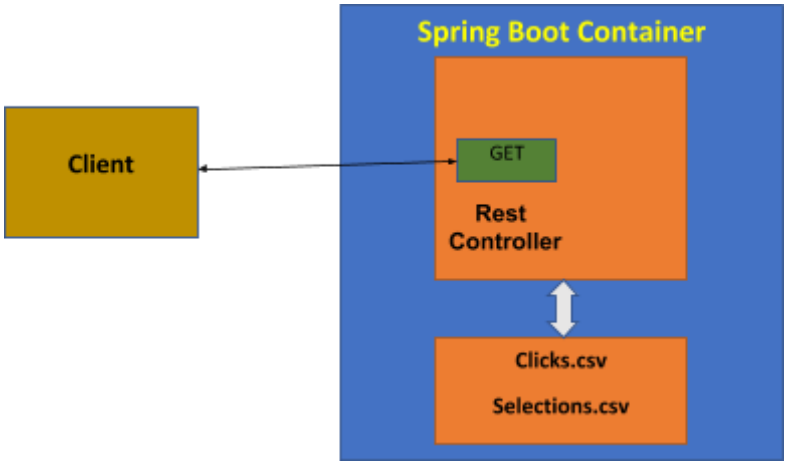
Product Catalogue Service



Functionalities:

#1. for a particular user, what are the top-N amenities selected.

HTTP GET : http://localhost:9090/productcatalogue/hotels/1063458341271753948 (valid User Id)

Response :


```
{
  "1063458341271753948": [
    {
      "updatedAt": 1547312012,
      "usrId": 1063458341271753948,
      "hotelId": 3501478,
      "hotelRegion": "Southern Region"
    },
    {
      "updatedAt": 1547309031,
      "usrId": 1063458341271753948,
      "hotelId": 2675020,
      "hotelRegion": "Central Region"
    },
    {
      "updatedAt": 1547310369,
      "usrId": 1063458341271753948,
      "hotelId": 3224804,
      "hotelRegion": "Central Region"
    },
    {
      "updatedAt": 1547308851,
      "usrId": 1063458341271753948,
      "hotelId": 3769748,
      "hotelRegion": "Central Region"
    }
  ]
}
```

#2. for a particular user, what are the top-K hotels clicked on the most for a particular user.

HTTP GET: http://localhost:9090/productcatalogue/amenities/2835680135910998

Response ;


```
{
  "2835680135910998": [
    {
      "updatedAt": 1547281508,
      "usrId": 2835680135910998,
      "amenityId": 967149002
    },
    {
      "updatedAt": 1547282781,
      "usrId": 2835680135910998,
      "amenityId": 7505603
    },
    {
      "updatedAt": 1547283925,
      "usrId": 2835680135910998,
      "amenityId": 212820802
    },
    {
      "updatedAt": 1547284000,
      "usrId": 2835680135910998,
      "amenityId": 337409602
    }
  ]
}
```

- What are the assumptions that you made during the implementation?

  Ans :

  1. The csv files are the final set of data.
  2. No write operation are being performed on those file.
  3. We are only going to retrieve the data on the basis of certain UserId.
  4. csv file are well formed, nothing is corrupted.

- If you could load test it, what do you expect to see in the result?

  Ans : I'm expecting the average waiting time for the GET request is 2.5 second. As files are being read as byte stream I have no control for optimization from the moment when the file pointer reads the file on an InputStreamReader. For file read operation is pretty time consuming as the file pointer points to the file then lock the file and scan it as byte streams. Now for my request the file is being filtered for a certain User Id, and it is more complicated.

- If you had more time, how would you improve your solution?

  Ans: I would love to add some more features to the application

  1) I would love to add an in memory database. Once I read all the the value then I would put all the value inside the database.
  2) I would then read the data from the in memory database. Even I would add two POST request that actually write to the database.
  3) I would love to add a few more request to get data for more then one user.
  4) And obviously I need to add then jUnit and slow tests.
  5) I could have made some method generic in service level and Util class.

- What other user insights could we possibly generate from this data?

  Ans : We can get to know user preferences better from this type of data. And we can make UI better according to user's need. That might be for selecting filters, or might be used for neural networks to produce better facilities through AI/ML.

- If you had to update the data source in real time, how would your solution change?

  Ans : I think

  1. If I could get an API or some data source that get updated with every hit in the Trivago server, from your side through some headless browser, then I can

scrap the data through that headless browser and parse the HTML or JSON values. I'll write a job that will run after a defined time interval.With this web scraping I'll have updated data every time.

2. Or the simplest solution would be to write to POST request that will write the to given csv file. Then to get the real time data just after the POST request I need to hit the GET request.

- What comments would you expect when this goes to a code review?

  Ans :

  1. Some comment about code reusing in middle tire.
  2. Could get comment about using some open source jar to read csv files.
  3. Proper logging (could have use log4j, slf4j).
  4. About validations.

To Run the application please follow :

1. from the root folder run 'mvn clean install'
2. run 'docker build -t trivago .'
3. docker image ls (to check the created image)
4. docker ps -a (to check status)
5. docker run -p 9090 trivago (Run the app)
6. Hit URL from postman.