Blue Optima Assignment

Rate Limiter

Sparsh Jain

# 

## Approach 1:

Use a Database or any persistent store to keep all configurations and check them with every API hit.

**Cons** : Adds latency to the API for DB calls.

# Approach 2:

Keep the configuration about User and limits assigned per API in ConcurrentHashMap to keep everything in memory.

**Pros** : It is a simplistic solution and won’t add much latency.

**Cons** : ConcurrentHashMap doesn't lock the entire collection while performing modification. So, it may not provide accurate information about its size using size() operation.

## Approach 3(Implemented):

Use a Redis to keep all those configuraitons in cache.

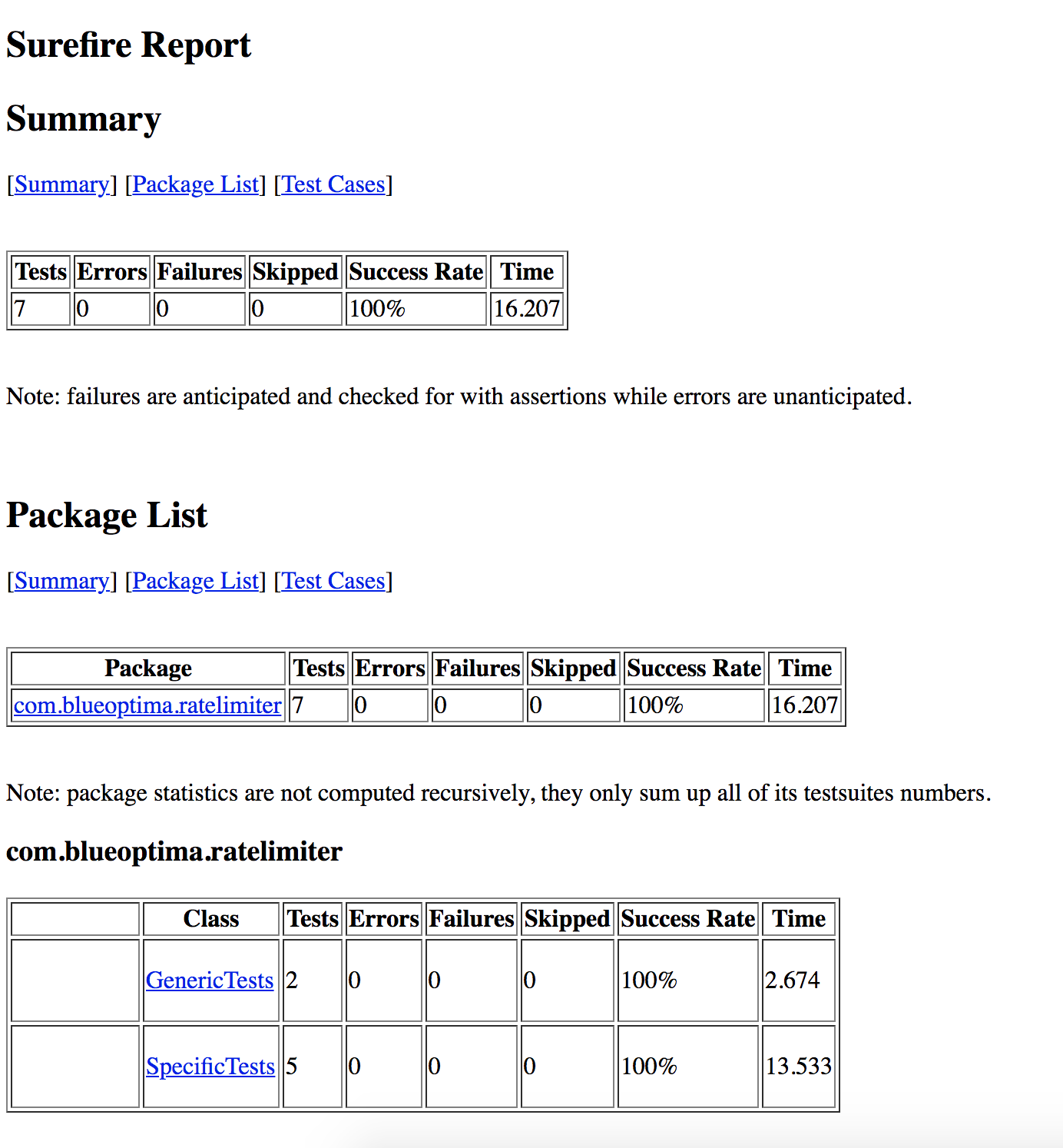
**Pros** : 1. Won’t add latency

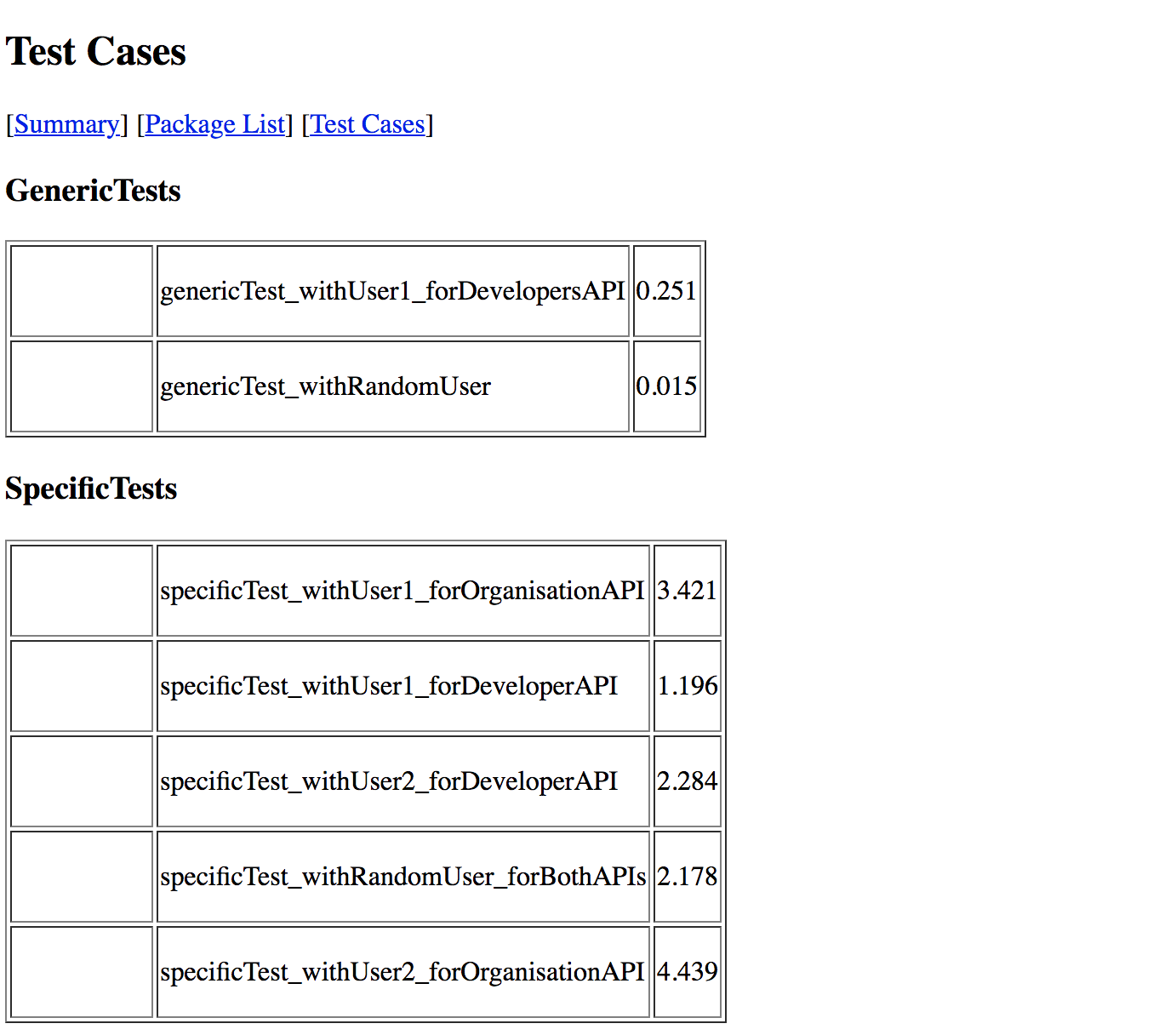
2. Can take advantage of atomic locking in Redis transactions.

**Highlights:**

1. Two approaches are used, one for global rate limiting and other for very specific rate limting which was asked in requirement doc.
2. **429** status code is returned whenever the limit get exhausted in any of the cases.
3. Two annotation are created 🡪 GenericLimit and SpecificLimit
4. In GenericLimit, you can configure number of permits per TimeUnit to a user for all the APIs.
5. This annotation is used for global configuraitons.
6. SpecificLimit is used to configure number of permits per user per API level.
7. Both the scenarios are described with example in GenericController and SpecificController respectively.
8. APIs asked to implement in requirement doc are implemented using both the approaches.
9. Tests have been written to support the required implementation and asked scenarios.

**Test Reports :**

****

****

**Steps to use/run this Project:**

1. Bring redis docker container up using docker-compose file given in project.

**docker-compose up -d**

1. Build project using

**mvn clean install test surefire-report:report**

1. To manually test the APIs and rate limiting feature using postman use below command and curls.

**mvn spring-boot:run**

**curl --location --request GET 'http://localhost:8888/generic/developers' \**

**--header 'userid: user2'**

**curl --location --request GET 'http://localhost:8888/generic/organisations' \**

**--header 'userid: user2'**

**curl --location --request PUT 'http://localhost:8888/specificConfig' \**

**--header 'Content-Type: application/json' \**

**--data-raw '{**

**"controllerName" : "DemoController",**

**"methodName" : "dynamicTest",**

**"userId" : "user2",**

**"permits" : 4**

**}'**

Other approaches and ideas can be taken from :

1. Using spring-cloud with redis 🡪 couldn’t really compare this to other ideas
2. Implementing Throttling

<https://medium.com/teamarimac/implementing-throttling-in-java-spring-boot-ec4723cfce9f>

1. Basic API rate limiting(This one uses ConcurrentHashMap only)

<https://dzone.com/articles/basic-api-rate-limiting>

1. Creating a thread safe library

<https://stripe.com/en-in/blog/rate-limiters>

1. <https://github.com/bit33/rate-limiter>
2. AsyncRateLimter : <https://github.com/seyran/AsyncRateLimiter/blob/master/AsyncRateLimiter.java>
3. Using Lettuce while this project uses caffeine

<https://github.com/alexthered/demo-api-rate-limiter>

1. Some code help is taken and customized from this:

<https://github.com/tangaiyun/redislimiter-spring-boot>