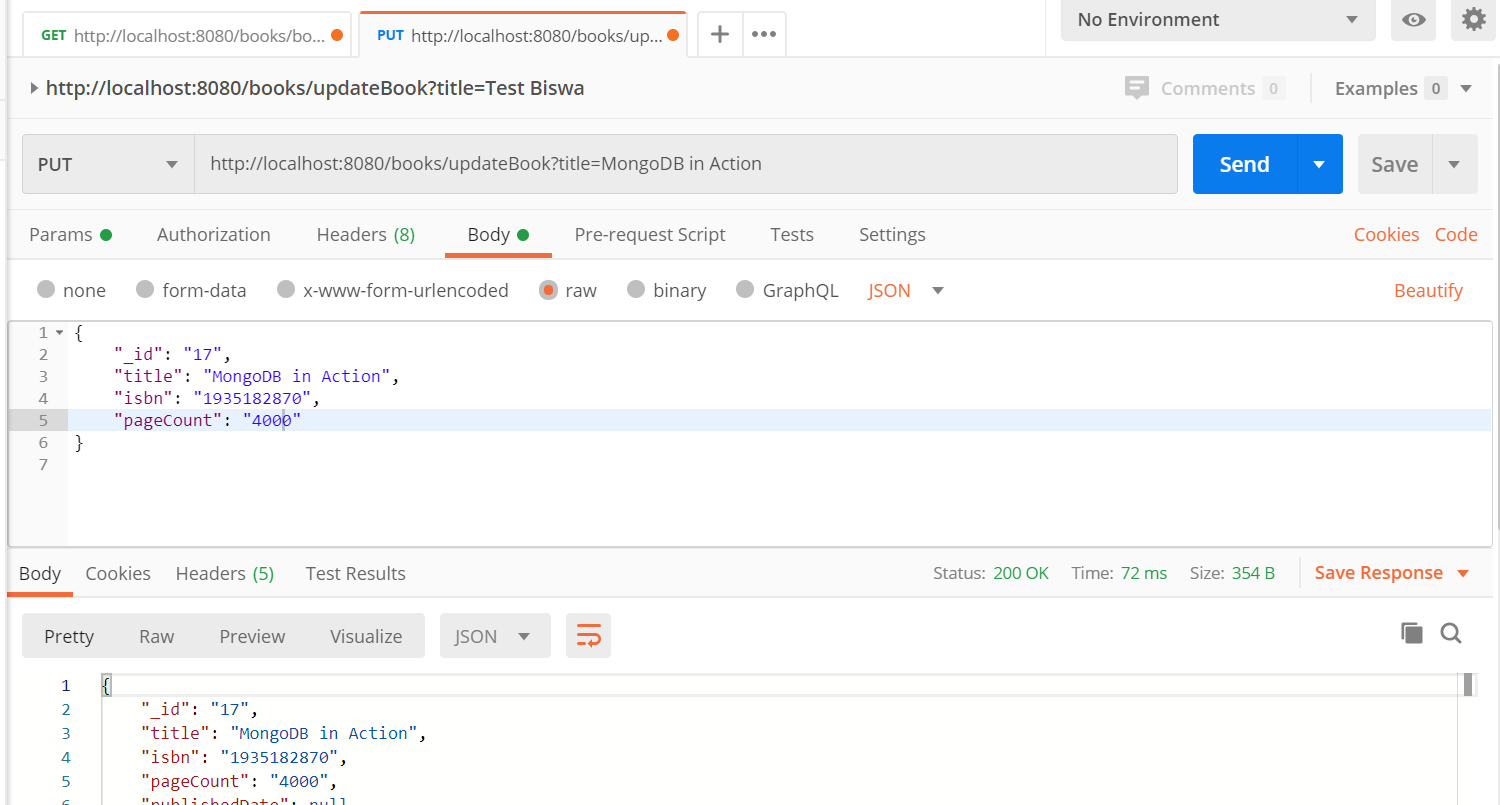
**Cache Evicted afrer 5 mins:**



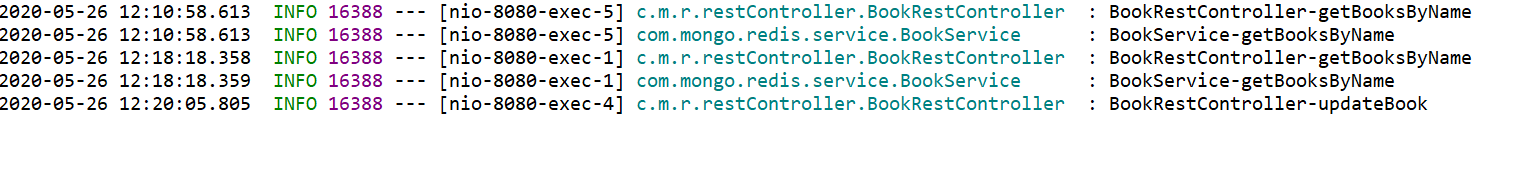
Logs: As cache is not there so it again hit the db and updated cache.



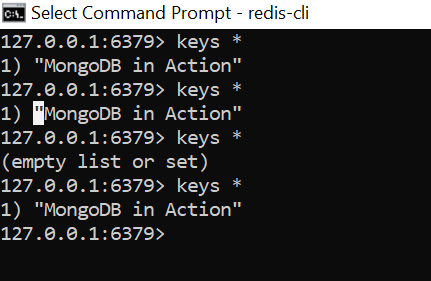
**Hit Put method to update this Book:**



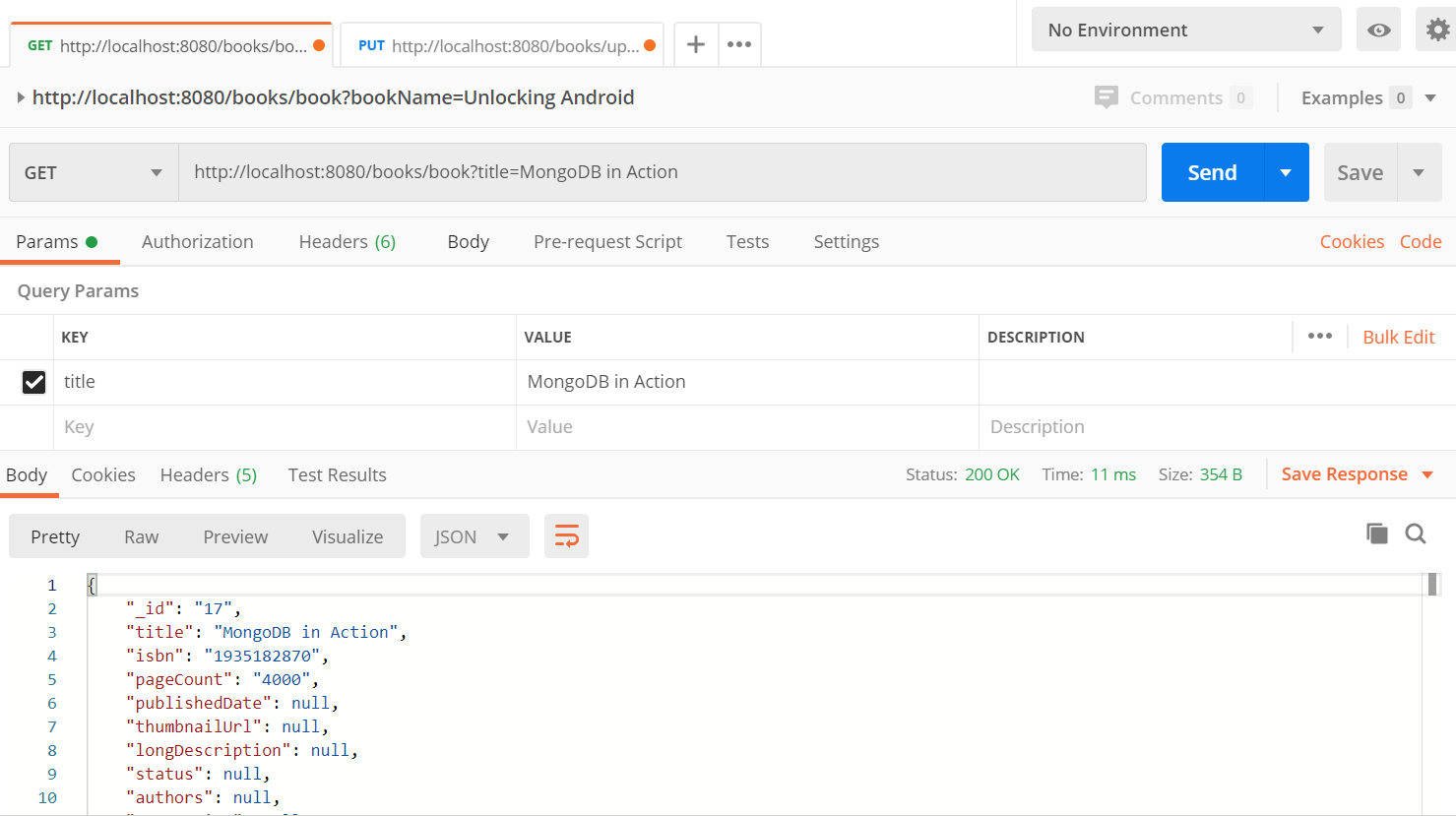
Logs:

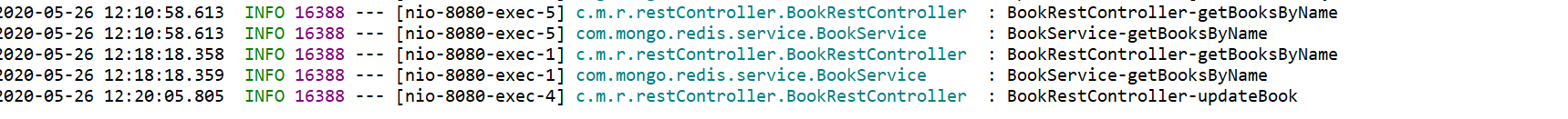


Cache Keys:



**Hit get method again which take the updated value directly from cache. Did not hit the db:**



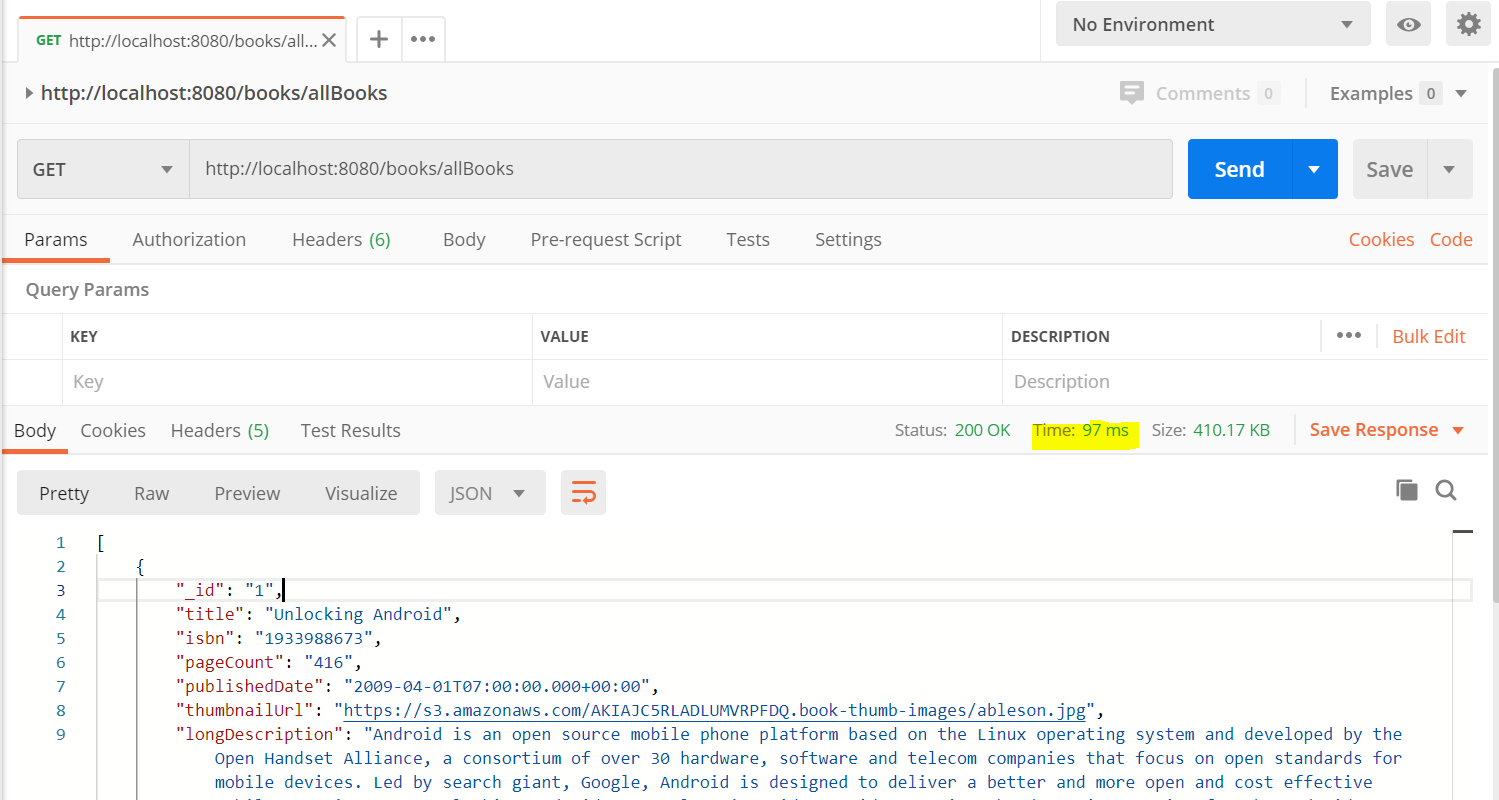


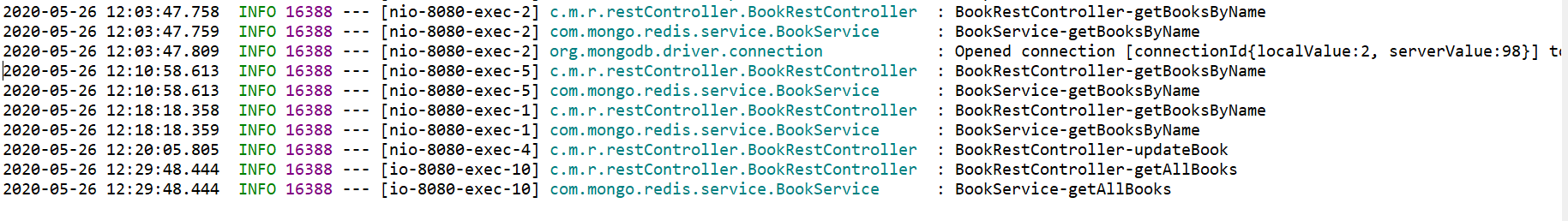
2.

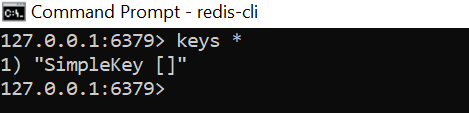
* Here the cache will be stored without any key.
* Once cache is set it will not hit the db again.
* There is no option to update this cache.
* Once the cache is expired



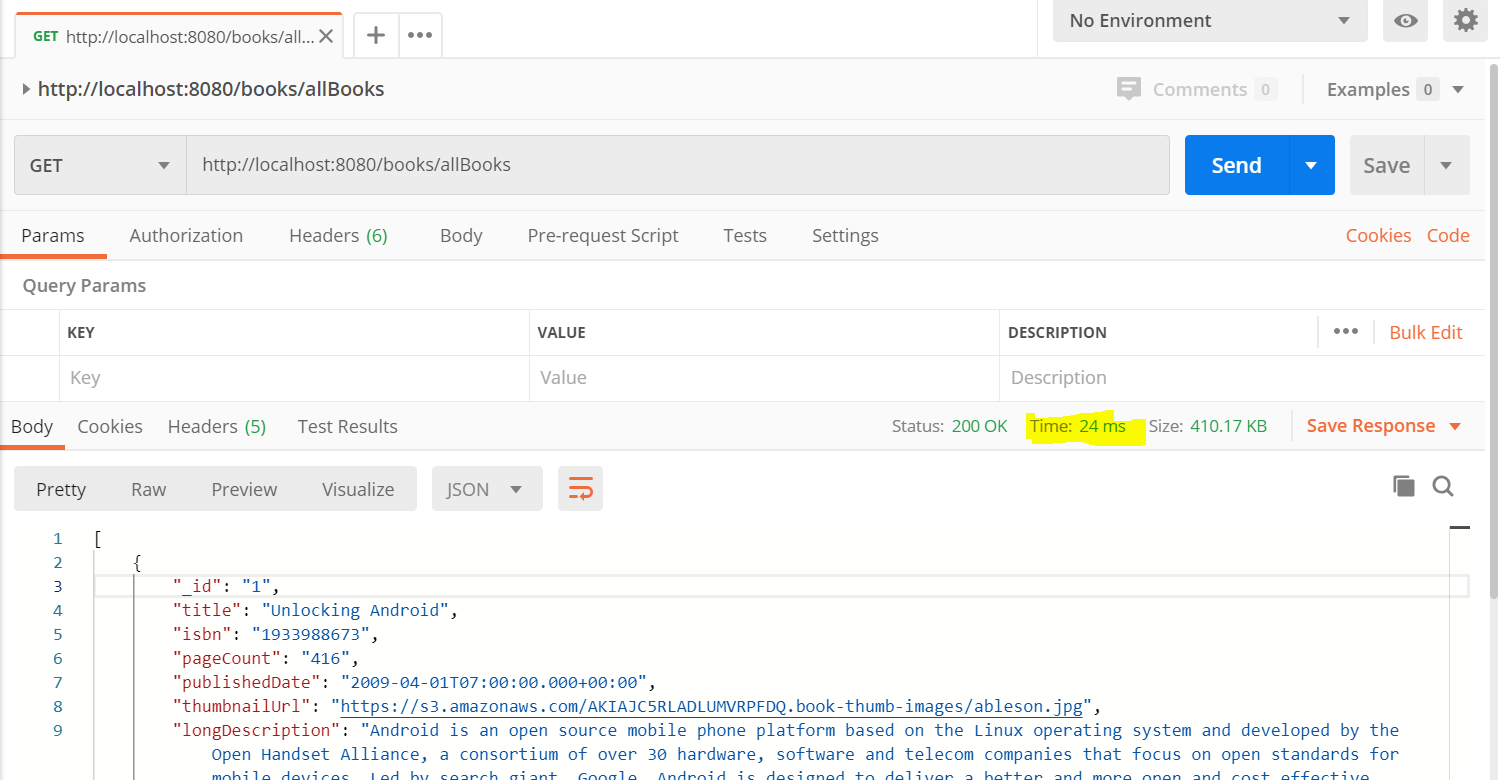
**First Time Hit:**



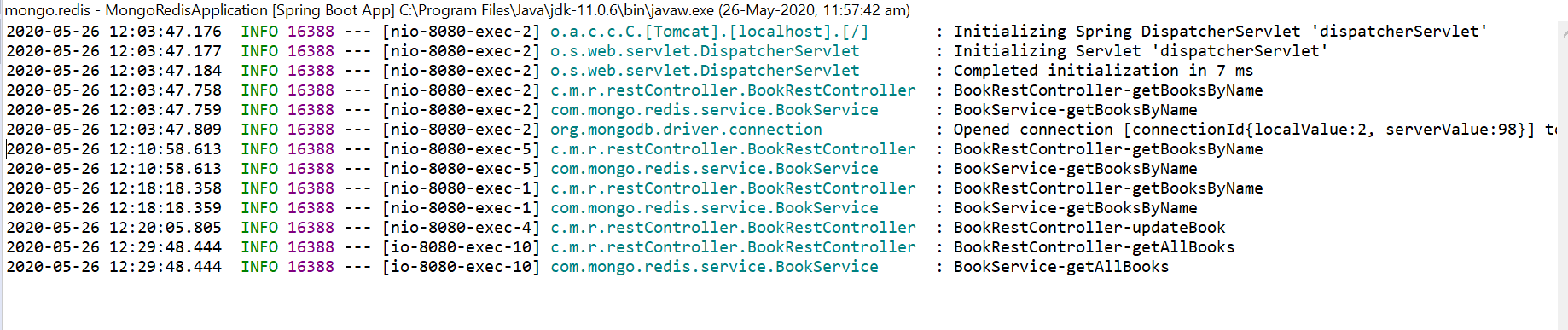




Second time hit:



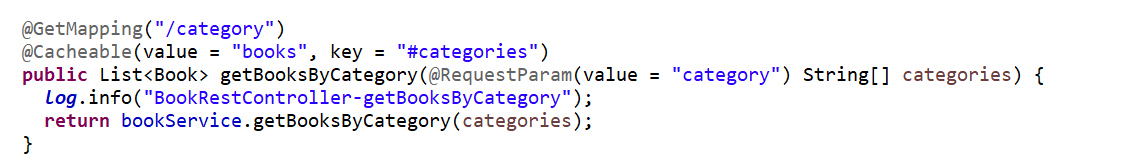
Did not go inside controller/service methods:

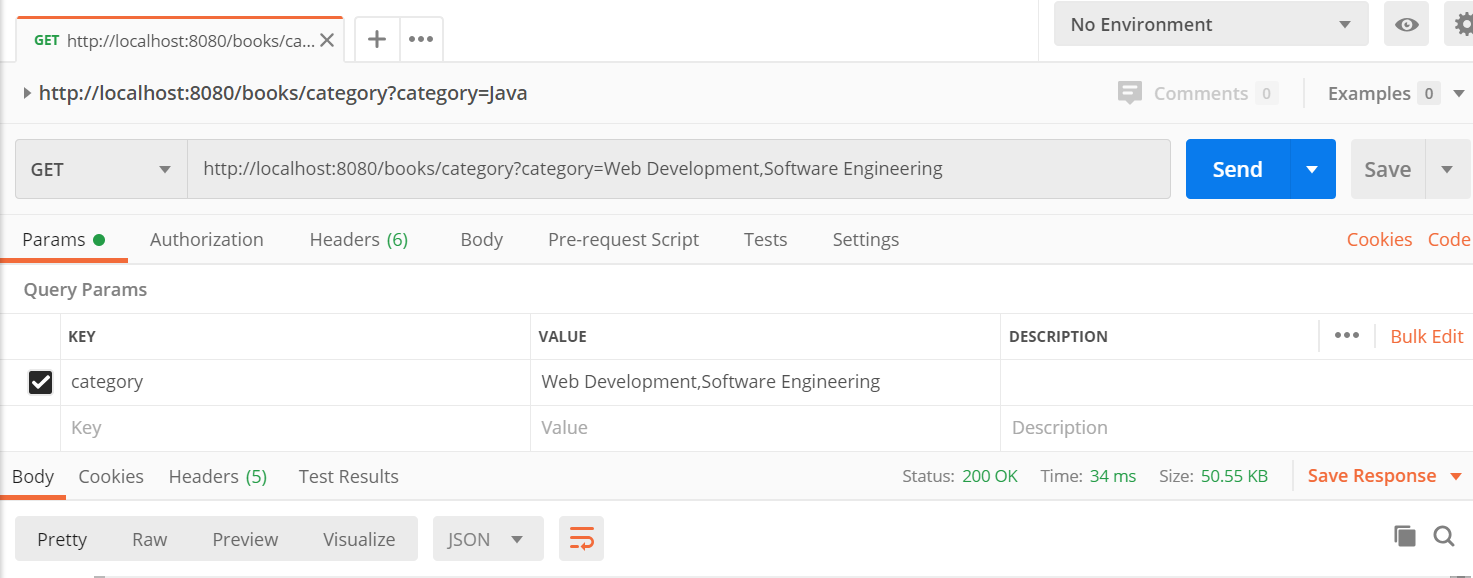


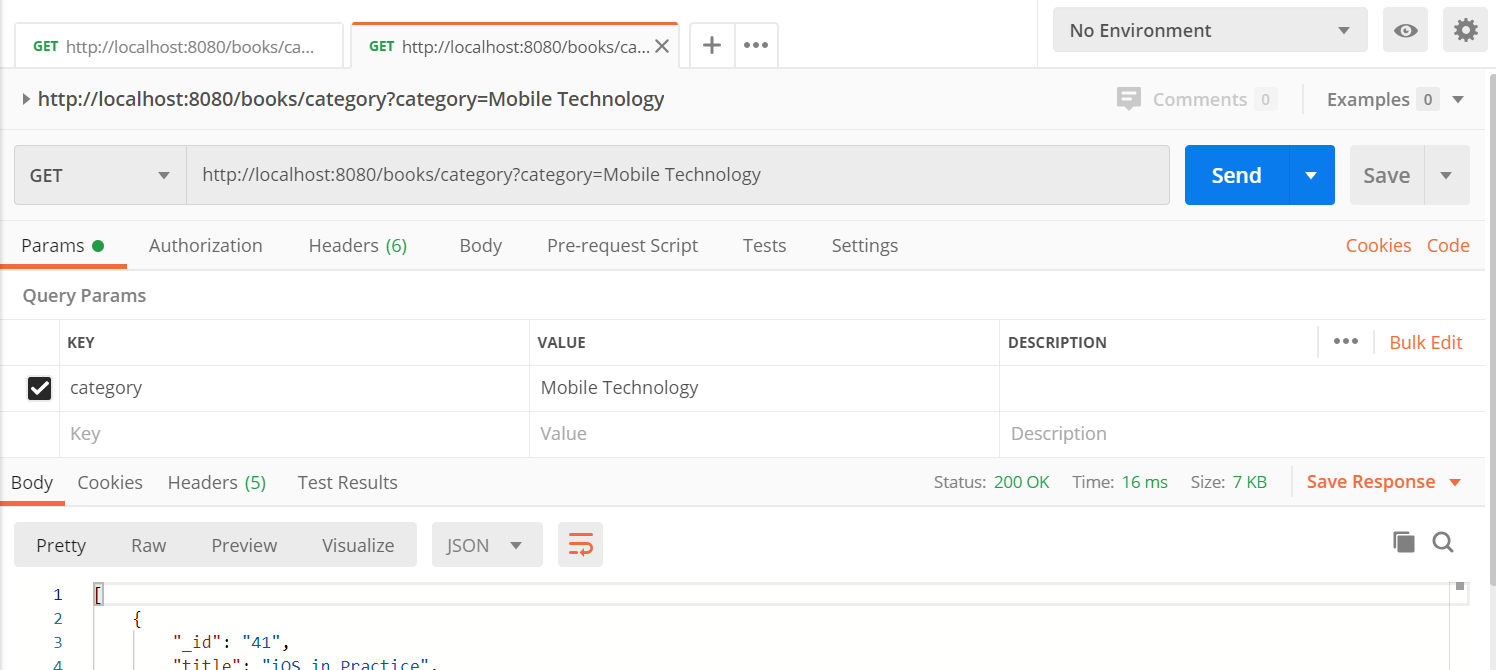
3.

* Here the cache key is the parameter “categories” .
* Each time the parameter value is changed it will hit db otherwise will fetch it from cache.
* We don’t have any separate method to update cache.
* Once the cache is expired then it will hit db and update the cache with appropriate result.

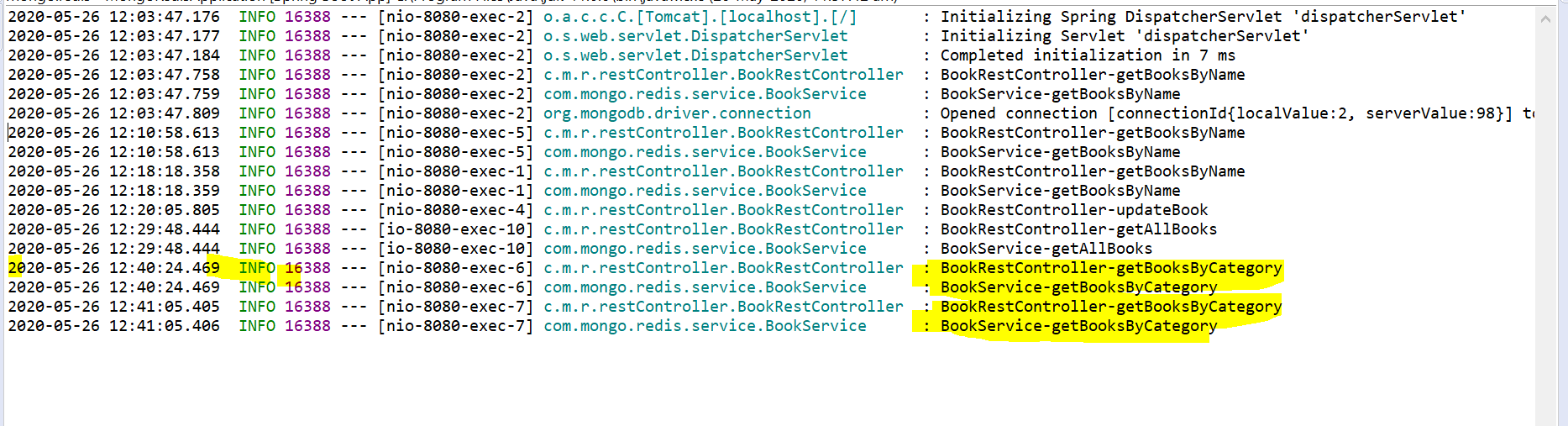
Controller:



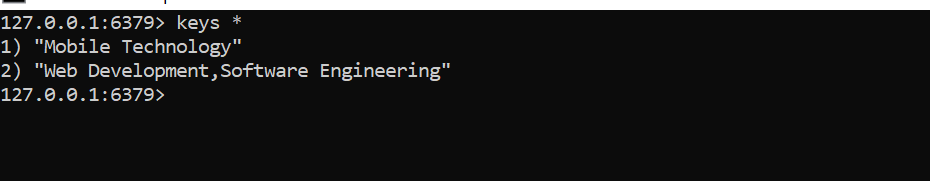




Logs:

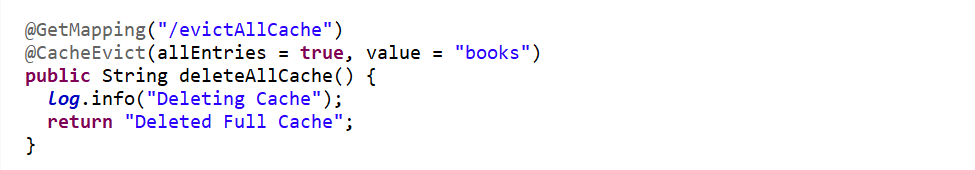


Keys:

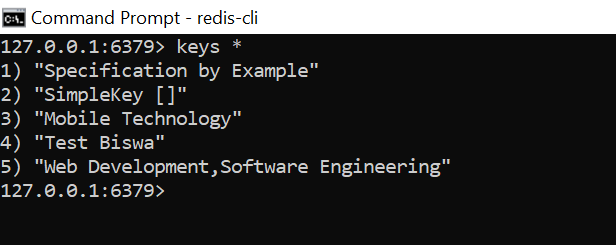


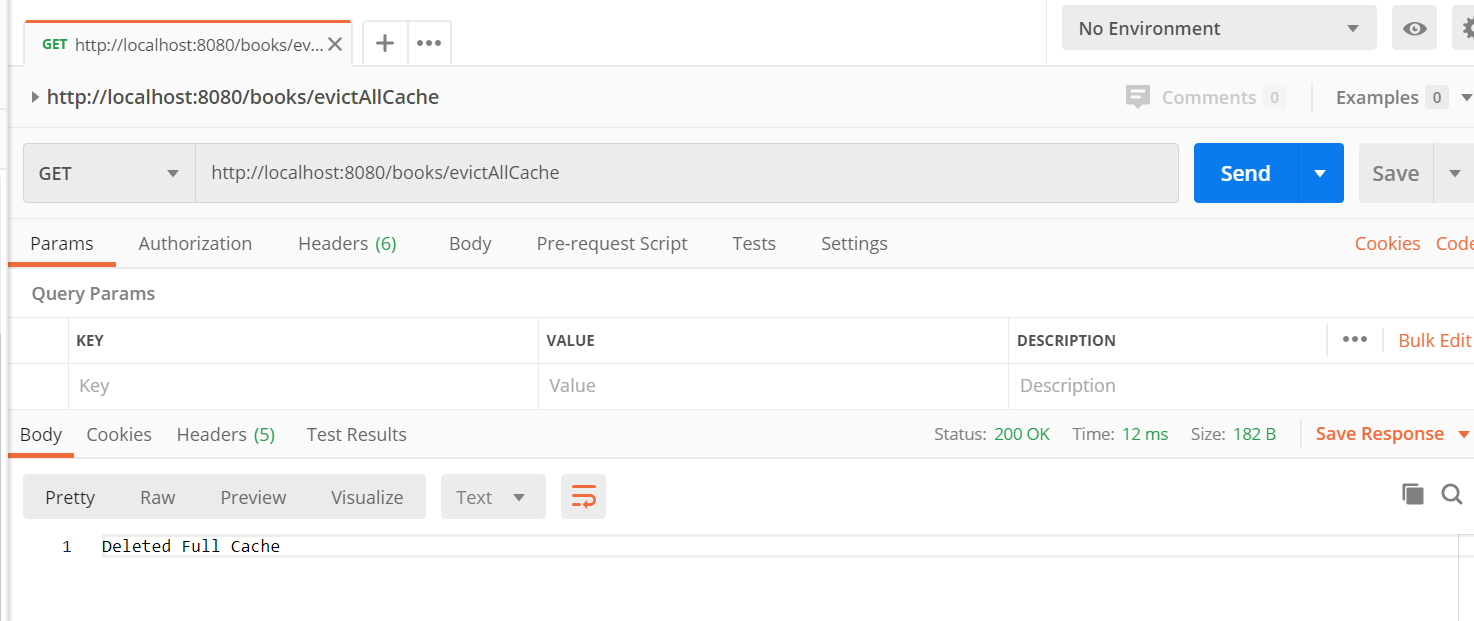
**Manual Eviction:**

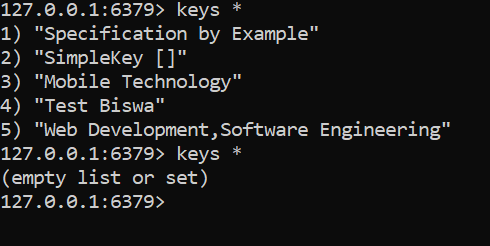
Controller:



Before run:

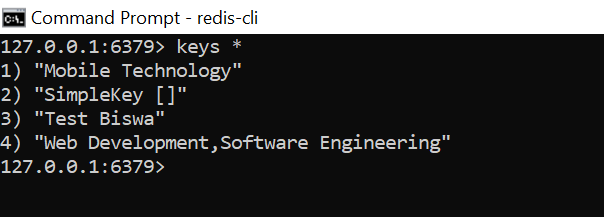




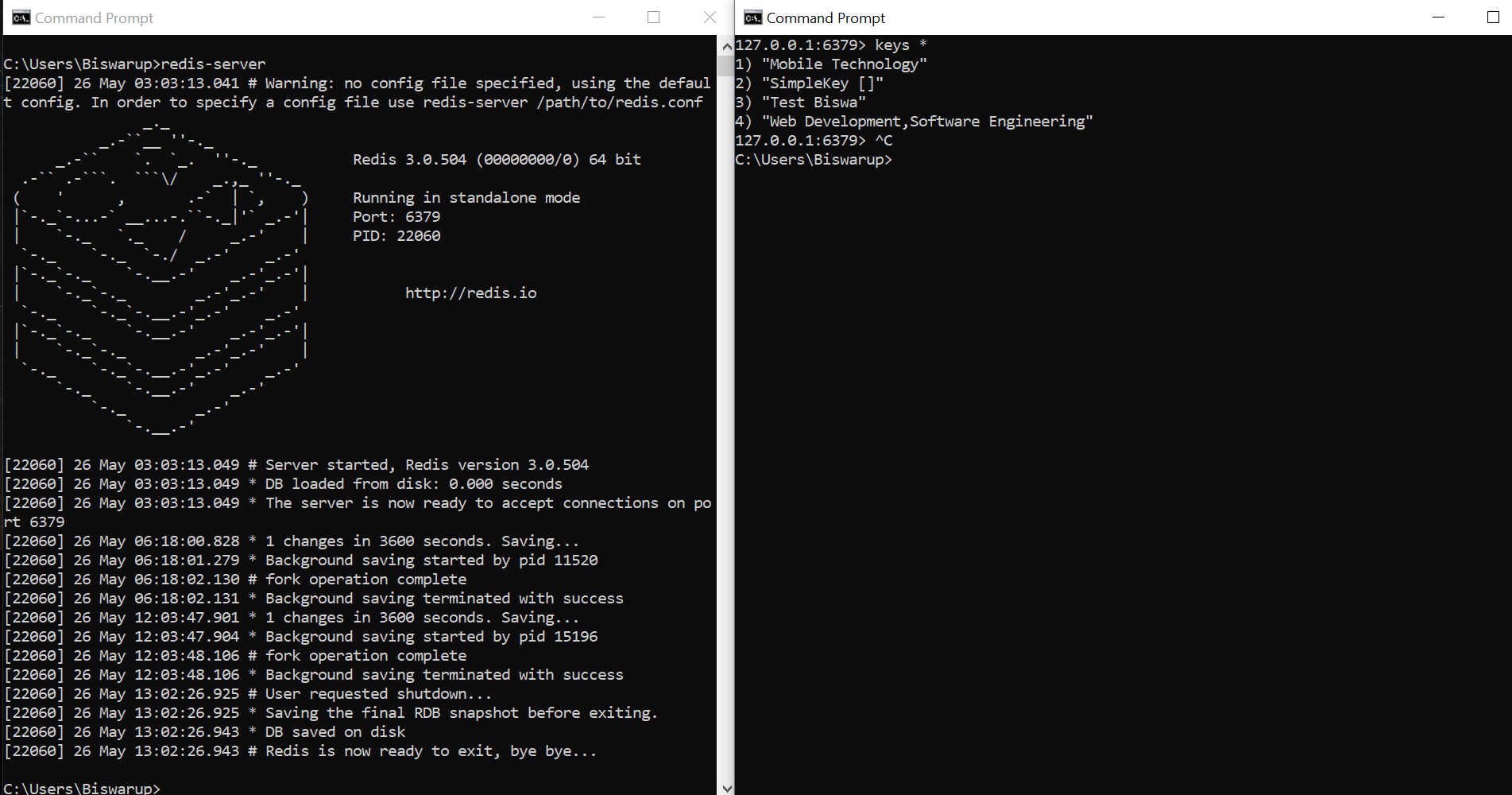


**Cache Persistence:**

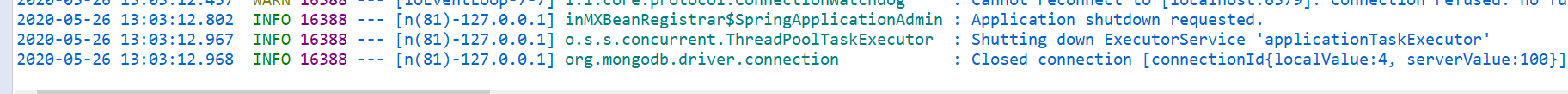
Available key:



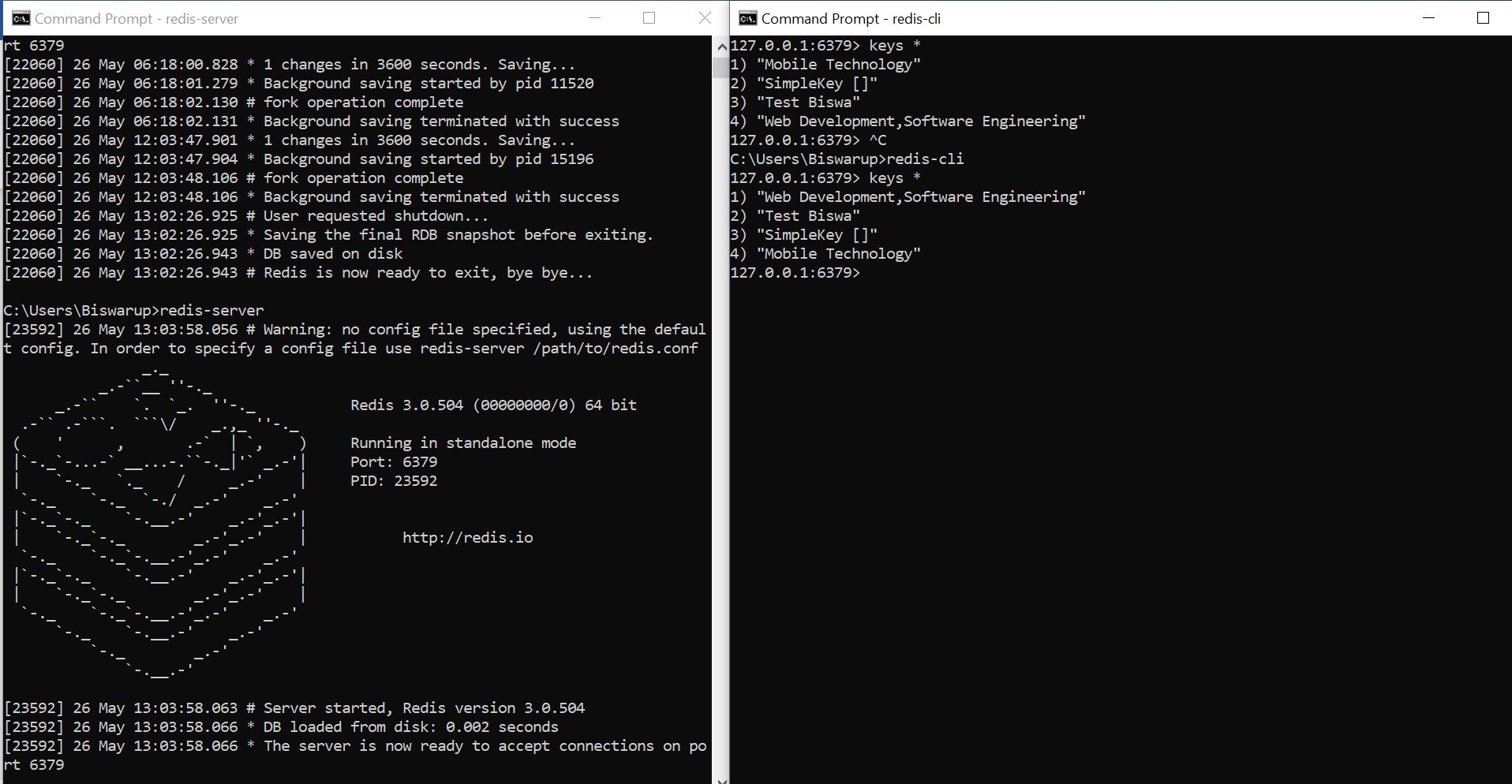
Stop the redis server:



Stop the application:



Cache are still available:



**Need to find Solution:**

Not able to create cache keys which can be used more than one method. So that any update in the database can update cache for all methods.

1.Currently my cache key is method parameter dependent. So methods with same parameter can take full advantage of the cache .

e.g we saw above that put method which updates book by title updated the cache key with “#title” during its execution and next time get method to fetch the book get the updated value directly from cache.

2. But where same key is not present for cache evict those are dependent on cache expiry.

e.g.

getAll() method fetched 400 records from db and store in the cache.

Next five more records are created and ten records are updated.

But we won’t get the updated result from this get method until cache is expired.