IBA Java Workshop VER.2.0
Week #3 Activities (31.01.2023 - 07.02.2023)

author: Mikalai Zaikin (mzaikin@ibagroup.eu)

Project description:

For week #3 we have a project which introduces Telegram Bot. The web application is not used this week, so it was temporarily removed for simplicity (and will return next week).

The project structure is the same – Maven-based and consists of 2 modules (common and Telegram Bot). We will learn how to run a Spring Boot application from a command line, although it is less convenient than to run as a Docker container.

The telegram bot demonstrates very basic functionality without state support. It means each user input assumes a new command, not data for the previous command. Traditionally TG bot commands start with a slash character.

We will add TG bot states next week. For now you will need to get familiar with the provided telegram bot project, understand main classes, fix some bugs, and add new commands which interact with MongoDB. Also, you will observe how MongoDB data changes after you run a TG bot command.

Goals:

1) Fork a project from GitLab.
2) Create "fat JAR" from a command line.
3) Run Spring Boot application from a command line (using `java` interpreter and using Maven).
4) Extend TG bot by creating new commands.
5) Share GitLab project with another developer.
6) Learn some software design patterns (Command and Factory method).

Steps:

1. Login to IBA GitLab at:

   ```
   https://code.iby.scdc.io/
   ```

2. Open **Java Workshop 2.0 Stream 1** group using Groups menu. Find the shared project `registration-at-iba-week-3`

3. Create a project fork using the **Fork** button at the top right corner, select your own account as the target namespace.

4. Start IntelliJ IDEA IDE.

5. Select menu: **File > New > Project from Version Control...**

6. Use the URL to import from as follows:

   ```
   git@code.iby.scdc.io:<YOUR_GITLAB_USER>/registration-at-iba-wee
   k-3.git
   ```

   NOTE, if you will use URL as original source project:

   ```
   git@code.iby.scdc.io:mzaikin/registration-at-iba-week-3.git
   ```

   You will be able to clone the project, but won't be able to push your code. Use **your own username** (namespace) in the URL.

7. Get familiar with the top level `pom.xml` file

   - the project still consists of 2 modules.
   - if needed, edit **`mwnw.cmd`** and add as the first line:

   ```
   SET JAVA_HOME=C:\Program Files\Eclipse
   Adoptium\jdk-17.0.2.8-hotspot\
   ```

   (use your path to Java 17)

8. The structure of the **`registration-common`** project has not changed.

   - Get familiar with `eu.ibagroup.common.mongo.collection.Session` class. It's a MongoDB **Document** class, and used to access the `sessions` collection in DB.
   - The `chatId` attribute is a unique number which identifies your Telegram client. It will be the same for all TG bots you use. So, it's your unique ID.
   - Open the `sessions` collection using MongoDB Compass utility and make sure it is empty.

9. Get familiar with **`registration-bot`** project:

   - The main class is `RegistrationBotApplication`. It starts the application which stays in the background and does not listen to any networking port. The TG bot only communicates to external TG servers. This makes it possible to host TG bot on any machine (e.g. at home) which is not reachable from outside.

- The bot is represented by the `RegistrationBot` class which extends `TelegramLongPollingBot` from Telegram API (you can check dependency in `pom.xml` file).
- Check the methods with `@Override` annotation in the `RegistrationBot` class, they used to communicate with the TG server.
- The bot is registered by the `eu.ibagroup.bot.telegram.Initializer` class. Check the `@EventListener` annotation. Also, you should know that two bot instances with the same API key (token) may not be registered at the same time. You must stop one instance first.
- The TG bot executes the logic by using the Command design pattern.
- Each supported command has its own `eu.ibagroup.bot.command.Command` instance.
- Each TG command must implement the `eu.ibagroup.bot.telegram.command.BotCommand` interface.
- Implementations of the interface can be found in the `eu.ibagroup.bot.telegram.command` package. You will need to create a new command there.
- Commands are obtained by TG bot from the factory class (`BotCommandFactory`) which is part of the Factory method design pattern.

10. Populate database name, user ID, and password to connect to MongoDB in `registration-common/src/main/resources/application-common.properties`

    - MongoDB database name and user ID are provided in the spreadsheet
    - MongoDB password will be shared in private message in Telegram

11. Populate the bot name and bot access token (API key) in `registration-bot/src/main/resources/application.properties`

    - You should have been provided them when registered bot in week #0
    - If you forgot your bot name and token, goto https://t.me/BotFather and look up

12. **ACTIVITY №1: Run Spring Boot application from command line**

    Run the project by using `java` interpreter from command line

    - Open command line shell
    - Goto `registration-at-iba-week-3` directory
    - Run command: `mvnw clean install`
    - Make sure no errors in console log
    - Change directory to `registration-bot/target`
    - Make sure the `registration-bot-1.0.0.jar` file exists
    - Inspect the file content by running this command (it contains application classes and all JAR dependencies):

```
jar -t -v -f registration-bot-1.0.0.jar
```

- Optionally check inside the JAR (you can use any unzip utility, or `jar -x -v -f XX`) file `META-INF/MANIFEST.MF` – it contains Spring bootstrap class as `Main-Class`, and Bot application class as `Start-Class`.
- Run the bot:

```
java -jar registration-bot-1.0.0.jar
```

- Open in web browser URL: `https://t.me/<BOT_USERNAME>`
- In Telegram client type `/start` command
- Check the `sessions` collection state in MongoDB using Compass utility, it must contain 1 document.
- Stop the application using **Ctrl +C**.
- Goto `registration-at-iba-week-3` directory (top directory of the project)
- You can run the TG bot application also using the Maven command (note since application multi-module, you need to provide module name explicitly, also POM must contain information about "main" class which runs by default):

```
mvnw clean install exec:java -pl registration-bot
```

- Stop the application (**Ctrl + C**).

13. **ACTIVITY №2: Fix the `/about` command**

- Open the class: `eu.ibagroup.bot.telegram.command.AboutBotCommand`. It is responsible to produce an output when user types `/about` command.
- Figure out the logical error.
- Fix the logical error (also you can try to make the command code efficient)
- Test the command's new code.

14. **ACTIVITY №3: Create the `/status` command**

- Create code which will respond to the `/status` command from the user.
- The command must produce information about current user `Session`:
a) if user confirmed (authenticated) – **yes** or **no**
b) user name (if the user not confirmed – type "**n/a**")
c) user email (if the user not confirmed – type "**n/a**")

NOTE 1: you do not have yet functionality in the code to authenticate the user, so the command will print "no", "n/a", "n/a". This is normal at the moment.

NOTE 2: if you want to fully test the functionality of this command, you can manually update `sessions` collection (`email, name, isConfirmed`) in MongoDB and re-run the command in the TG bot.

15. **ACTIVITY №4: Create the `/exit` command**

    - Create code which will respond to the `/exit` command from the user.
    - The command must delete the user's session from MongoDB and produce output: "**`All your data was deleted.`**"
    - Try this command in the bot and make sure in MongoDB Compass that the `sessions` collection is blank.

    NOTE: do not try to use the `/about` command to check the number of sessions, as it always runs after the session was created, so you will never see zero sessions in the output.

16. Stop the `registration-bot` application (if it running)

17. In IntelliJ IDEA open Git panel (click **Git** tab on the bottom left)

18. Open **Local Changes** tab, expand the "**Changes**" section and "**Unversioned Files**" section, and review the changes.

    Select all changes and unversioned files you worked on this week, and right click and select "**Commit Files…**".

    NOTE: do not commit locally generated files:
    `.gitignore` file,
    `.idea` folder
    `target` folders in each project
    `*.class` files

    Commit and push the files.

19. Open your project in GitLab UI at:

    ```
    https://code.iby.scdc.io/<YOUR_GITLAB_USER>/registration-at-iba
    -week-3
    ```

    Make sure your commit is visible in the web UI.

20. On the left side use menu **Settings > Members** to add instructor as **Reporter** role member to your week #3 project.

Self education activities (not optional), use Google, online articles, JavaDocs, sample code online, etc..

1. Get familiar with Command design pattern:

   https://en.wikipedia.org/wiki/Command_pattern

2. Get familiar with Factory method design pattern:

   https://en.wikipedia.org/wiki/Factory_method_pattern

3. MongoDB `@Document` annotation

4. `org.springframework.data.mongodb.repository.MongoRepository` interface (and it's parent interfaces)

5. MongoDB `@Query` annotation (e.g. see `EventRepository` for examples)

Once you complete all steps notify the instructor.

Any questions or problems: ask in the workshop group chat or in personal messages to the instructor.

Have fun!