

# COUNT SPECIFIC CHARACTER - SPRING BOOT MAVEN PROJECT

Sunday, May 14, 2017 10:55 AM

## Tools/Technology Used:

JAVA (JDK 1.8)  
ECLIPSE STS  
GIT  
MAVEN  
MONGODB  
RESTFUL  
POSTMAN

## Business Requirement

### BASE CASE

Count the number of times a particular character occurs in a string.

Ex: ('Vivek', v)

Output : 2 (v occurs twice in Vivek)

**Converted the CountSpecificCharacter-PlainJava project into Spring Boot Maven project.**

Multiple inputs so far have been provided in a text file.

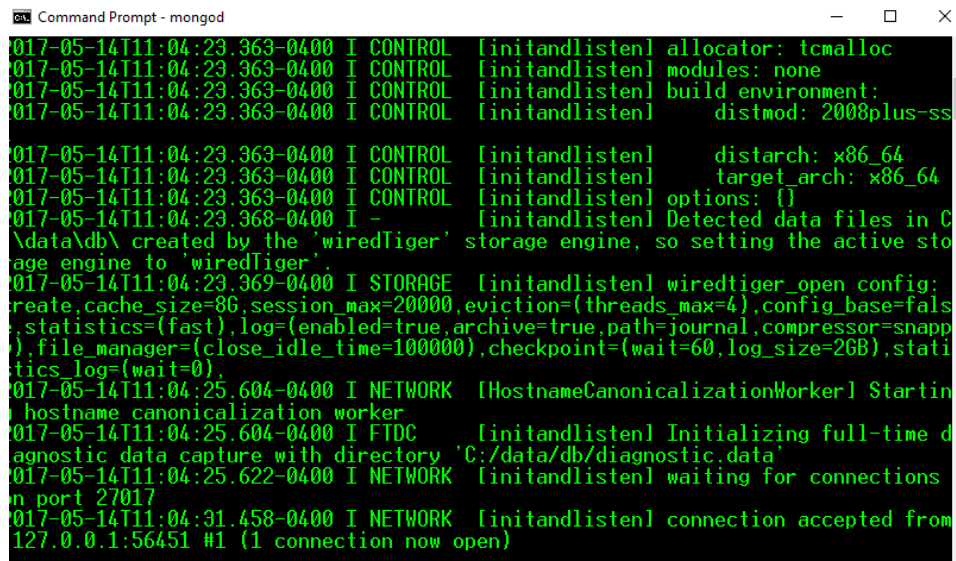
The outputs will also be provided in the text file upon running the service.

Upon running the project, the following URI can be used to run the service to get the result

Of all the outputs that are currently redirected to a output file.

- Currently no interaction is present with Mongodb. But, still mongodb dependency has been added for future implementation.

To run this project, mongodb port needs to be open on your local.



```
Command Prompt - mongod
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] allocator: tcmalloc
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] modules: none
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] build environment:
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] distmod: 2008plus-ss
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] distarch: x86_64
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] target_arch: x86_64
2017-05-14T11:04:23.363-0400 I CONTROL [initandlisten] options: {}
2017-05-14T11:04:23.368-0400 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-05-14T11:04:23.369-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=8G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0).
2017-05-14T11:04:25.604-0400 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2017-05-14T11:04:25.604-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2017-05-14T11:04:25.622-0400 I NETWORK [initandlisten] waiting for connections on port 27017
2017-05-14T11:04:31.458-0400 I NETWORK [initandlisten] connection accepted from 127.0.0.1:56451 #1 (1 connection now open)
```

URI PATH: <http://localhost:<server>//count/char/>

Sample Output Response (Obtained Through POSTMAN):

```
[
{
```

```

    "1": [
      "drum",
      "d",
      "1"
    ]
  },
  {
    "2": [
      "van",
      "v",
      "1"
    ]
  },
  {
    "3": [
      "sucessful",
      "s",
      "3"
    ]
  },
  {
    "4": [
      "triumph\\",
      "m",
      "1"
    ]
  },
  {
    "5": [
      "queenelizabeth",
      "q",
      "1"
    ]
  },
  {
    "6": [
      "poopooguys",
      "o",
      "4"
    ]
  },
  {
    "7": [
      "samosa",
      "z",
      "0"
    ]
  }
}
]

```

#### **ADDED IMPLEMENTATION**

//Input Type: JSON

https:<server>/CntCharService/json/addInput  
//Add string and characters to db (single and multiple request)

**SAMPLE REQUEST 1**

```
[
    {"stringInput": "Ram", "charInput": "v"},
    {"stringInput": "JohnJ", "charInput": "j"}
]
```

**SAMPLE RESPONSE 1 (NORMAL)**

```
{
    "STATUS": "SUCCESS",
    "DESCRIPTION": "3 records inserted into db"
}
```

**SAMPLE RESPONSE 2(FAILURE)**

```
{
    "STATUS": "FAILURE",
    "DESCRIPTION": "Error while inserting records into db"
}
```

**https:<server>/CntCharService/json/UpdateInput**

//Modifying the string and characters to db (single or multiple request)

**SAMPLE REQUEST 1**

```
[
    {"id": "1", "stringInput": "Vivek1", "charInput": "v"},
    {"id": "2", "stringInput": "Vivek2", "charInput": "j"}
]
```

**SAMPLE RESPONSE 1 (NORMAL)**

```
{
    "STATUS": "SUCCESS",
    "DESCRIPTION": "2 records updated into db"
}
```

**SAMPLE RESPONSE 2(FAILURE)**

```
{
    "STATUS": "FAILURE",
    "DESCRIPTION": "Error while updating records into db"
}
```

**Next Step Implementations**

- ☐ Integration with MongoDB
- ☐ Perform input validations, Exceptional handling
- ☐ Add additional scenarios in REST
- ☐ Add additional business cases itself
  - Support for Reading/Writing XML
  - Support for Reading/Writing EXCEL
- ☐ JAVA 8 CONCEPTS (Streams, Lambda)
- ☐ Using polymorphism , abstract, overriding concepts
- ☐ Unit testing (JUnit + Mockito +Coverage)

- ☐ Add security layer
- ☐ Possible Integration testing
- ☐ Integration to Angular Frontend
- ☐ Building Project using Jenkins
- ☐ Data structure concepts - Binary search, constant time
- ☐ Possible Threading scenario