Universität Trier
Professur für Softwaretechnik
Prof. Dr. Stephan Diehl
Lucas Kreber, M.Sc.

Fortgeschrittene Softwaretechnik
(WS 2024/25)
Zweites Übungsblatt

Gradle und CI/CD

Aufgabe 1: Gradle (4 Punkte)

Unter folgendem Link finden Sie den Programmcode für die Aufgabe.

```
https://gitlab.uni-trier.de/kreberl/fst-a1
```

In dem Code-Repository gibt es zwei Branches **fx** und **web**. Erstellen Sie aus beiden Branches ein gemeinsames Projekt (siehe unten). Clonen Sie das Code-Repository und verwenden Sie Gradle, um folgende Aufgaben zu erfüllen:

1. **Multi-Project-Struktur einrichten:**
   Passen Sie die Projektstruktur so an, dass ein Multi-Project-Build unterstützt wird. Das Projekt soll zwei Module umfassen:
   - **fx:** Enthält das JavaFX-Modul (startet die FX-Desktopanwendung)
   - **web:** Enthält das Web-Modul (startet die Spring Boot Webanwendung)
2. **Abhängigkeiten über Maven Central beziehen:**
   Alle Module sollen ihre Abhängigkeiten über das Repository *mavenCentral* beziehen.
3. **Java-Anwendungen festlegen:**
   Beide Module werden als Java-Anwendungen definiert.

4. **Abhängigkeiten identifizieren und automatisch auflösen:**
   Identifizieren Sie die benötigten Abhängigkeiten für jedes Modul und konfigurieren Sie Gradle so, dass diese automatisch aufgelöst werden. Benötigt werden:

   - **fx-Modul** (zum Testen):
     - `junit-jupiter:5.9.2`
   - **web-Modul:**
     - **Zum Bauen:**
       * `spring-boot-starter-web`
       * `spring-boot-starter-websocket`
       * `spring-boot-starter-thymeleaf`
     - **Zum Testen:**
       * `junit-jupiter:5.9.2`

5. **Build-Skripte für unabhängige Modulstarts erstellen:**
   Erstellen Sie Build-Skripte, um beide Module unabhängig voneinander starten zu können. Ein möglicher Aufruf könnte wie folgt aussehen:

   - **fx-Modul:** `:fx:run`
   - **web-Modul:** `:web:bootRun`

   Die Tasks `run` und `bootRun` werden durch hinzufügen von JavaFX und Spring Boot in Gradle automatisch zur Verfügung stehen.

Testen: Der Befehl `gradle :fx:run` im Projekt-Root startet ein Chat-Fenster als JavaFX-Anwendung (siehe README im Git-Repository). Mit dem Befehl `gradle :web:bootRun` wird eine Spring Boot-Anwendung gestartet. Anschließend kann über `localhost:8080/chat` direkt im Browser ein Chatfenster aufgerufen werden (siehe README im Git-Repository).

Abgabe:

- Das zusammengefügte Projekt inkl. aller Gradle Skripte, als .zip gepackt

Aufgabe 2: GitLab CI/CD (4 Punkte)

Erstellen Sie für das Projekt aus Aufgabe 1 eine CI/CD-Pipeline auf GitLab, angelehnt an das in der Vorlesung gezeigte Beispiel. Sie können entweder einen neuen Branch in Ihrem Projekt aus der Vorlesung erstellen oder ein neues Projekt anlegen.

Die Pipeline soll die folgenden Schritte ausführen:

1. **Testen**: Alle Testfälle der Anwendung sollen mit Hilfe von Gradle ausgeführt und überprüft werden.
2. **Build**: Erstellen Sie einen Docker-Container für die Anwendung und pushen Sie ihn anschließend auf DockerHub. Der Name des Containers sollte folgendem Format entsprechen: `lucaskreber/fst_uebung:fstchat-1.1-IhrName`. Verwenden Sie für den Login die in der Vorlesung bereitgestellten Zugangsdaten.
3. **Deployment**: Der Container soll auf einem Server (136.199.55.227) bereitgestellt werden. Nutzen Sie dazu SSH, um auf dem Server die Befehle `docker build` und `docker run` auszuführen.Achten Sie darauf, eventuell laufende Container auf dem gleichen Port vorher zu beenden. Verwenden Sie hier ebenfalls den SSH-Key aus der Vorlesung.
4. Die einzelnen Schritte sollen nacheinander ausgeführt werden. Verwenden Sie dazu **Stages** in Ihrer Pipeline.

Hinweis:

- Bitte verwenden Sie ausschließlich den Ihnen zugewiesenen Port, um die Docker-Container zu starten und zu beenden. Anders als in der Vorlesung, läuft diese Anwendung standardmäßig intern auf Port 8080.
- Sollten Sie nicht in der Vorlesung gewesen sein und die Übung dennoch gerne abgeben möchten, wählen Sie hier (`https://docs.google.com/spreadsheets/d/1cv8P1M9UYz_6unx2G-kw8pFKQXTUDN5pew9OvtTxkIk/edit?usp=sharing`) einen Port aus und löschen Sie diesen aus der Liste
- Verwenden Sie das Java JDK 21 zum Ausführen der Anwendung und Gradle 8.4, um Kompatibilitätsprobleme zu vermeiden.
- Für das Erstellen des Ziel-Dockercontainers (Docker-in-Docker) benötigen Sie Gradle. Sie können frei entscheiden, ob Sie Gradle in Ihrer Pipeline nachinstallieren oder einen bereits konfigurierten Docker-Container für den Build verwenden. (Erinnerung: auf `https://hub.docker.com/` finden Sie konfigurierte Docker-Container)

Sie haben die Aufgabe erfolgreich gelöst, wenn sich in Ihrem Browser unter der Adresse `http://136.199.55.227:<ihr_port>/chat` ein Chat-Fenster öffnet (nur im Netz der Uni verfügbar). Dort können Sie mit den Anderen chatten.

Abgabe:

- Ihr Pipeline-Skript aus Gitlab
- Ein Link in Ihr Repository

Abgabe: bis Donnerstag, 14.11.2024, 23:59 Uhr

Abgabeformat: Die Abgabe der Übungsblätter erfolgt über Stud.IP. Bitte beachten Sie die folgenden Hinweise. Abgaben, die nicht dem beschriebenen Format entsprechen, werden ignoriert und mit 0 Punkten bewertet:

- Bei **normalen Übungsblättern**:
  Die Abgabe erfolgt als **einzelne PDF-Datei** (keine Ordner, kein ZIP-Archiv, nicht in mehreren Teilen, keine anderen Dateiformate). **Bitte vermerken Sie in dieser PDF-Datei gut lesbar Ihren Namen und Ihre Matrikelnummer.** Verwenden Sie folgendes Schema für den Dateinamen der PDF-Datei: `fst24-uebxx-123456` (`xx` durch Nummer des entsprechenden Übungsblattes ersetzen, z.B. ueb01 für das erste Blatt, und `123456` durch Ihre Matrikelnummer ersetzen).

- Bei **Übungsblättern mit Programmieraufgaben**:
  Neben der evtl. vorhandenen PDF-Datei soll nur der Quellcode abgeben werden (also keine `class`-Dateien o.Ä.). **Bitte vermerken Sie in jeder Quellcodedatei Ihren Namen und Ihre Matrikelnummer als Kommentar.** Packen Sie in diesem Fall alle benötigten Dateien (Quellcode und ggf. PDF-Datei) als **ZIP-Datei** mit dem Namen `fst24-uebxx-123456.zip` (`xx` durch Nummer des entsprechenden Übungsblattes ersetzen, z.B. ueb01 für das erste Blatt, und `123456` durch Ihre Matrikelnummer ersetzen).