Intro

I was using **Singleton Desing Pattern and most Simplistic approach**, because of the simple nature of application functionality and no other specific requirements were specified. This statement valid for both Backend and Frontend parts.

Given more time, I'd use one of another UI libraries which gives to the App much better UX and UI consequently.

- Filter by creation and modification dates of items
- Filter by Item text
- Search by dates
- Sorting by dates and item id
- All deletion should be done offline depends on certain time or deletion queue reaching certain size. For user it would give possibility to restore deleted items for instance. Deleted items would be taken from UI and just marked for deletion.
- Import/export of ToDo items
- Allow multiuser usage for the app that each user has his own ToDo list and must authenticate to use app.
- Create reports for past ToDo items (from date to date, on certain date, having certain text, or mix of all)
- Create certain behavioral patterns for machine learning to make application easier for users
- Make data of the list secure by encrypting it or use blockchain as a storage

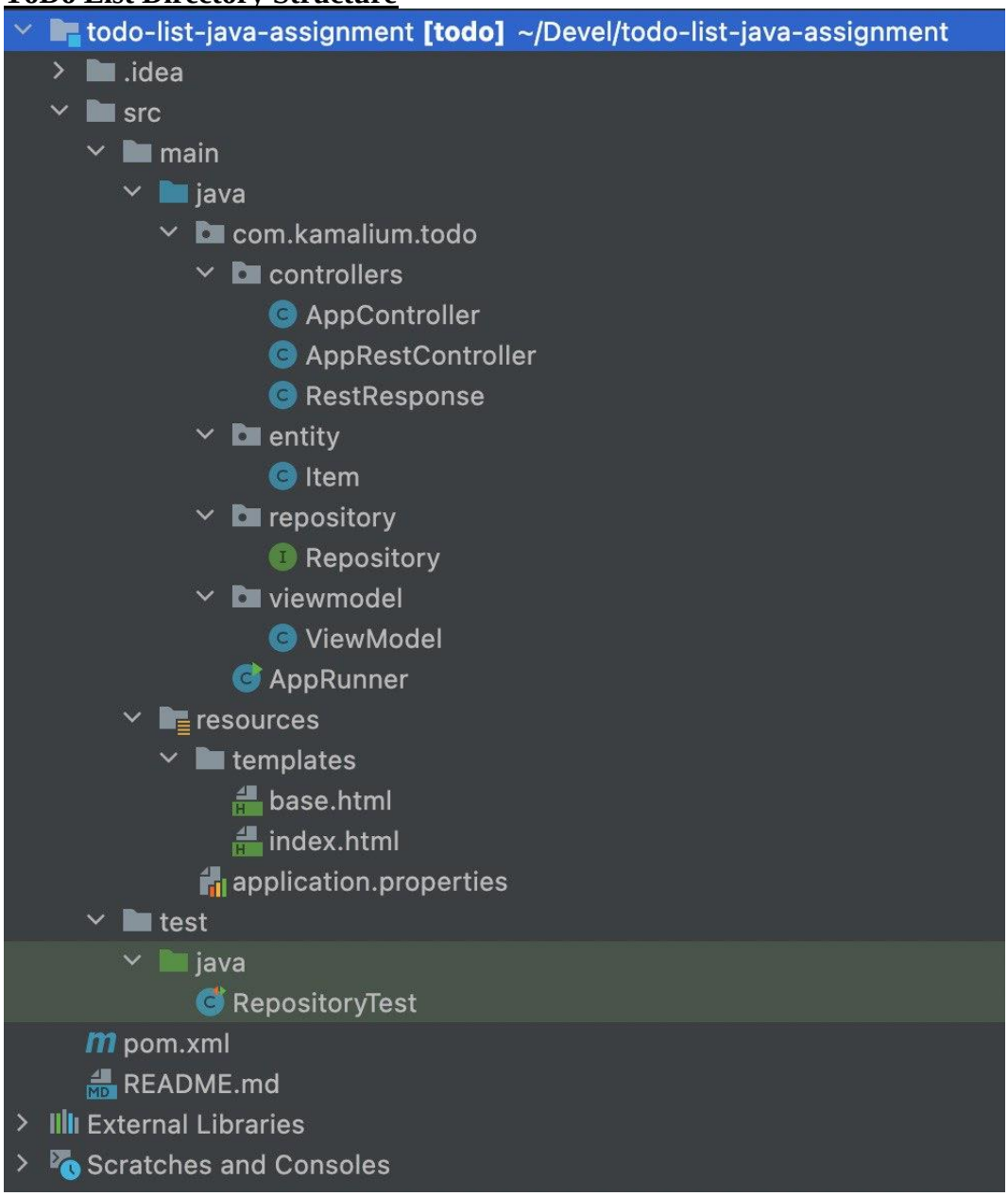
Would run beta with 100 users and ask them what must be added/changed/removed from their point of view.

What else is relevant? Definitely application scaling for performance in multiuser environment. I would use VertX instead of Spring, though Spring is more popular. This task should be defined as multiuser and also include LB and any Message Queuing for instance RabbitMQ to process high load data stream from users.

Tools used:

- MacOS 12.5
- JDK 11
- IntelliJ IDEA 2021.3.2
- Maven 3.8.5
- Git 2.32.1
- For server part used Spring Boot 2.2.6
- For client part used Bootstrap
- Chrome 103.0.5060.134, Safari 15.6 (checked with both) and Postman 9.27

ToDo List Directory Structure



Getting Application

Check out code from repository using GIT command from terminal:

git clone https://github.com/GreyCardina1/todo-list-java-assignment.git

or directly from IntelliJ using “Get from VCS” option.

Building Application

- Application built as single file with all dependency classes included for execution
- Project can be built as both: JAR or WAR by changing value in pom.xml <packaging> setting.

```
<packaging>war</packaging>
```

- AppRunner is the start up class for the Application.

Application can be built from both IntelliJ directly or CLI. To build project for CLI use following commands from terminal:

cd todo-list-java-assignment

mvn clean install

IntelliJ building application automatically before running by pressing **Play** button on the left of AppRunner class code.

```
@SpringBootApplication
public class AppRunner {

    public static void main(String[] args) { SpringApplication.run(AppRunner.class, args); }

}
```

In order to start Application with logging to terminal:

java -jar target/todo-0.0.1-SNAPSHOT.jar

with logging to the file **todo-list.log**:

java -jar target/todo-0.0.1-SNAPSHOT.jar >> todo-list.log

Logging

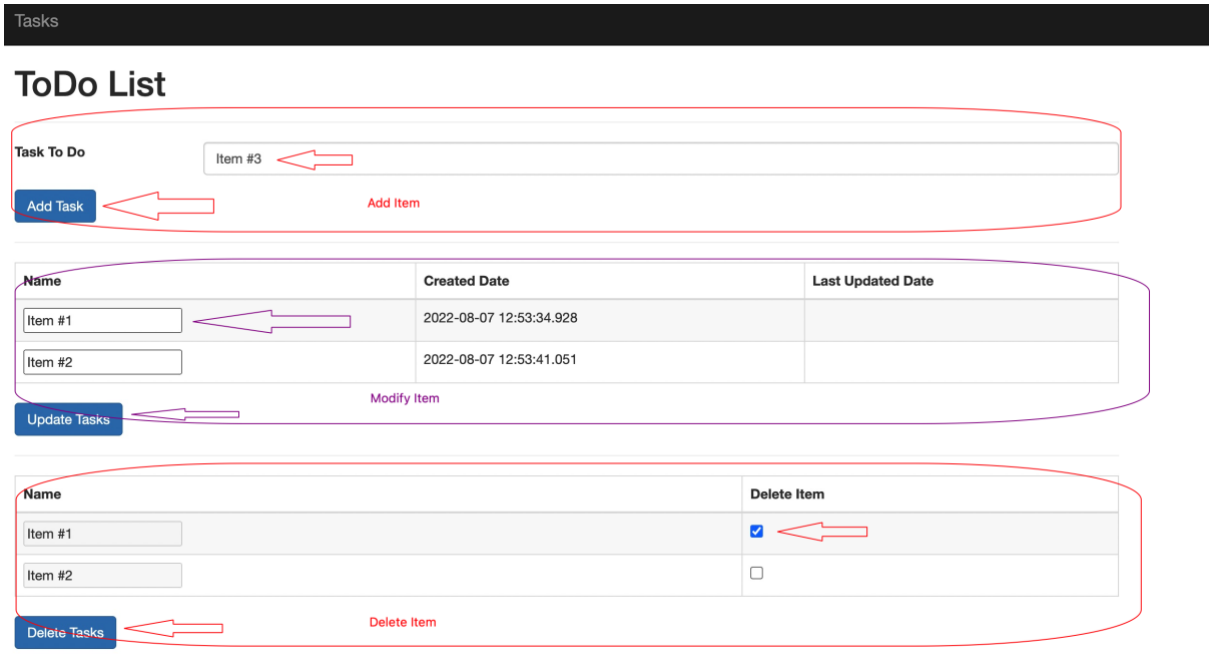
Logging can be changed from DEBUG to INFO level by changing Spring settings in application.properies file. After settings change application must be restarted to take effect.

Unit Test

Located in RepositoryTest class. More unit tests can be added at anytime.

Use ToDo tasks in browser

http://localhost:8080



There are 3 screen areas

- Add Task
- Update Tasks
- Delete Tasks

1. Add Task works by entering task text and pressing “Add Task” button
2. Update tasks works with in place editing. Saves modified Item by pressing Enter or button “Update Tasks”
3. Delete Tasks works by selecting checkbox on the right of the task user wants to delete and pressing button “Delete Tasks”.

Use ToDo tasks API's in Postman

Add Task - POST: http://localhost:8080/todo/add
JSON request:

```
{
    "item_value":"1Item1"
}
```

JSON response:

```
{
    "status": "ADDED",
    "item": {
        "id": 1,
        "item_value": "1Item1",
        "created_date": "2022-08-07T15:05:41.601+0000",
        "updated_date": null,
        "delete_item": false
    }
}
```

Update Task – POST: http://localhost:8080/todo/update
JSON request:

```
{
    "id":1,
    "item_value":"ToDo Item"
}
```

JSON response:

```
{
    "status": "UPDATED",
    "item": {
        "id": 1,
        "item_value": "ToDo Item",
        "created_date": "2022-08-07T15:05:41.601+0000",
        "updated_date": "2022-08-07T15:05:56.441+0000",
        "delete_item": false
    }
}
```

Delete Task – DELETE: http://localhost:8080/todo/delete?id=1
JSON response:


```
{
    "status": "DELETED",
    "item": {
        "id": 1,
        "item_value": null,
        "created_date": null,
        "updated_date": null,
        "delete_item": false
    }
}
```


Show All - GET: http://localhost:8080/todo/add?item_value=Task0001
JSON response:


```
[
    {
        "id": 1,
        "item_value": "ToDo Item",
        "created_date": "2022-08-07T15:05:41.601+0000",
        "updated_date": "2022-08-07T15:05:56.441+0000",
        "delete_item": false
    }
]
```