

Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog <https://seblog.cs.uni-kassel.de/ws2324/programming-and-modelling/> zu berücksichtigen.

Abgabefrist ist der 08.02.2024 - 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst sowie für die Betreuer und Korrektoren sichtbar.

Für die Hausaufgabe benötigst du **kein neues** Repository. Es wird das gleiche Repository benutzt, das bereits in Hausaufgabe 10 angelegt wurde. Dieses kann über folgenden Link erstellt werden, falls nicht bereits geschehen:

https://classroom.github.com/a/VB_hZStE

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet!

Aufgabe 1 - Websockets implementieren (5P)

Als Vorbereitung auf die nachfolgenden Aufgaben muss das Projekt um ein paar Klassen erweitert werden. Darüber hinaus muss die `build.gradle` angepasst werden.

Gradle

Füge die folgenden Zeilen in die `build.gradle` in die dependencies ein:

```
implementation group: 'javax.websocket', name: 'javax.websocket-api', version: '1.1'
implementation group: 'org.glassfish.tyrus', name: 'tyrus-client', version: '1.15'
implementation group: 'org.glassfish.tyrus', name: 'tyrus-container-grizzly-client',
    version: '1.15'
```

ClientEndpoint

Kopiere die Klasse `de.uniks.pmws2324.nopm.ws.ClientEndpoint` aus den vorgegebenen Dateien¹.

MessageEventService

Erstelle die Klasse `de.uniks.pmws2324.nopm.ws.MessageEventService` mit folgendem Inhalt:

- **private** `ClientEndpoint endpoint;`
- **public** `MessageEventService(ClientEndpoint endpoint)`
Setzt den `endpoint`.
- **public** `Runnable subscribeTopic(String topic, Consumer<MessageDto> callback)`
Verbinde den `ClientEndpoint`. Dieser meldet sich für das Event `topics.<topic>.messages` an und gibt eingehende Messages an das `callback` weiter. Weiterhin gibt der `ClientEndpoint` ein `Runnable` zurück, welches beim Ausführen das Event wieder abmeldet.

Ergänze in deiner Klasse für Konstanten folgende Konstante:

- **public static final** `String WEBSOCKET_URL = "wss://nopm.uniks.de/ws/v1";`

¹<https://github.com/sekassel/pmws2324-files/blob/main/HA12/ClientEndpoint.java>

Aufgabe 2 - ChatService (5P)

Erweitere den [de.uniks.pmws2324.nopm.service.ChatService](#) mit der Methode:

```
public Runnable subscribeTopic(Topic topic)
```

Implementiere die Methode, indem du den [MessageEventService](#) im Konstruktor speicherst und [subscribeTopic](#) verwendest. Gib das [Runnable](#) unverändert zurück. Im Callback musst du zunächst das [MessageDto](#) in eine [Message](#) umwandeln und danach dem Topic hinzufügen.

Verändere die [sendMessage](#)-Methode, sodass die [Message](#) **nicht** mehr dem Topic hinzugefügt wird – dies wird nun durch den WebSocket realisiert.

Passe die [App](#) an, sodass der [MessageEventService](#) und der benötigte [ClientEndpoint](#) erstellt und an den [ChatService](#)-Konstruktor übergeben werden.

Aufgabe 3 - Controller anpassen (3P)

In dieser Aufgabe muss der `ChatController` angepasst werden.

Entferne zunächst den `Timer` und den `lastFetchTimestamp`. Erstelle stattdessen die Klassenvariable:

```
private Runnable cleanup;
```

Verwende in `init` die Methode `ChatService.subscribeTopic`, um dich für neue Nachrichten anzumelden und speichere den Rückgabewert in `cleanup`. Rufe `cleanup` in `destroy` auf.

Achte auf die Verwendung von `Platform.runLater()` im `PropertyChangeListener`, da Nachrichten nun nebenläufig zu einem Topic hinzugefügt werden können.

Aufgabe 4 - AppTest (10P)

Ersetze in deiner `build.gradle` diese Dependency:

```
implementation group: 'com.konghq', name: 'unirest-java', version: '3.14.5'
```

gegen die folgenden Einträge:

```
// https://mvnrepository.com/artifact/com.konghq/unirest-java
implementation group: 'com.konghq', name: 'unirest-java-core', version: '4.2.7'
implementation group: 'com.konghq', name: 'unirest-mocks', version: '4.2.7'
implementation group: 'com.konghq', name: 'unirest-objectmapper-jackson', version: '4.2.7'
```

Ändere außerdem den static block im `ChatApiService` zu folgendem Code ab:

```
static {
    Unirest.config().addDefaultHeader("Content-Type", "application/json");
    Unirest.config().setObjectMapper(new JacksonObjectMapper());
}
```

Füge nun außerdem die **Mockito-Dependency** in deiner `build.gradle` unter `dependencies` ein:

```
// https://mvnrepository.com/artifact/org.mockito/mockito-core
testImplementation group: 'org.mockito', name: 'mockito-core', version: '5.1.0'
```

Passe deine `App` und deinen `AppTest` wie in Vorlesung und Übung gezeigt an. Erstelle einen `ChatService` mit gemockten `ChatApiService` und `MessageEventService` und definiere deren Verhalten mittels `when` und `verify`. Der `AppTest` soll den gleichen Ablauf prüfen wie in Hausaufgabe 10, darf aber **nicht** mit dem Server kommunizieren.

Abgaben, deren `AppTest` eine Serververbindung voraussetzt, werden mit 0 Punkten bewertet.

Abschluss

Nach Implementierung der Funktionalität dieser Hausaufgabe ist die Anwendung NoPM für dieses Semester **abgeschlossen**. Weitere Features dürfen bei gegebener intrinsischer Motivation umgesetzt werden, sind jedoch nicht Pflicht. In diesem Fall sollte jedoch der Commit, welcher die Aufgabenstellung abschließt, deutlich markiert werden, damit eventuelle Erweiterungen nicht zu Abzügen führen.

Dies (Hausaufgabe 12) ist die letzte reguläre Hausaufgabe. In der kommenden Woche wird das Aufgabenblatt zum Projekt für Studierende, die 6CP erhalten, veröffentlicht.