LAS3014 Spring Framework

Study Unit Assignment
Paul Cutajar

Necessary Preamble:

This document describes the assignment for study unit LAS3014, Spring Framework. This assignment is worth 100% of the total, final mark for this unit. You will be required to demonstrate (and be able to discuss) your assignment solutions in a 15-minute presentation. You are expected to allocate 45 hours to complete the assignment. The deadline for this assignment is **Friday 9th June 2017 at noon**. Late submissions will not be accepted.

Questions regarding the assignment should only be posted in the Assignment VLE forum. This is an individual assignment. Under no circumstances are you allowed to share the design and/or code of your implementation. The Centre for the Liberal Arts and Sciences takes a very serious view on plagiarism. For more details refer to plagiarism section of the University of Malta website[1]. You are to upload all of the code and documentation on the VLE website.

Important Note: The main objective of this assignment is to demonstrate you understood the Spring Framework concepts presented in class. Submitting a perfectly working solution, implemented without using Spring Framework will result in a failing grade.

Deliverables

The following deliverables are expected by the specified deadline. Failure to submit any of these artefacts in the required format will result in your assignment not being graded. Replace NAME and SURNAME below with your name and surname respectively.

- **LAS3014_SURNAME_NAME_assignment_code.zip** - A single zip file containing your **assignment code** and the **database schema**. This needs to be uploaded to VLE. It is your responsibility to make sure that this zip file has been uploaded to VLE correctly (by downloading and testing it). Failure in opening the zip file will result in your assignment not being graded. Note: *Do not include compiled class files in the zip file.*

- **LAS3014_SURNAME_NAME_assignment_doc.pdf** - The assignment documentation in .pdf format. The documentation has to be uploaded to VLE, together with your code. The report should not be longer than 7 pages. This report should not include any code listings.

[1] https://www.um.edu.mt/ict/Plagiarism

Technical Specification

Any code you supply will be run on Linux (distribution Ubuntu 14.04.3 LTS) and/or OSX (distribution Sierra). You are required to implement the tasks assigned using Java (version 8u*). Any external libraries that are used must be defined in a Maven POM file. These libraries must be downloadable from the public Maven Central Repository. Note that you should not have any absolute paths hardcoded in your programs.

The Era of Data Overload

With the introduction of internet connected mobile phones, and the rise of social media, a ton of content is being produced daily. People are waking up each morning bombarded with news, comments, links, videos etc. According to research by Pew Research Center, in America 38% of adults read their daily news online[2]. Also according to an infographic by Delvv[3], on average each person spends 60/h per week online and is surfaced 285 content articles every day! That translates to approximately 54,000 words and 443 minutes of video daily. The issue of surfacing this much data is that there is no way for users to weigh it / validate it resulting in data overload. To help alleviate this issue a system which can aggregate news and surface only personalized news is direly needed.

Solution

You have been tasked to create a personalized news aggregation system that connects to a news website and fetches all the articles published. It should then be able to match retrieved news articles based on the user's preference(s) and surface only those news articles that match.

For this project the news website that will be providing the articles will be Hacker News. Hacker News is a social news website that focuses on computing related news. The system implemented should connect to Hacker News API and retrieve new stories hourly while updating the score of already fetched ones. Once this process is complete the system should then match each story title retrieved/updated with the user preference(s) in store. For each preference the user has indicated, the system should surface the story with highest score at that moment given by Hacker News API (score changes over time).

A must-have feature for any news aggregation system is to create weekly digests. A weekly digest should consist of surfacing the the top three stories of the week irrelevant of the user preference and the top 3 stories of the week for each criteria given.

[2] http://www.journalism.org/2016/07/07/pathways-to-news/
[3] http://delvv.com/blog/?p=161

Features

The system you produce should have the following features:

- Business Processes:

    a) Connect hourly to Hacker News API, retrieve the newest stories and store them in a persistent storage. If the story already exists the system should update its score.

    b) The system should curate the highest ranked story per preference per user registered. The matching with the preference(s) is to be done by checking whether the keywords given are present in the title of the story.

    c) Every saturday, 9:00am the system is to create a weekly digest for every user registered as per specification above. A year long weekly digest information should be held in permanent storage.

    d) Stories that are older than 7 days which are not associated with a weekly digest should be deleted.

- REST APIs

    a) An API that allows a user to register with the system, allowing the user to provide a username, password and a set of topics (aka user preferences) interested in.

    b) A login API should be created that accepts a username and password. The system should then validate them with its persistent storage and if matching a user session token is created. A session expires after 30 min of inactivity.

    c) An API where a logged-in user can query for the current daily story per preference in store. For each preference the API should return the title of the story and the link where it can be accessed from.

    d) An API where a logged-in user user can query for stored weekly digests. The API is to accept a date range as input and surface the weekly digests that fall within the range specified.

e) An API where a logged-in user can query for the weekly digest. The latest curated weekly digest should be surfaced.

f) An API that allows a registered user to change the set of topics (user preferences) interested in. This call should not affect previous weekly digest(s) in store.

g) A logout API that invalidates a user session.

- Security

a) All APIs that expose user data must only be accessible by logged-in users. Only data associated with the current logged-in user should be surfaced.

b) Passwords must be encrypted using a one-way hashing algorithm.

Documentation

In any system created, good developers always document what decisions they took when selecting algorithms/technologies to solve the problem at hand.

a) Document the reasons/assumptions and justify your approach on any major decisions/algorithms used/implemented in the system.

b) The system should be horizontally scalable, meaning it should be able to be deployed on multiple servers running concurrently. Document what work would need to be done if any to allow such a production configuration.

Grading Scheme

The following criteria will be taken into consideration when grading your assignment:

- Ability to the solve the above problem, showing a deep understanding of the technology.
- Ability to critically evaluate your work (shortcomings, assumptions, etc.).
- Complete documentation of solutions as indicated above including: technical approach, testing, critical evaluation and limitations. Must be properly presented (e.g. no loose papers, page numbers, table of contents, captions, references, etc.)

- Adherence to coding standards, consistency, readability, comments, (no) compiler warnings, code organization using packages/namespaces, etc.
- Completeness and adherence to tasks' specification. Also, correctness of solutions provided.
- Use of exceptions and proper exception handling. Proper (and demonstrable) testing of solutions. No unexpected program crashes.
- Proper use of Spring Dependency Injection.
- Creation of a View layer, making use of but not limited to Controllers, RestControllers, input validation, RequestBody and URL parameters.
- Creation of a persistence layer, making use but not limited to JPQL, Ordering and sorting, ORM concepts such as relationships, transaction isolation and transaction propagation levels.
- Use of caching mechanisms where applicable
- Use of AOP advices where applicable
- Use of Spring Batch and/or Task Scheduling where applicable
- Use Thread pools and threading where applicable
- Unit tests, the business logic should be adequately tested.
- Java Docs on important class/methods.

Marking Scheme

Description	**Mark**
Use of Spring Technology as defined in the Grading Section	30
Completeness of Features as defined in the Features section	26
Documentation	17
Security	11
Tests	9
Adherence to coding standards, readable code, etc as per Grading Section point 4	7
Total	**100**

- END -