Technische Hochschule Ingolstadt

Dokumentation

MyMusic

Java EE Anwendung

Erstellt im Studienfach:

Implementierung von Informationssystemen

von

Michael Obermüller

Inhaltsverzeichnis

1 MyMusic

1.1 Fachliche Kurzbeschreibung

MyMusic ist eine Java EE Anwendung, welche zur Verwaltung einer Musiksammlung dient. Hierzu können Musikalben mit ihren Interpreten, Songs und Albumcovers angelegt werden. Die Musiksammlung ist anhand von Songtitel, Albumtitel und Interpret durchsuchbar. Die Suche bietet eine Autocomplete-Funktionalität, welche bei der Eingabe von mindestens 4 Zeichen erfolgt. Die Ergebnisse der Suche werden jeweils in einer Liste für Alben, Interpreten und Songs aufbereitet und dargestellt. Das Anlegen, Bearbeiten und Löschen von Alben ist nur als angemeldeter Benutzer möglich. Die Benutzerverwaltung zur Anlage von neuen Benutzern ist mithilfe des Admin Benutzers aufrufbar.

1.2 Demo Anwendung

Die Anwendung MyMusic ist unter der URL: **http://46.101.120.33:8080/mymusic/** im Internet erreichbar. In der nachfolgenden Tabelle sind bereits angelegte Testbenutzer der Demo Anwendung aufgelistet.

Benutzername	**Passwort**
Admin	admin
User	user

Zusätzliche Benutzer können mithilfe des Admin Benutzers unter dem Navigationsreiter: Benutzerverwaltung angelegt werden.

2 Technologien

Die technische Realisierung der MyMusic Applikation wurde mit den folgenden Technologien umgesetzt:

- JavaServer Faces (JSF)
- Contexts and Dependency Injection (CDI)
- Enterprise Java Beans (EJB)
- Java Persistence API (JPA)
- Webservices: JAX-WS (SOAP) und JAX-RS (REST)

Zudem wurden die folgenden Laufzeitumgebungen bzw. Testframeworks verwendet:

- Arquillian (Integrationstests)
- Mockito
- Wildfly 9.0.2
- Docker

3 Architektur

3.1 Überblick

Die Architektur ist in drei wesentlichen Schichten unterteil:

- **User Interface**, welche die Views der Webanwendung beinhaltet. Die Views sind .xhtml Dateien und nutzen ein zentrales Template (template.xhtml) für das Basislayout der Webseite. Zudem werden CDI Beans als Model und Controller für die Views verwendet.
- **Business Logik**, welche fünf Services beinhaltet. Dabei handelt es sich um einen zentralen Service für das Management der Alben, einen Service für Suchanfragen zu Alben, Interpreten und Songs und einen Service für die Benutzerverwaltung. Des Weiteren sind jeweils ein Service für die Rest-Schnittstelle und ein Service für die SOAP-Schnittstelle in der Business Logik Schicht enthalten.
- **Persistence**, welche einen zentralen CrudService für das Anlegen, Bearbeiten, Löschen und Abfragen von Entity-Objekten beinhaltet.

3.2 Komponenten und deren Beziehung

Die Applikation besteht aus den wie zuvor beschriebenen drei wesentlichen Schichten: User Interface, Business Logik und Persistence. Eine detaillierte Betrachtung der Komponenten und deren Beziehung untereinander wird in Abbildung 1 dargestellt.

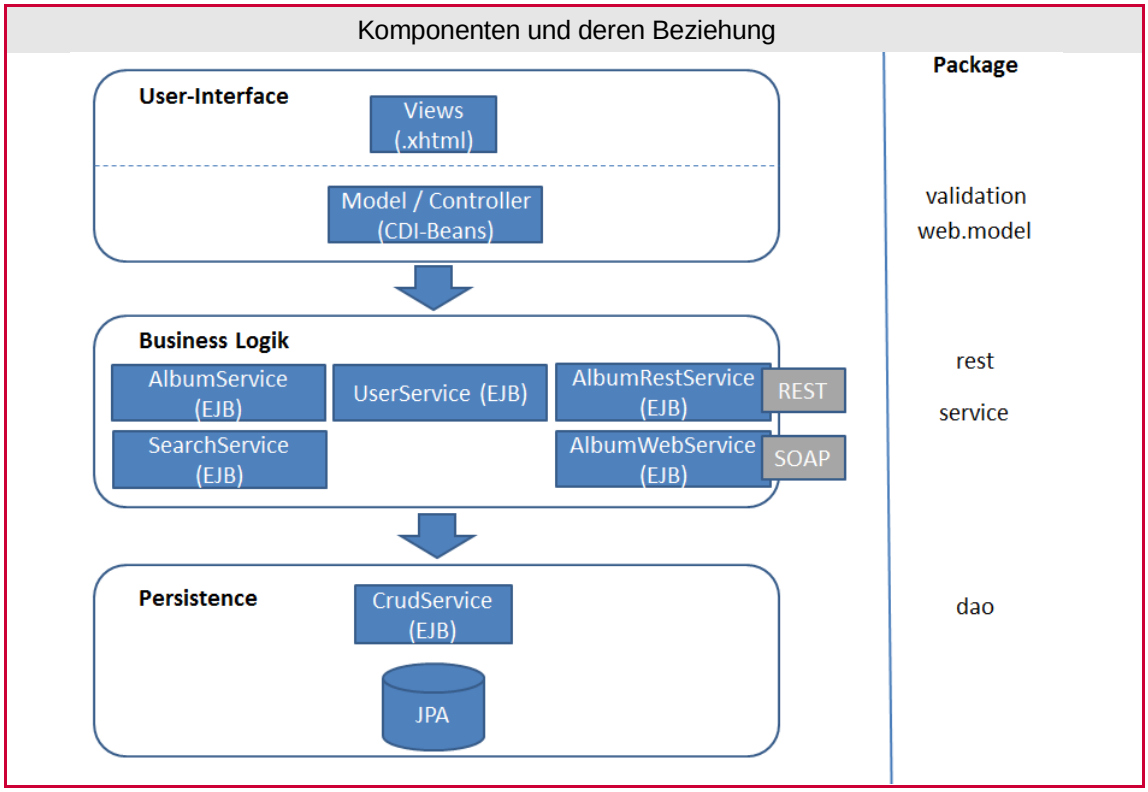

Abbildung 1: MyMusic Komponenten und deren Beziehung

Zudem zeigt Abbildung 2 die Package Struktur der MyMusic Applikation.

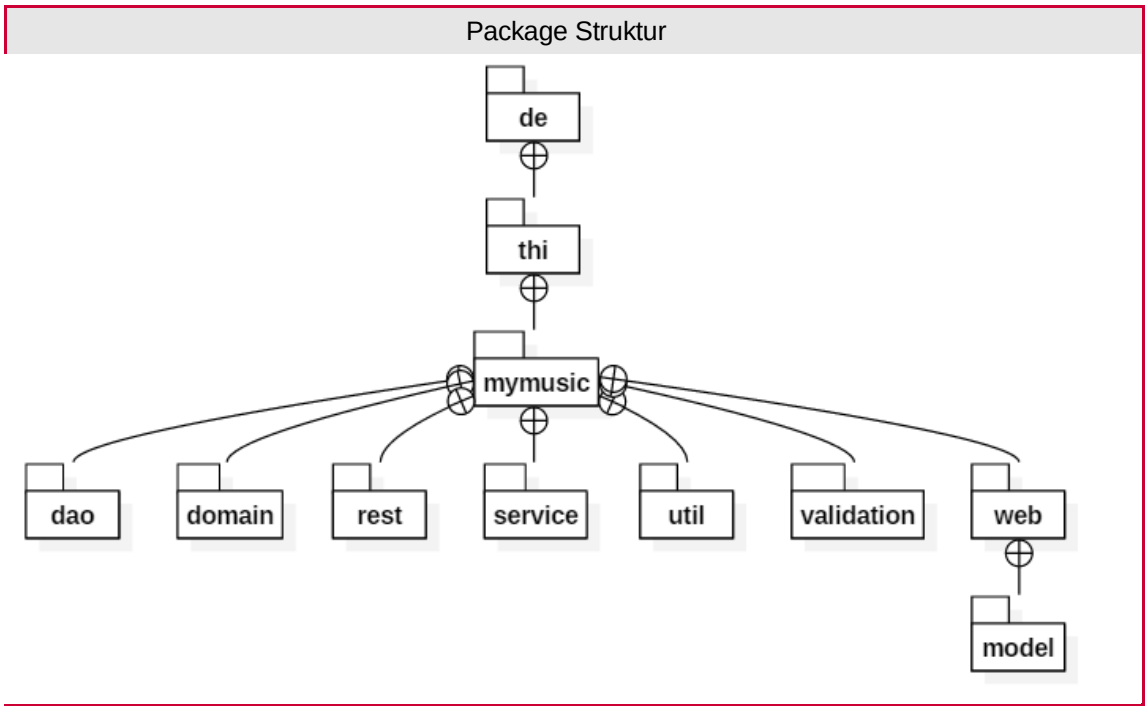

Abbildung 2: MyMusic Package Struktur

4 Fachliches Datenmodell

In der nachfolgenden Abbildung 3 wird das fachliche Datenmodell als Entity-Relationship Diagramm dargestellt.

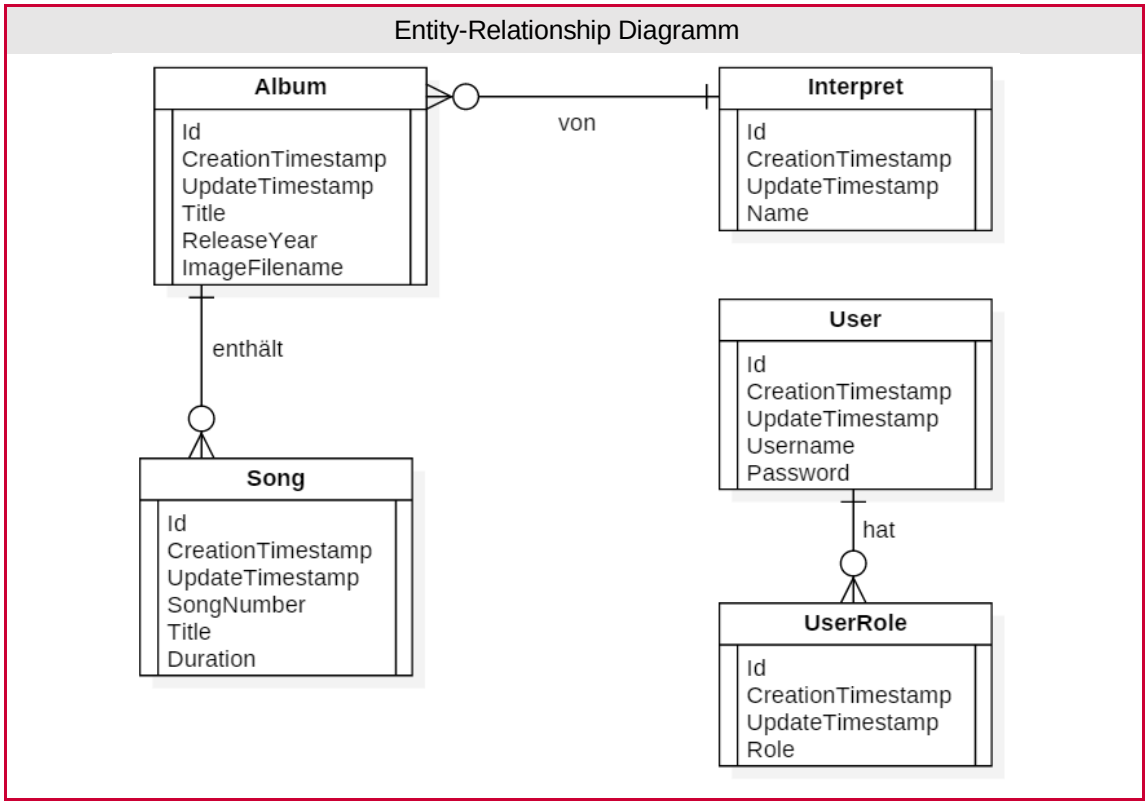

Abbildung 3: MyMusic Entity-Relationship Diagramm

5 Schnittstellen

Für die Abfrage der Musiksammlung stehen eine REST- und eine SOAP-Schnittstelle zur Verfügung.

5.1 Rest-Schnittstelle

Die Rest-Schnittstelle wird über die URL: http://46.101.120.33:8080/mymusic/api/ aufgerufen. Es stehen zwei Schnittstellen zur Verfügung:

- Auflistung aller Alben über *albums/*
- Abfrage eines einzelnen Albums mit einer Id über *albums/$id/*

5.2 SOAP-Schnittstelle

Unter der URL: http://46.101.120.33:8080/mymusic/AlbumWebService?wsdl ist die Beschreibung der SOAP-Schnittstelle (WSDL) erreichbar. Die SOAP-Schnittstelle ermöglicht die Abfrage eines Albums und die Auflistung aller Alben der Musik-Bibliothek.

6 Installation mit Docker

6.1 PostgresSQL

Build Image

Im Verzeichnis src/main/docker/postgressql ist das Dockerfile zur Erstellung des Postgressql-Images zu finden. Befehl zur Erzeugung des Images:

docker build --rm=true -t michaelobermueller/mymusic_postgressql:9.4 .

Run Image

Befehl für das erstmalige Starten des Containers:

docker run -i -t -p 5432:5432 --name pg_music michaelobermueller/mymusic_postgressql:9.4

Mithilfe des vergebenen Container-Namen pg_music kann der Container einfacht gestoppt und wieder gestartet werden:

docker stop pg_music

docker start pg_music

6.2 Filecontainer

Build Image

Im Verzeichnis src/main/docker/filecontainer ist das Dockerfile zur Erstellung eines Datencontainers der zur Ablage der Bilddateien dient.

Befehl zur Erzeugung des Images:

docker build --rm=true -t michaelobermueller/filecontainer .

Run Image

Erstmaliges Starten des Containers:

docker run -d --name imagecontainer michaelobermueller/filecontainer tail -f /dev/null

Anschließendes Starten und Stoppen des Containers:

docker stop imagecontainer

docker start imagecontainer

6.3 Wildfly

Build Image

docker build --rm=true -t michaelobermueller/mymusic:0.1 .

Run Image

docker run -i -t --volumes-from imagecontainer -p 8080:8080 -p 9990:9990 --name mymusic michaelobermueller/mymusic:0.1

Anschließendes Starten und Stoppen des Containers:

docker stop mymusic

docker start mymusic