# Kumuluz : JEE Micro Runtime Framework

## 1. Überblick

KumuluzEE ist ein "lightweight open-source microservice framework" und stellt eine Alternative zum Betrieb von JEE-Anwendungen auf klassischen Applikationsservern dar. Der Serveranteil wie auch die Implementierungen der diversen EE-Bestandteile werden mit dem Anwendungscode zu einer kompakten Anwendung integriert. Eine separate Installation ist also nicht nötig. Vielmehr wird KumuluzEE in Form diverser Dependencies (aka Libraries) im Build der Anwendung einbezogen.

Dieses Dokument beschreibt im Folgenden die Grundzüge der Nutzung von KumuluzEE insbesondere im Hinblick auf die im Seminar verwendeten Demo- und Übungsprojekte und die darin genutzten Features von KumuluzEE. Auf der Projektseite https://ee.kumuluz.com/ finden ausführlichere Informationen dazu sowie diverse Tutorials und Links zu Beispielanwendungen.

## 2. Integration von KumuluzEE im Build einer Anwendung

KumuluzEE ist wie ein Baukasten aufgebaut. Neben einer stets benötigten Core-Komponente werden nur diejenigen Bausteine als Dependencies referenziert, die von der Anwendung benötigt werden. Um die Versionen bequem und konsistent zu verwalten, steht ein sog. BOM (Bill of Material) zur Verfügung. Wir nutzen im Seminar die Version 3.4.0. Damit sieht die Maven-Projektdefinition im betreffenden Ausschnitt so aus:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.kumuluz.ee</groupId>
      <artifactId>kumuluzee-bom</artifactId>
      <version>${kumuluzee.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
```

```
    </dependencyManagement>

    <dependencies>
      <dependency>
        <groupId>com.kumuluz.ee</groupId>
        <artifactId>kumuluzee-core</artifactId>
      </dependency>
      <dependency>
        <groupId>com.kumuluz.ee</groupId>
        <artifactId>kumuluzee-servlet-jetty</artifactId>
      </dependency>
      ...
```

Unsere Demo- und Übungsprojekte benötigen i. A. die folgenden KumuluzEE-Module:

- `kumuluzee-core`: Core-Modul; wird stets benötigt.
- `kumuluzee-servlet-jetty`: Servlet Container.
- `kumuluzee-cdi-weld`: CDI Container.
- `kumuluzee-jax-rs-jersey`: REST-Implementierung.
- `kumuluzee-jpa-hibernate`: JPA Provider.
- `kumuluzee-jta-narayana`: Transaktionmanager.
- `kumuluzee-jsp-jetty`: JSP-Implementierung; wird für JSF benötigt, selbst wenn nicht direkt genutzt.
- `kumuluzee-jsf-mojarra`: JSF-Implementierung.
- `kumuluzee-el-uel`: Unified Exlession Language; wird für JSF benötigt.

Zudem wird ein Datenbank-Treiber benötigt; wir nutzen die H2-Datenbank.

## 3. Anwendungsstart

Die Anwendung wird gestartet, indem die Klasse `com.kumuluz.ee.EeApplication` gestartet wird. Wir nutzen im Seminar kein sog. Uber JAR, bei dem alle Anwendungsteile in einem einzelnen JAR integriert wären. Stattdessen müssen sich die oben aufgeführten Dependencys und ihre transitiven Abhängigkeiten im Classpath befinden. Die Anwendungsklassen selbst dürfen nicht in ein JAR gepackt werden; KumuluzEE erwartet sie 'exploded' in einem Classpath-Verzeichnis (in Maven-Projekten i. d. R. `target/classes`).

Beim Start sollte die System Property `java.util.logging.manager` auf `org.apache.logging.log4j.jul.LogManager` gesetzt werden, wenn im Projekt zum Logging Log4j genutzt wird (was unsere Demo- und Übungsprojekte tun), damit auch die Log-Ausgaben von KumuluzEE selbst im korrekten Format ausgegeben werden (s. a. Logging weiter unten).

## 4. Besonderheiten unserer Demo- und Übungsprojekte

Unsere Seminarprojekte sind weitestgehend portabel aufgebaut, d. h. sie können zum Bau eines klassischen WAR-Files genutzt werden, das zum Betrieb auf einen JEE-Server deployt wird. Die KumuluzEE-Unterstützung ist daher in einem sog. Maven-Profil definiert, was zur Nutzung von KumuluzEE explizit aktiviert werden muss. Dazu dient die Maven-Option `-Pas_kumuluzee`. Die IDEs bieten dazu i. A. bequeme Auswahlmöglichkeiten, z. B. in Eclipe: Rechtsklick auf das Projekt im Project Explore → `Maven` → `Select Maven Profiles` → `as_kumuluzee`.

Die Aktivierung des Profils `as_kumuluzee` bewirkt folgendes:

- Die o. a. Dependencies werden deklariert.
- Die Projektverzeichnisse `src/kumuluzee/java` und `src/kumuluzee/resources` werden zu sog. Source Directories deklariert (wie die Standardverzeichnisse `src/main/java` und `src/main/resources`).

  In diesen Verzeichnissen befinden sich Java-Klassen und Konfigurationsdateien, die nur für den Betrieb mit KumuluzEE gültig sind. Dies sind i. W. Klassen, die das Verhalten von KumuluzEE an das einer Standard-JEE-Anwendung angleichen. Details finden Sie in den Kommentaren der betreffenden Quellen. Zudem enthalten die Verzeichnisse Konfigurationsdateien.
- KumuluzEE erwartet Web Resources (HTML-Files, Facelets etc.) im Classpath-Verzeichnis `webapp`, während in einer klassischen JEE-Anwendung dafür das oberste Verzeichnis des erzeugten WAR-Files vorgesehen ist. Die Quelldateien befinden sich in `src/main/webapp`. Im Profil `as_kumuluzee` wird der Build-Prozess daher so konfiguriert, dass die Dateien aus `src/main/webapp` nach `target\classes\webapp` kopiert werden.
- KumuluzEE erwartet die Anwendungsklassen 'exploded' im Classpath. Daher wird im Profil `as_kumuluzee` kein JAR- oder WAR-File gebaut.

## 5. Konfiguration der Anwendung

KumuluzEE kann verschieden konfiguriert werden, u. a. mit der Datei `config.yaml` im Verzeichnis `src/kumuluzee/resources`, die durch den Build in den Classpath kopiert wird. In unserem Seminarprojekten wird darin i. A. der Web-Kontext der Anwendung eingestellt (=Projektname) und ein oder mehrere Datasources definiert.

Als Logging-Framework nutzen wir Log4j, wobei auch andere APIs wie JUL und Slf4j genutzt werden können und in Log4j umgeleitet werden. Die Konfiguration von Log4j erfolgt mit der Datei `log4j2.xml` im Verzeichnis `src/kumuluzee/resources`.

## 6. Einschränkungen

KumuluzEE unterstützt nur einen Teil der JEE, i. W. das JEE Web Profile ergänzt um diverse Zusätze (Micro Profile, Event Streaming, …). Demzufolge sind nur bestimmte Seminarprojekte für KumuluzEE geeignet. Im Wesentlichen sind dies die Projekte wie `CDI_LABS`, `WS_LABS` oder `JSF_LABS`, die EE-Grund- und Aufbauseminare begleiten. Sie können die Projekte daran erkennen, dass sie in ihrer jeweiligen Projektdefinitionsdatei `pom.xml` die Property `enabled.as_kumuluzee` mit dem Wert `true` enthalten.