Project 4 Task 2 – Find Activity App

By Enliang (Leo) Wu

Description:

My application takes a username string from the user, and to give user recommend activity by clicking “Find Activity” Button. User is able to choose “Do it” or “ Try another activity” by clicking the corresponding button and also able to return to find activity just by clicking “Back to find activity” button.
And my web application will record all operations from clients. Those records will be processed by analysis functions. User will first see three tables of analysis, they are “High frequency Activity Types TOP 10”, “Just DO IT Activities TOP 10” and “DISLIKE Activities TOP 10”.
The dashboard URL is:
https://frozen-castle-25997.herokuapp.com/
Or
https://frozen-castle-25997.herokuapp.com/index.jsp

1. Log useful information

In my web application, I logged useful information as the below table shows:

Name	Meaning	Example
Action Name	The name of the operation.	Client Request, Request API
User	The name of the user from android client input.	John
Action Time	The exactly date and time when this action performed.	2022-03-30T04:50:07.161
Activity ID	The id of an activity.	6553978
Client	The name of the client.	Mozilla/5.0, okhttp/4.9.3
Client Request	HTTP request method and target route.	GET /activity/doit
Client Reply	The content of reply to client.	REPLY: name: Look at pictures and videos of cute animals, id: 2565076, type: relaxation, result: true, message: ""
3rd Request	HTTP request method and URL to 3rd party.	GET http://www.boredapi.com/api/activity/
3rd Reply	The content of reply from 3rd	REPLY: activity "Uninstall unused

	party.	apps from your devices" with type "busywork" KEY(2850593) price: 0.0, participants: 1, accessibility: 0.0, link:

2. Store the log information in a database

I use InfoStore class to implement all operations with mongodb in InfoStore.java class Module.
I use POJO classes to store and retrieve data from mongodb.
I created to POJO classes: BoringActivity and ActivityLog.
Reference link: https://www.mongodb.com/docs/drivers/java/sync/v4.3/quick-start/
Both these two classes have empty constructor methods that will ensure mongodb lib being able to create POJOs.
Then all operations with mongodb will be easier and better understanding.
An example of POJO class:

```
public class BoringActivity {
	private String activityName;
	private String activityType;
	private String activityId;

	public String getActivityName() { return activityName; }
	public void setActivityName(String value) { activityName = value; }
	public String getActivityType() { return activityType; }
	public void setActivityType(String value) { activityType = value; }
	public String getActivityId() { return activityId; }
	public void setActivityId(String value) { activityId = value; }

	public BoringActivity() { }
}
```

An example of inserting to collections:

```
// get infos collection
MongoCollection<ActivityLog> collection = db.getCollection("infos", ActivityLog.class);
// insert log record
collection.insertOne(_info);
```

3. Display operations analytics and full logs on a web-based dashboard

Here is a screenshot of how they being displayed to user:

View All logs

High frequency Activity Types TOP 10

Rank	Activity Type	Get Times
1	busywork	5
2	social	4
3	recreational	3
4	education	2
5	cooking	2
6	relaxation	1

Just DO IT Activities TOP 10

Rank	Activity Name	Times
1	Write a note of appreciation to someone	1
2	Pull a harmless prank on one of your friends	1
3	Organize your dresser	1
4	Learn to greet someone in a new language	1
5	Learn calligraphy	1
6	Have a jam session with your friends	1
7	Go on a fishing trip with some friends	1
8	Fill out a basketball bracket	1
9	Clean out your car	1
10	Bake a pie with some friends	1

DISLIKE Activities TOP 10

Rank	Activity Name	Times
1	Plan a vacation you've always wanted to take	2
2	Go to the gym	2
3	Write a short story	1
4	Teach your dog a new trick	1
5	Take your dog on a walk	1
6	Study a foreign language	1
7	Start a daily journal	1
8	Start a blog for something you're passionate about	1

3.1 A unique URL addresses a web interface dashboard for the web service.

The unique URL address is:
https://frozen-castle-25997.herokuapp.com/index.jsp

3.2 The dashboard displays at least 3 interesting operations analytics.

“High frequency Activity Types TOP 10”, “Just DO IT Activities TOP 10” and “DISLIKE Activities TOP 10” are my 3 interesting operations analytics.
High frequency Activity Types TOP 10:
Among those random activities I grouped them by the types of activities. And counted the getting times of type of activity. Sorted by getting times by descending order.
Just DO IT Activities TOP 10: The top 10 activities user clicked “DO IT!” button to.
DISLIKE Activities TOP 10: The top 10 activities user clicked “TRY ANOTHER” button to.

3.3 The dashboard displays formatted full logs.

Considering a mess of things will show in dashboard page, I moved full logs to another url which is https://frozen-castle-25997.herokuapp.com/logs.jsp and there also a link in dashboard page.

```
<a href=”logs.jsp”>View Full Logs</a>
```

The formatted full logs table was implemented by tags in jsp looping:

```
<% for (ActivityLog log : logs) { %>
        <tr>
            <td><%=log.getActionName()%></td>
            <td><%=log.getUserName()%></td>
            <td><%=log.getActionTime()%></td>
            <td><%=log.getActivityId()%></td>
            <td><%=log.getClientInfo()%></td>
            <td><%=log.getClientRequestInfo()%></td>
            <td><%=log.getClientReplyInfo()%></td>
            <td><%=log.getApiRequestInfo()%></td>
            <td><%=log.getApiReplyInfo()%></td>
        </tr>
    <% } %>
```

Here is a screenshot of full logs page:

Action Name	User	Action Time	Activity ID	Client	Client Request	Client Reply	3rd Request	3rd Reply
Request API	John	2022-04-09T23:33:49.135	2896176	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Clean out your car" with type "busywork" KEY(2896176) price: 0.0, participants: 1, accessibility: 0.08, link:
Client Request	John	2022-04-09T23:33:49.211	2896176	okhttp/4.9.3	GET /activity	REPLY: name: Clean out your car, id: 2896176, type: busywork, result: true, message: ""	null	null
Client Request	John	2022-04-09T23:33:52.546	2896176	okhttp/4.9.3	GET /activity/doit	REPLY: id: 2896176, result: true, message: ""	null	null
Request API	John	2022-04-09T23:33:54.027	1288934	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Pull a harmless prank on one of your friends" with type "social" KEY(1288934) price: 0.1, participants: 1, accessibility: 0.2, link:
Client Request	John	2022-04-09T23:33:54.044	1288934	okhttp/4.9.3	GET /activity	REPLY: name: Pull a harmless prank on one of your friends, id: 1288934, type: social, result: true, message: ""	null	null
Client Request	John	2022-04-09T23:33:55.181	1288934	okhttp/4.9.3	GET /activity/doit	REPLY: id: 1288934, result: true, message: ""	null	null
Request API	John	2022-04-09T23:33:56.174	1770521	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Write a note of appreciation to someone" with type "social" KEY(1770521) price: 0.0, participants: 1, accessibility: 0.0, link:
Client Request	John	2022-04-09T23:33:56.185	1770521	okhttp/4.9.3	GET /activity	REPLY: name: Write a note of appreciation to someone, id: 1770521, type: social, result: true, message: ""	null	null
Client Request	John	2022-04-09T23:33:57.238	1770521	okhttp/4.9.3	GET /activity/doit	REPLY: id: 1770521, result: true, message: ""	null	null
Request API	John	2022-04-09T23:33:58.226	4387026	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Go to the gym" with type "recreational" KEY(4387026) price: 0.2, participants: 1, accessibility: 0.1, link:
Client Request	John	2022-04-09T23:33:58.257	4387026	okhttp/4.9.3	GET /activity	REPLY: name: Go to the gym, id: 4387026, type: recreational, result: true, message: ""	null	null
Request API	John	2022-04-09T23:33:59.389	7687030	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Contribute code or a monetary donation to an open-source software project" with type "charity" KEY(7687030) price: 0.1, participants: 1, accessibility: 0.0, link: https://github.com/explore
Client Request	John	2022-04-09T23:33:59.396	4387026	okhttp/4.9.3	GET /activity/dislike	REPLY: name: Contribute code or a monetary donation to an open-source software project, id: 7687030, type: charity, result: true, message: ""	null	null
Request API	John	2022-04-09T23:34:01.008	4286250	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Go for a walk" with type "relaxation" KEY(4286250) price: 0.0, participants: 1, accessibility: 0.1, link:
Client Request	John	2022-04-09T23:34:01.020	7687030	okhttp/4.9.3	GET /activity/dislike	REPLY: name: Go for a walk, id: 4286250, type: relaxation, result: true, message: ""	null	null
Request API	John	2022-04-09T23:34:01.852	1638604	null	null	null	GET http://www.boredapi.com/api/activity/	REPLY: activity "Have a football scrimmage with some friends" with type "social" KEY(1638604) price: 0.0, participants: 8, accessibility: 0.2, link:
Client Request	John	2022-04-09T23:34:01.862	4286250	okhttp/4.9.3	GET /activity/dislike	REPLY: name: Have a football scrimmage with some friends, id: 1638604, type: social, result: true, message: ""	null	null
Request	John	2022-04-	7265395	null	null	null	GET	REPLY: activity "Plan a vacation you've always wanted to take" with type "relaxation"

4. Deploy the web service to Heroku

I used a tomcat docker image to deploy to Heroku.
ROOT.war package and tomcat_starter.sh script are built into this image.
After my docker image being pushed to Heroku web application “frozen-castle-25997”.
My web service is available at: https://frozen-castle-25997.herokuapp.com/