# Service Architecture PAC Conference

# Verwendete Technologien

- Es werden entsprechend dem Kundenwunsch JBoss 7.1.1, JSF 2.2 (TODO version) und JPA verwendet.

- Als JPA Implementierung wird Hibernate 2.0 als etablierter Standard gewählt

TODO arquillian

# Umgebung

Als Umgebung wird von einem JBoss version 7.1.1 ausgegangen.

## Datasources

Es wird eine Datasource java:jboss/datasources/ConferenceDS benötigt

## Benutzer

Es wird ein Benutzer im ApplicationRealm mit der Rolle conferenceBackoffice benötigt.

## Queues

Es wird eine QueueConnectionFactory mit dem Namen „ConnectionFactory“ benötigt.

Es wird eine Queue mit dem JNDI Namen „queue/test“ benötigt.

# Aufbau der Quellen im Repository

Das Repository "PAC" gliedert sich in drei Unterordner:

## conference

Dies ist der Wurzelknoten des Java Workspace

Die einzelnen Java-Projekte sind als Maven Multi Module Projekte organisiert:

conference Oberprojekt

conference-base Querschnittliche Funktionen

conference-room Oberprojekt der Room-Komponente

conference-room-api RoomService API und Entität

conference-room-impl Implementierung des RoomService

conference-speaker Oberprojekt der Speaker-Komponente

conference-speaker-api SpeakerService API und Entität

conference-speaker-impl Implementierung des RoomService

conference-conference Oberprojekt der Conference-Komponente

conference-conference-api Conference API und Entitäten

conference-conference-impl Implementierung Conference

conference-talk Oberprojekt der Talk-Komponente

conference-talk-api TalkService API und Talk Entität

conference-talk-impl Implementierung TalkService und TalkSpeakerAssignment Entität

conference-web JSF GUI

conference-rest REST Services

## conference-base

Dieses Projekt beinhaltet querschnittliche Funktionen.

* persistence.xml
  + Da alle Serviceprojekte letztlich auf die gleiche DB zugreifen und die Tabellen teilweise per FK miteinander verbunden sind, wird nur eine querschnittliche persistence.xml verwendet.
* Producer für Logger, QueueConnection, InitialContext und EntityManager
* PerformanceLoggingInterceptor
* ServiceLoggingInterceptor

## conference-room

Implementierung des RoomService und dessen Entität Room.

Keine Beziehungen zu anderen Services. So wird sichergestellt, dass Rooms im Objektgeflecht ein „Blatt“ bleiben und keine unnötigen Zirkulärbeziehungen entstehen

## conference-speaker

Implementierung des SpeakerService und dessen Entität Speaker.

Keine Beziehungen zu anderen Services. So wird sichergestellt, dass Speaker im Objektgeflecht ein „Blatt“ bleiben und keine unnötigen Zirkulärbeziehungen entstehen

## conference-conference

Beinhaltet die Implementierungen von ConferenceService sowie der Conference Entität.

Keine Beziehungen zu anderen Services. So wird sichergestellt, dass Rooms im Objektgeflecht ein „Blatt“ bleiben und keine unnötigen Zirkulärbeziehungen entstehen

## conference-talk

Beinhaltet die Implementierungen von TalkService sowie der Talk Entität.

Diese Komponente hat Abhängigkeiten zu conference-speaker-api, conference-room-api und conference-conference-api

Die n:m Zuordnung zwischen Talk und Speaker wird über die Hilfsentität TalkSpeakerAssignment abgebildet. Diese ist in conference-talk-impl abgelegt und somit nicht Teil des conference-talk-api jars, welches diese Entität ansonsten exponieren würde.

## conference-web

TODO

## conference-rest

TODO

## doc

Dokumentation. Enthält dieses Dokument

## soapui

Enthält ein zugehöriges SoapUi Projekt zum Testen der Rest Services

## Sql

Enthält SQL DDL Skripte zum Anlegen der Tabellen, Testdaten und Benutzer

# Standard für Coding

* Es wird der Standard-Codeformatter von Eclipse Kepler WTP verwendet.
* Codesprache ist Englisch
* Methodenparameter sind minimal zu halten. Wenn statt einer Entität dessen id ausreicht, dann wird auch nur nach einer id verlangt.

## Namenstandards

## Projekte

Einzelne Komponenten sind jeweils in einem eigenen Maven Modul untergebracht, welches sich wiederum in weitere Untermodule mit öffentlichem (API) und privatem (Implementierung) Teil gliedern.

Das API Modul ended jeweils auf –api, die Implementierung auf -impl

Die Module sind jeweils entsprechend ihrem vollen Namen/Funktion separiert mit Minus benannt.

Also z.B: Das Modul conference-talk beinhaltet conference-talk-api und conference-talk-impl

## Packages

## Interfaces

Namenstandard: Es wird ein gewöhnlicher fachlicher Klassenname verwendet, kein „I-Präfix“

Z.B. „RoomService“

## Services

Die Implementierung von Services ended auf –Impl und beginnt mit dem Namen des Interfaces.

Alternative Implementierungen können durch einen Einschub zwischen diesen beiden Namenskomponenten differenziert werden

Z.B. “RoomServiceImpl”, “RoomServiceBlaBlaBlubImpl”

Namensstandard für lesende Services: get???()

Namensstandard für schreibende Services: save??? (), create??? (), delete??? ()

## Klassen

# Exceptions

Es werden unchecked Exceptions verwendet.

Für technische Probleme stehen durch Java bereits verschiedene RuntimeExceptions zur Verfügung; falls nötig wird in RuntimeException gewrappt und weitergeworfen.

Für fachlich gültige Fehlersituationen gibt es die unchecked Exception „BusinessException“, bei Bedarf auch als Oberklasse. Durch die Verwendung dieser Exception oder einer Ableitung davon werden fachliche Exceptions für das Monitoring kenntlich gemacht.

Namensstandard: Der Name von Exceptions endet mit „…Exception“