Abstract

Use Java to write a production quality RESTful web service application that provides the functionality described below.

Problem description

We assume a table called login in a database called data. The table contains login data and has the structure shown below.

login_time (Timestamp)
user (String)
attribute1 (String)
attribute2 (String)
attribute3 (String)
attribute4 (String)

The objective is to write a web application called test that exposes a RESTful web service. The web service targets the following URLs:

http://server/test/dates
Retrieves a JSON array of all the unique dates (ignoring time) in the table
The resulting JSON array needs to be sorted ascending.

http://server/test/users?start=YYYYMMDD&end=YYYYMMDD
Retrieves a JSON array of all the unique users for which there is a login record between the start and end date.
Both parameters are optional, so there can be a start date, an end date, or both.
The resulting JSON array needs to be sorted ascending.

http://server/test/logins?start=YYYYMMDD&end=YYYYMMDD&attribute1=AAA&attribute2=BBB&attribute3=CCC&attribute4=DDD
Retrieves a JSON object where the key is the user name and the value is the number of times a user logged on between the start and the end date.
All parameters are optional.
The values used for the attributes are used as filters, i.e. only the records should be counted for which the attribute values are equal to the ones specified in the parameters.
For one attribute, multiple values might be present, e.g.
http://server/test/logins?attribute1=AA1&attribute1=AA2&attribute1=AA3

The assignment is purposefully simple, and the focus is on writing production quality code that performs well.

Deliverables

Delivery of the project is expected within one week after you receive this assignment. If you need more time let us know.

Maven project

- The complete application should be built using Maven 3.
- Include a mechanism to provision the database with 100,000 dummy records. Consider that this table can grow to millions of records in the near future.

Design document

- Optionally, you can include a document to explain the system and certain design decisions that you made. This can be useful if you feel that you had to make too many assumptions or did not have enough information to make an informed decision about a certain aspect.

Live application
Optionally, provide a URL where we can test the application, e.g., DigitalOcean, Heroku, AWS.