

**Fachhochschule Bielefeld   
Studienort Bielefeld**

**Studiengang Ingenieurinformatik**

**Stundenplan Applikation**

Projekt

Winter- / Sommersemester 2020/2021

vorgelegt von: Topp, Manuel

Matrikelnummer: 1029598

vorgelegt von: Demirci, Anil

Matrikelnummer: 1000007

Erstprüfer/in: Prof. Grünwoldt

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc73985272)

[2 Ziel des Projekts 4](#_Toc73985273)

[3 Aufbau der Datenbank 5](#_Toc73985274)

[4 Vorbereitung zum Projekt 9](#_Toc73985275)

[4.1 Verbindung zur Datenbank 9](#_Toc73985276)

[4.2 Erstellung eines Maven-Projekts 14](#_Toc73985277)

[4.3 Einstellungen des Projekts 16](#_Toc73985278)

[4.4 Erstellen der beans.xml 19](#_Toc73985279)

[4.5 Erforderliche Libraries 20](#_Toc73985280)

[4.6 Erstellung der Entities 22](#_Toc73985281)

[5 Javadoc und Internationalisierung 24](#_Toc73985282)

[5.1 Javadoc 24](#_Toc73985283)

[5.2 Internationalisierung 26](#_Toc73985284)

[6 Master Template und Use Cases 29](#_Toc73985285)

[7 Verwendung von EJBs 32](#_Toc73985286)

[8 Controller 33](#_Toc73985287)

[8.1 Entity Controller 33](#_Toc73985288)

[8.2 ModulController 37](#_Toc73985289)

[8.3 RegisterController 38](#_Toc73985290)

[8.4 LoginController 40](#_Toc73985291)

[8.5 ScheduleControllerCopy 42](#_Toc73985292)

[8.6 ScheduleControllerProf 44](#_Toc73985293)

[8.7 ScheduleController 46](#_Toc73985294)

[9 Vergleiche zwischen Eclipse und Netbeans 54](#_Toc73985295)

[10 Vergleichen und Testen der Server 56](#_Toc73985296)

[11 Fazit 60](#_Toc73985297)

[12 Anhang 61](#_Toc73985298)

[12.1 Konfiguration des WildFly Servers 61](#_Toc73985299)

[12.2 Installation des Glassfish Servers in Eclipse 65](#_Toc73985300)

[12.3 Einrichtung GlassFish Server 67](#_Toc73985301)

## 1 Einleitung

Im Rahmen der Projektarbeit soll eine Stundenplan Applikation für die Fachhochschule Bielefeld konzipiert und implementiert werden.

Die Fachhochschule Bielefeld stellt derzeit ihre Stundenpläne als PDF zum Herunterladen zur Verfügung. Viele der Stundenpläne sind zusammengelegt worden, sodass es ein wenig unübersichtlich für die einzelnen Studiengänge ist. Da in diesem Stundenplan viele Kürzel beinhalten, muss der Anwender diese im Anhang nachschlagen.

Das Programm, dass die FH-Bielefeld für die Erstellung der Stundenpläne benutzt, heißt S-Plus. Es gibt jeweils eine Datei einmal für das Sommersemester und Wintersemester, worin sich dann die Veranstaltungen verschieben lassen können. Für die neuen Stundenpläne werden die vorherige Version als Vorlage benutzt. Das Programm beinhaltet eine Konfliktbehandlung, sodass es nicht zu Überschneidungen bei den Veranstaltungen kommen kann. Die Kriterien hierfür sind die Räume, Dozenten, Module, Hilfskräfte und die Bürotage. Das Pflegen der Portfolios wird in dem Programm vorgenommen und deshalb ist es ein wenig unübersichtlich. Die erstellten Stundenpläne müssen einzeln erstellt, danach als PDF exportiert und am ende auf der FH-Homepage hochgeladen werden.

In der Zukunft soll die entwickelte Stundenplan Applikation die Stundenpläne online zur Verfügung stellen, damit jeder sich diese anschauen und herunterladen kann. Hinzu kommt, dass es zusätzliche Informationen aufrufen kann, wenn der Anwender auf einen Event klickt. Dies soll dabei helfen, den Stundenplan übersichtlicher zu gestalten und das Nachschlagen fehlender Informationen zu ersparen. Diese wird auf dem Stratos-Server immer online und aufrufbar sein. Aus der Sicht des Planers genügt es die Stundenpläne in der Applikation zu erstellen, da diese sofort online zur Verfügung stehen. Es wird gewährleistet, dass man nicht mehr alle Stundenpläne einzeln exportiert und auf der FH-Homepage hochgeladen werden muss.

## 2 Ziel des Projekts

Das Ziel des Projekts ist eine neue Stundenplan-Applikation mit einer Datenbank Verbindung zu entwickeln, welches den Benutzern und Administratoren erlaubt, über eine Weboberfläche diese Daten zu verwalten. Die Weboberfläche soll den Anwendern erleichtern mit den Daten zu arbeiten oder die gewünschten Informationen aufzurufen. Nach der Fertigstellung des Projekts soll die Stundenplan-Applikation in der Fachhochschule Bielefeld zum Einsatz kommen, damit die alte Version ersetzt wird. Gleichzeitig soll auch die Entwicklungsumgebung Eclipse genutzt werden, um die bisherige Erfahrung mit Netbeans zu vergleichen und am Ende eine Empfehlung zu geben.   
Das Programm wird zusätzlich mit zwei unterschiedlichen Servern ausgeführt, diese sind der WildFly-und der GlassFish-Server. Am Ende des Projekts werden die Server auf die Geschwindigkeit und die Funktionalität getestet und verglichen.   
Die empfohlene Entwicklungsumgebung und der empfohlene Server wird in zukünftigen Sommersemestern im Modul Datenbankanwendungen von den Studenten genutzt.

Folgende Fragen ergeben sich aus der Projekt Vorbesprechung:

* Sind die Stundenpläne anschließend inhaltlich gleichwertig?
* Werden bei der Nutzung des neuen Systems bisher nicht entdeckte Bugs gefunden?
* Gibt es eine Effizienzsteigerung?
* Welche der beiden Server ist am sinnvollsten zu nutzen?

Durch das Bearbeiten des Projekts sollen die Fähigkeiten, die bisher im Studium erlernt wurden wie Konzipierung der Datenbank und Programmierkenntnisse erweitert werden.

## 3 Aufbau der Datenbank

Für die erstellte Datenbank wird das Open Source Programmpaket XAMPP verwendet. Die Software ermöglicht das Installieren und Konfigurieren von einem Apache Webserver mit einer MySQL Datenbank, auch MariaDB genannt. Die Webanwendung zur Verwaltung und Administration erfolgt durch phpMyadmin für die MySQL-Datenbank. Die Datenbank heißt stundenplanung, welche 14 Tabellen beinhaltet, diese sind „accounts“, „benutzergruppe“, „dozenten“, „faculty“, „lehrveranstaltungsart“, „location“, „modul“, „pruefcodes“, „raum“, „sgmodul“, „studiengang“, „stundenplaneintrag“, „stundenplansemester“ und „stundenplanstatus“. Die meisten der Tabellen waren im Vorfeld schon gegeben, da das Thema schon in dem Modul „Datenbanken“ behandelt worden war. Die angegebenen Tabellen stehen in der dritten Normalform zu einander. Im Folgenden werden die einzelnen Tabellen und die Beziehungen der verschieden Tabellen untereinander beschrieben. Jegliche Beziehungen dieser verbundenen Tabellen sind 1:n Beziehungen, was bedeutet, dass ein Element mehreren Elementen der verbunden Tabelle zugeordnet sein kann. Was das explizit bedeutet wird bei der entsprechenden Tabellenbeschreibung erläutert.

In der „benutzergruppe“ Tabelle werden alle Rechte aufgelistet, die für den Benutzer zugeteilt werden. Jeder Benutzer hat bestimmte Rechte, die sich als Administrator, Raumzeitplaner, Studiengangsleiter und Nobody/Student unterscheiden, um später bei der Applikation Aktionen durchführen zu können. Diese Tabelle ist mit der „accounts“-Tabelle mittels einer 1:n Beziehung verknüpft, das bedeutet, dass eine Benutzergruppe ein oder mehrere Accounts haben kann. Die Tabelle „benutzergruppe“ hat vier Spalten. Diese sind „GroupID“ mit dem Datentyp tinyint(4), „BGName“ mit dem Datentyp varchar(50), BGShortName mit dem Datentyp varchar(5) und BGRechte mit dem Datentyp tinyint(4). Der Primärschlüssel ist die Spalte „GroupID“. Jeder Primärschlüssel hat die Extrafunktion Auto\_Increment, was bedeutet, wenn neue Datensätze erstellt werden, wir die ID um eins Inkrementiert.

Die „accounts“ Tabelle ist dafür da, dass die Benutzer der Applikation sich mit ihrem registrierten Account in der Webapplikation anmelden können. Falls dieser nicht existiert, kann der Benutzer sich einen Account erstellen in dem er sich Registriert. Zusätzlich werden auch die Tabellen „benutzergruppe“ und „faculty“ mit dem jeweiligen Fremdschlüssel verbunden. Dadurch wird dem Benutzer wie bereits beschrieben seine Rechte und den Fachbereich, zu dem er gehört zugeteilt. Die „accounts“-Tabelle besteht aus sechs Spalten, welche „AccID“ mit dem Datentyp int(6), „AccName“ mit dem Datentyp varchar(50), „AccPwd“ mit dem Datentyp varchar(20), „AccEmail“ mit dem Datentyp varchar(100), „FK\_GroupID“ mit dem Datentyp tinyint(4) und „FK\_FBID“ mit dem Datentyp int(5) sind. Der Primärschlüssel in dieser Tabelle ist die Spalte „AccID“ und die Fremdschlüssel sind „FK\_GroupID“ für die Benutzergruppe und „FK\_FBID“ für den Fachbereich. Der Unterschied zwischen Primärschlüsseln und Fremdschlüsseln sind, dass es nur ein Primärschlüssel geben kann und mehrere Fremdschlüssel. Diese Fremdschlüssel werden für die Beziehungen verwendet, sodass die Verbindung immer zwischen einem Primärschlüssel und einem Fremdschlüssel stattfinden kann. Dort sind alle Account-Namen, die einzigartig sind und die Accountdetails aufgelistet.

Die Fachhochschule Bielefeld hat verschiedene Fachbereiche und um die Stundenpläne nach den Fachbereichen zu unterscheiden, werden alle Fachbereiche in der „faculty“ Tabelle gepflegt. Die Tabelle „faculty“ beinhaltet drei Spalten, diese enthalten „FBID“ mit dem Datentyp int(5), „FacName“ mit dem Datentyp varchar(256) und „FacShortName“ mit dem Datentyp varchar(20). „FBID“ ist der Primärschlüssel in dieser Tabelle. Die „faculty“-Tabelle ist mit zwei Tabellen verknüpft, diese sind die „accounts“-Tabelle und die „studiengang“-Tabelle. Das heißt, dass der Primärschlüssel einmal mit dem Fremdschlüssel „FK\_FBID“ in der „accounts“-Tabelle und die „studiengang“-Tabelle verbunden ist. Sodass ein Fachbereich mehrere Studiengänge zugeordnet werden kann.

Darauffolgend hat jeder Fachbereich verschiedene Studiengänge, wo jeder Studiengang einen eigenen Stundenplan auflistet. Die „studiengang“-Tabelle enthält fünf Spalten. Diese sind „SGID“ mit dem Datentyp int(5), SGName mit dem Datentyp varchar(30), „SGKurz“ mit dem Datentyp varchar(30), „Semester“ mit dem Datentyp int(1) und „FK\_FBID“ mit dem Datentyp int(5). Der Primärschlüssel in der Tabelle liegt auf der Spalte „SGID“ und ein Fremdschlüssel auf „FK\_FBID“. Die Beziehungen dieser Tabelle sehen folgendermaßen aus, dass ein Studiengang mehrere Studiengangs-Module haben kann und ein Studiengang mehrere Prüfcodes enthält. Es wurde eine Zwischentabelle erstellt umso eine n:m Beziehung aufzulösen, das heißt, dass durch diese Zwischentabelle Datensätze erschaffen werden können die mehrere Abhängigkeiten haben. In diesem Fall von Modulen und Studiengänge. Da sonst ein Modul mehreren Studiengänge und andersherum zugeordnet werden könnte. Hier stehen die Studiengänge mit den jeweiligen Semestern.

Die „sgmodul“-Tabelle ist abhängig von den Tabellen „studiengang“, „dozenten“ und „modul“. Das bedeutet, dass einem Studiengangs-Modul ein Modul, Studiengang und Dozent zugewiesen kann. Hierbei wird erkannt, welche Module zu welchen Studiengänge und Dozenten gehören. Diese Tabelle enthält sechs Spalten, welche „SGMID“ mit dem Datentyp int(11), „FK\_ModID“ mit dem Datentyp int (11), „FK\_SGID“ mit dem Datentyp int(5), „FK\_DID“ mit dem Datentyp int(5), „ModSemester“ mit dem Datentyp int(1) und „SGMNotiz“ mit dem Datentyp varchar(256). Die Spalte „SGMID“ ist der Primärschlüssel und umfasst drei Fremdschlüssel welche auf „FK\_ModID“, „FK\_SGID“ und „FK\_DID“ liegen. Die Beziehung der „sgmodul“-Tabelle ist verbunden mit der „stundenplaneintrag“-Tabelle, das heißt, dass ein Eintrag in einem Stundenplan einem Modul mit Studiengang und Professor zugeteilt werden kann.

Alle Lehrbeauftragte der Fachhochschule Bielefeld werden in der „dozenten“ Tabelle aufgelistet. Die Tabelle „dozenten“ besteht aus fünf Spalten. Diese sind „DID“ mit dem Datentyp int(5), „DName“ mit dem Datentyp varchar(60), „DVorname“ mit dem Datentyp varchar(60), „DTitel“ mit dem Datentyp varchar(40) und DKurz mit dem Datentyp varchar(6). „DID“ ist der Primärschlüssel dieser Tabelle. In dieser Tabelle sind die Dozenten mit ihren Titeln aufgelistet.

In der „modul“ Tabelle sind alle Module verzeichnet und beinhaltet vier Spalten, diese sind „ModID“ mit dem Datentyp int(11), „ModName“ mit dem Datentyp varchar(50), „ModKuerzel“ mit dem Datentyp varchar(50) und „PCID“. Die Spalte „ModID“ hat extra noch einen Primärschlüssel und „PCID“ einen Fremdschlüssel von der „pruefcode“ Tabelle, da jedes Modul ein Prüfcode besitzt.

Für die Prüfcodes wurde eine eigene Tabelle angelegt und bezieht alle Prüfcodes für das jeweilige Modul vom Prüfungsamt mit ein. Die „pruefcodes“ Tabelle besitzt fünf Spalten. Diese sind „PCID“ mit dem Datentyp int(11), „PrCode“ mit dem Datentyp int(11), „FK\_SgID“ mit dem Datentypint(5), „VertiefungsrichtungShortName“ mit dem Datentyp varchar(10), „PflichtOderWahl“ mit dem Datentyp varchar(15). Die Spalte „PCID“ hat zusätzlich noch den Primärschlüssel und „FK\_SgID“ einen Fremdschlüssel. Hier kann ein Prüfcode mehrere Module zugeordnet werden.

In der „stundenplanstatus“-Tabelle sind alle Status aufgelistet, die ein einzelner Stundenplan annehmen kann. Hierzu zum Beispiel gehören Früher Vorschlag, In Diskussion, Veröffentlicht, Veraltet oder Zukünftige Planung. Diese Tabelle enthält vier Spalten, welche „SPSTID“ mit dem Datentyp int(11), „SPSTBezeichnung“ mit dem Datentyp varchar(256), „SPSTHint“ mit dem Datentyp varchar(256) und „PColor“ mit dem Datentyp varchar(7). Diese Tabelle ist mit der „stundenplansemester“-Tabelle verbunden, sodass ein Stundenplan ein oder mehrere Status annehmen kann. Der Primärschlüssel befindet sich auf der Spalte „SPSTID“.

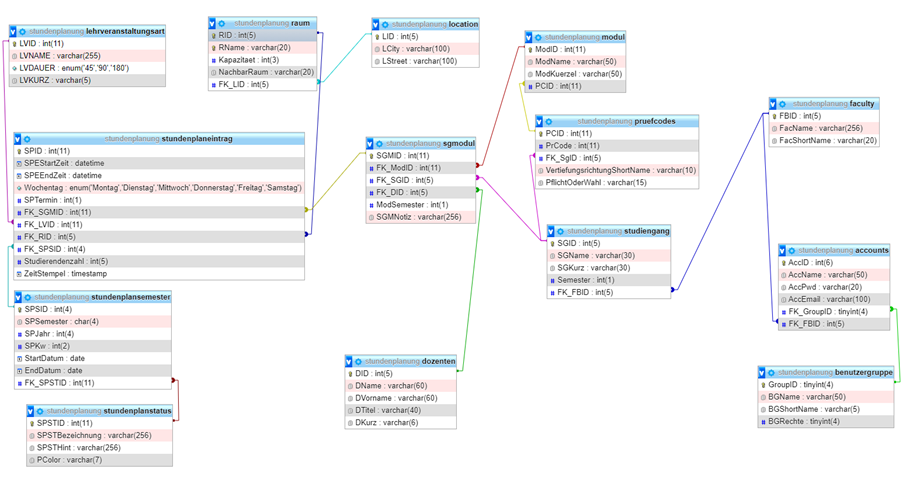
Die „stundenplansemester“-Tabelle stehen die Stundenpläne mit dem Anfangs- und Enddatum. Dazu kommen die Kalenderwochen, wie lange ein Stundenplan aktiv ist. In welchem Jahr der Stundenplan existiert und ob dieser ein Winter- oder Sommersemester ist. Diese Tabelle beinhaltet sieben Spalten, diese sind „SPSID“ mit dem Datentyp int(4), „SPSemester“ mit dem Datentyp char(4), „SPJahr“ mit dem Datentyp int (2), „SPKw“ mit dem Datentyp int(2), „StartDatum“ mit dem Datentyp date, „EndDatum“ mit dem Datentyp date und „FK\_SPSTID“ mit dem Datentyp int(11). „SPSID“ hat zusätzlich einen Primärschlüssel und „FK\_SPSTID“ einen Fremdschlüssel. Diese Tabelle hat eine Beziehung zu der „stundenplaneintrag“-Tabelle, somit wird einem Stundenplaneintrag einem Stundenplan zugeordnet.

Die Tabelle „location“ beinhaltet die Standorte der einzelnen Fachhochschulen mit den Städten und Straßen. Diese Tabelle enthält drei Spalten einmal die „LID“ mit dem Datentyp int(5), „LCity“ mit dem Datentyp varchar(100) und „LStreet“ mit dem Datentyp varchar(100). „LID“ ist der Primärschlüssel und hat eine Beziehung zu der „raum“-Tabelle. Somit kann ein Standort mehrere Räume haben und man kann die Räume den jeweiligen Standorten zuordnen.

Die Veranstaltungen müssen in den Räumlichkeiten, die zur Fachhochschule Bielefeld gehören stattfinden. Jeder Standort hat verschiedene Räume mit verschiedenen Kapazitäten und diese Räume sind in der Tabelle „raum“ aufgelistet. Diese Tabelle hat fünf Spalten, welche „RID“ mit dem Datentyp int(5), „RName“ mit dem Datentyp varchar(20), „Kapazitaet“ mit dem Datentyp int(3), „NachbarRaum“ mit dem Datentyp varchar(20) und „FK\_LID“ mit dem Datentyp int(5). Der Primärschlüssel liegt auf der Spalte „RID“ und der Fremdschlüssel auf „FK\_LID“ für die „location“ Tabelle. Diese hat eine Verbindung zu der „stundenplaneintrag“-Tabelle, wodurch ein Eintrag in dem Stundenplan den Räumen zugeordnet wird.

Die „lehrveranstaltungsart“-Tabelle enthält die drei Lehrveranstaltungsformen wie Vorlesung, Seminar und Praktikum. Darüber hinaus noch die Dauer der einzelnen Lehrveranstaltungen. „LVID“ mit dem Datentyp int(11), „LVNAME“ mit dem Datentyp varchar(255), „LVDAUER“ mit dem Datentyp enum(‘45‘, ‘90‘, ‘180‘) und „LVKURZ“ mit dem Datentyp varchar(5) sind die Spalten der Tabelle. Diese hat eine Beziehung zu der „stundenplaneintrag“-Tabelle, das bedeutet, dass ein Eintrag im Stundenplan eine Lehrveranstaltungsart zugewiesen werden kann.

Für die Einträge der Stundenpläne wurde die „studenplaneintrag“ Tabelle erstellt. Diese Tabelle enthält die ganzen Einträge in den jeweiligen Stundenplänen und werden in der Webapplikation in dem jeweiligen Wochenplan aufgelistet. Ein Eintrag besteht aus den Fremdschlüsseln „Sgmodul“, “lehrveranstaltungsart“, „raum“ und „stundenplansemester“. Zusätzlich hat ein Eintrag eine Start- und Endzeit mit einem Wochentag, damit der Eintrag eine exakte Zeit haben kann. Zudem noch einen Termin, da es vorkommen kann, dass es mehrere Gruppen gibt. Anschließend eine Studierendenanzahl, damit es eine Übersicht gibt, wie viele Studenten zu dem Termin kommen. Zum Schluss hat ein Eintrag noch einen Zeitstempel, sodass erkannt wird, wann der Termin bearbeitet worden ist. Der Umfang der „stundenplaneintrag“-Tabelle umfasst 11 Spalten, diese entsprechen „SPID“ mit dem Datentyp int(11), „SPEStartZeit“ mit dem Datentyp datetime, „SPEEndZeit“ mit dem Datentyp datetime, „Wochentag“ mit dem Datentyp enum(‘Montag‘, ‘Dienstag‘, ‘Mittwoch‘, ‘Donnerstag‘, ‘Freitag‘, ‘Samstag‘), „SPTermin“ mit dem Datentyp int(1), „FK\_SGMID“ mit dem Datentyp int(11), „FK\_LVID“ mit dem Datentyp int(11), „FK\_RID“ mit dem Datentyp int(5), „FK\_SPSID“ mit dem Datentyp int(4), „Studierendenanzahl“ mit dem Datentyp int(5) und „Zeitstempel“ mit dem Datentyp timestamp.

Abbildung 3.1: Datenbank Übersichtstabellen mit Beziehungen

## 4 Vorbereitung zum Projekt

## 4.1 Verbindung zur Datenbank

Um eine Verbindung zur Datenbank herzustellen wird ein MySQL Connector benötigt.  
Dieser wird über den Data Source Explorer eingestellt, falls dieser nicht in der Oberfläche zu finden ist, befindet sich diese unter dem Menüpunkt 🡪 Window 🡪 Show View 🡪Other… unter Data Management. Danach erscheint der Data Source Explorer in dem unteren Feld und es werden die Punkte Database Connections und ODA Data Source angezeigt.

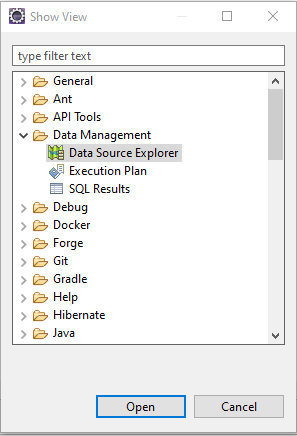


Abbildung 4.1: Einstellung der Oberfläche

Durch Rechtsklick auf den Punkt Database Connections erscheint ein kleines Fenster mit den Eigenschaften New und Refresh. Mit New öffnet sich das Fenster New Connection Profile.

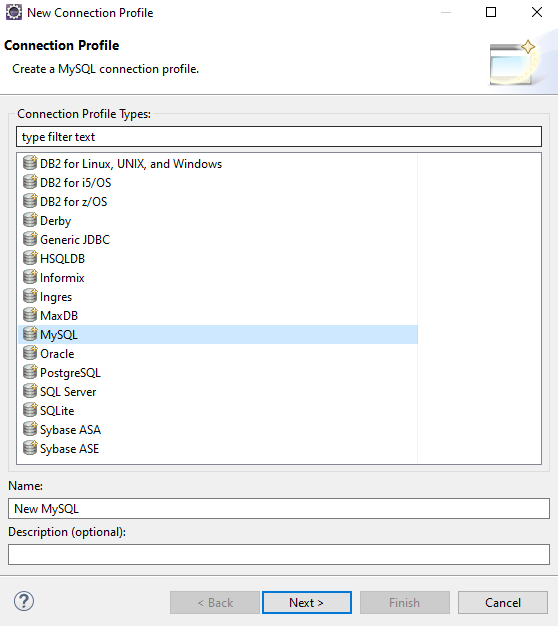


Abbildung 4.2: Verbindungsprofil

Da die Maria Datenbank auf MySQL basiert und Eclipse nicht mit dem MariaDB Connector arbeiten kann, wird der dazu kompatible MySQL Connector mit der Version 5.1.48 verwendet. Deswegen wird in der Liste MySQL ausgewählt, dazu einen passenden Namen für das Connection Profil eingetragen und anschließend auf Next geklickt.

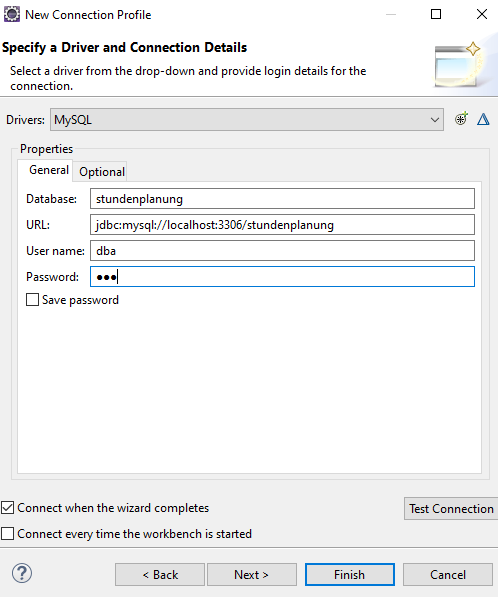


Abbildung 4.3: Verbindungsprofil für Treiber

Dann erscheint das obrige Fenster, wo die Daten zur Datenbank gepflegt werden soll.  
In dem Feld neben Database sollte die erstellte Datenbank, bzw. die Datenbank eingetragen werden zu der die Verbindung hergestellt werden soll. Die URL ist standartgemäß als „jdbc:mysql://localhost:3306/database“ einzutragen, in dem fall wird anstatt „database“ die Datenbank „stundenplanung“ ersetzt. Die Zugangsdaten werden aus der Applikation phpMyAdmin verwendet. Nachdem alle einträge Korrekt eingetragen wurden, wird nun die Verbindung getestet mit Test Connection. Dort sollte dann ein Fenster mit „Ping succeded“ erscheinen und zuletzt werden die Eingaben mit Finish gespeichert und beendet.

Falls der MySQL Connector nicht automatisch erscheint, muss dieser selbst hinzugefügt werden. Dazu ist das eingekreiste plus symbol neben den Drivers anzuklicken.

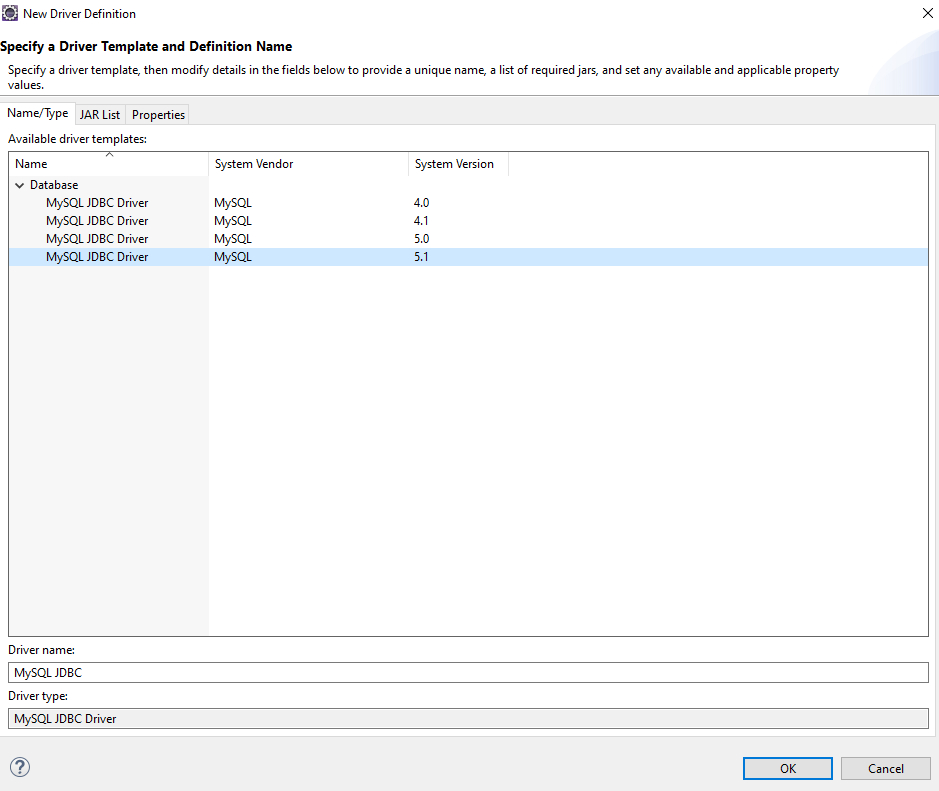


Abbildung 4.4: Einstellung des Treibertemplates

Nun wird eine Liste mit allen MySQL JDBC Driver angezeigt. Dort wird aus der Liste der kompitable Driver ausgewählt, welcher meistens mit der höchsten System version gekennzeichnet ist. Unten im Driver name muss ein eindeutiger Name vergeben werden, sonst entstehen Namenskonflikte. In dem Reiter JAR List wird die jar-Datei des Connectors eingebunden.

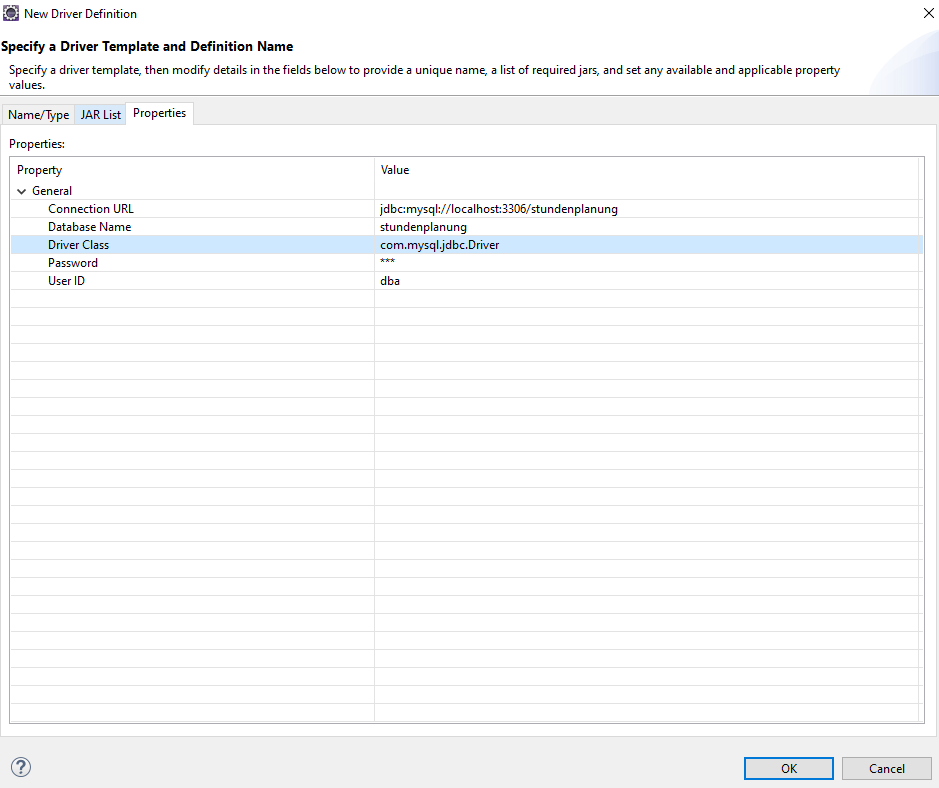


Abbildung 4.5: Einstellung der Treiberdetails

In dem letzten Reiter Properties werden die Daten für den MySQL Driver eingetragen und anschließend nochmal getestet, ob die Verbindung hergestellt werden kann.

## 4.2 Erstellung eines Maven-Projekts

Für die Erstellung eines Maven-Projekts wählt man im Menüpunkt unter File🡪New🡪Other…  
und sucht nach dem Maven-Project.

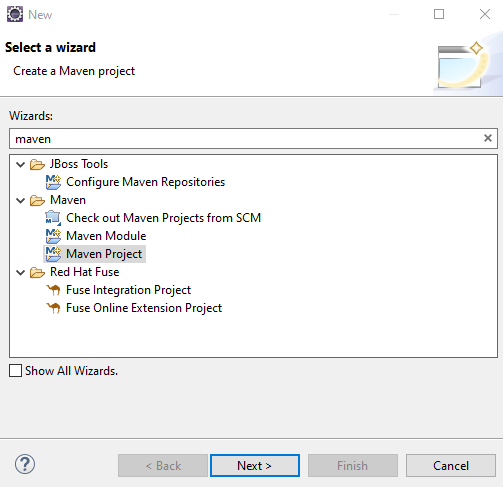


Abbildung 4.6: Auswählen des Projekts

Zu aller erst wird der Ort des Projekts festgelegt und danach wird der Archetype maven-archetype-webapp mit der Group ID org.apache.maven.archetypes ausgewählt.

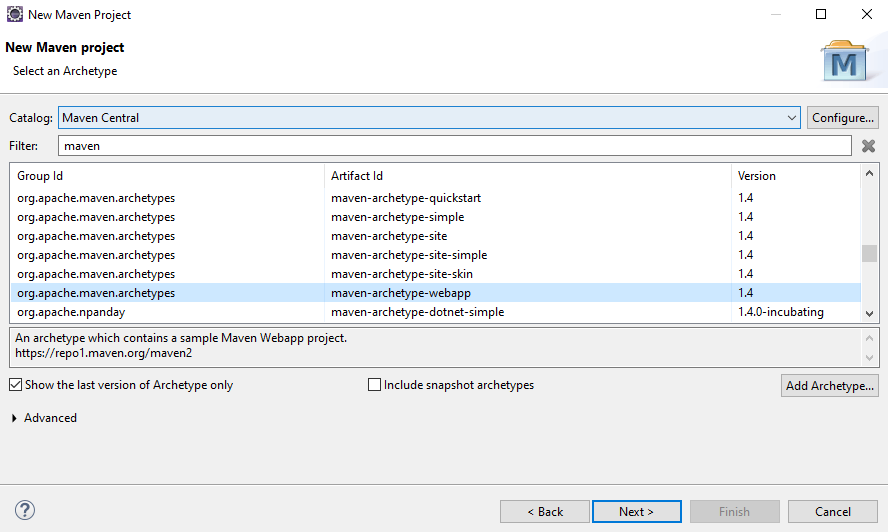


Abbildung 4.7: Auswahl des Mavenprojekts

Wenn der Archetype in der Liste gefunden wurde, wird dieser selektiert und anschließend auf Next geklickt.

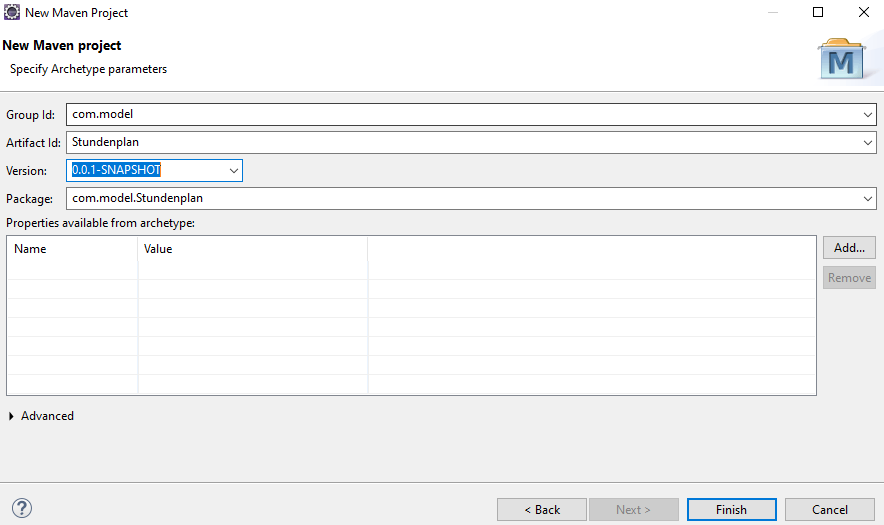


Abbildung 4.8: Name des Projekts

Für den Namen des Projektes wird die Artifact ID mit einem passenden Namen versehen und zuletzt mit Finish beendet.   
Nach der Erstellung des Projekts erscheint das Projekt im Project Explorer. Um ein lauffähiges Projekt zu bekommen, sind noch andere Schritte notwendig.

## 4.3 Einstellungen des Projekts

Als aller erstes werden die Eigenschaften des Projekts aufgerufen. Dies geschieht durch rechtsklick auf das Projekt🡪Properties.   
Danach wird der Menüpunkt Java Build Path aufgerufen und unter dem Reiter Libraries die JRE System Library editiert, sodass man die Java Version JDK 1.8 auswählen kann.

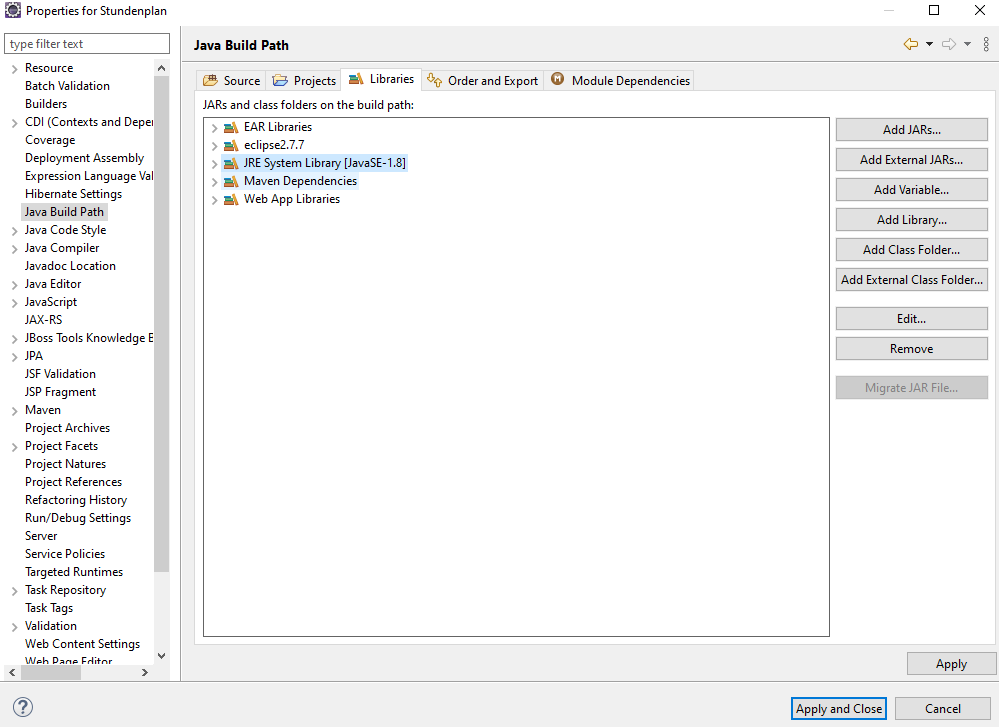


Abbildung 4.9: Java Build Path

Damit das Projekt neu erstellt wird, wird das Fenster mit Apply gespeichert. Des weiteren wird der Menüpunkt Project Facets ausgewählt und die JPA Version auf 2.0 gesetzt. Dort erscheint darunter eine Fehlermeldung mit „Further configuration required…“, diese wird angeklickt und es erscheint folgendes Fenster.

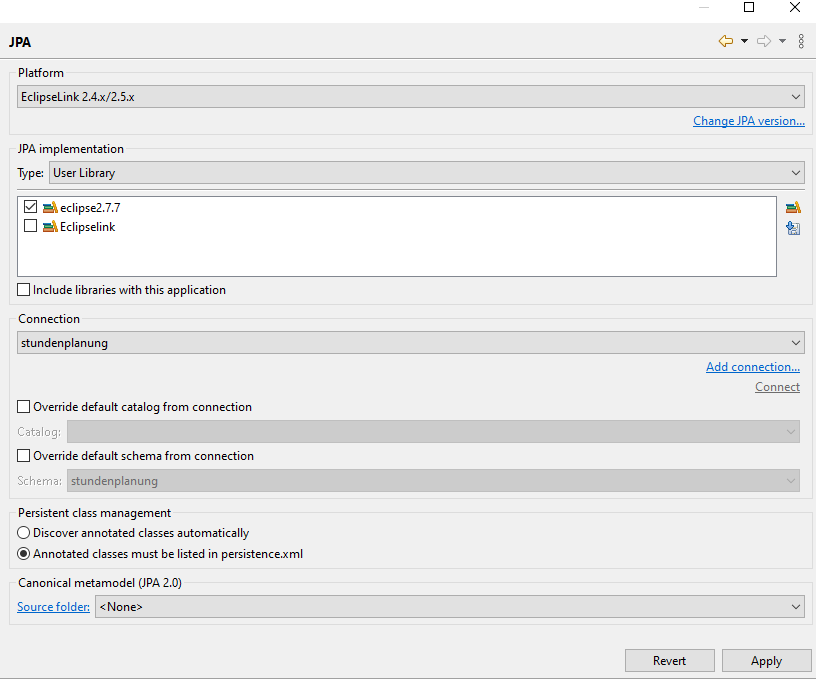


Abbildung 4.10: JPA Einstellung

Für die Platform wird EclipseLink 2.4.x/2.5.x ausgewählt und eine User Library mit EclipseLink eingebunden. Unter Connection wird die Datenbank selektiert, zu der man die Verbindung hergestellt hat. Anschließend wird die Eingabe mit Apply bestätigt. Danach sollte das Fenster wieder die Project Facets anzeigen.

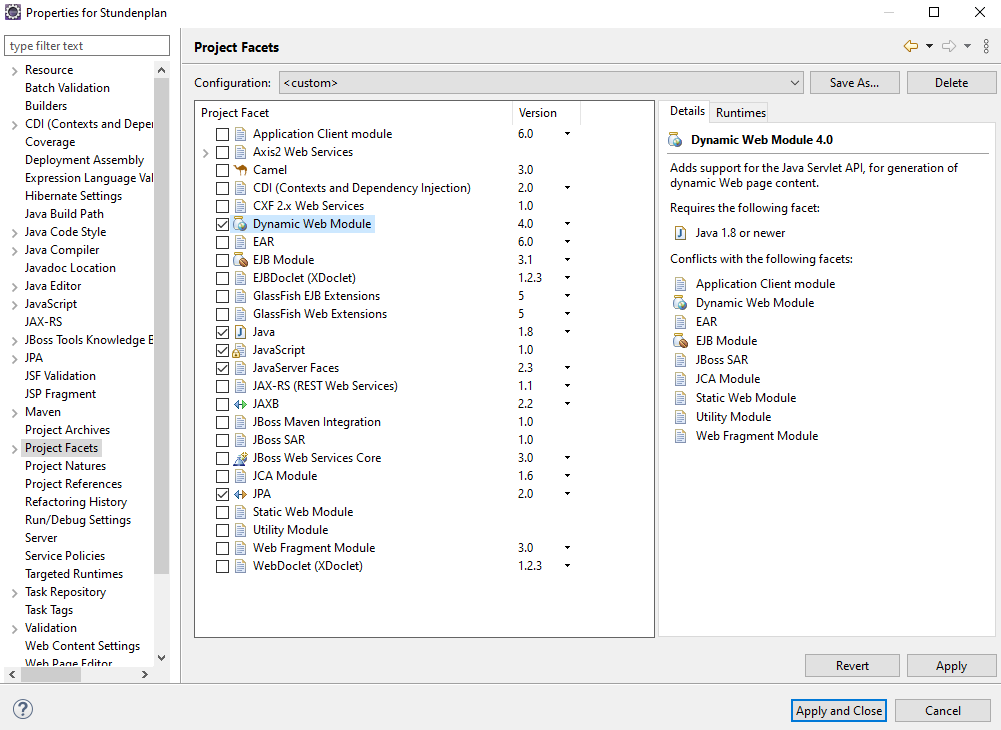


Abbildung 4.11: Projekt Einstellungen

In dem obigen Bild werden die Versionen der einzelnen Module mit folgender Reihenfolge eingestellt:

- Java 🡪 1.8  
- JavaScript 🡪 1.0  
- JavaServer Faces 🡪 2.3  
- Dynamic Web Module 🡪 4.0

Zum Schluss werden die Eingaben mit „Apply and Close“ bestätigt und gespeichert.

## 4.4 Erstellen der beans.xml

Nachdem erfolgreichen Abschluss der Einstellungen des Projekts, wird die beans.xml erstellt. Diese wird mit einem rechtsklick im Projektbaum unter WebContent🡪WEB-INF🡪New🡪beans.xml erstellt.

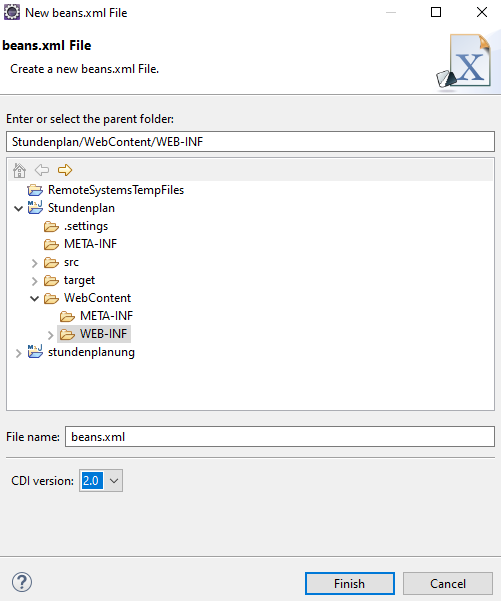


Abbildung 4.12: Beans.xml

Die CDI-Version wird auf 2.0 gesetzt und zuletzt mit Finish wird die beans.xml in dem Projekt-Pfad erstellt. Nachdem die beans.xml erstellt wurde, ist in dem Code der Bean-Discovery-Mode standartmäßig auf „annotated“ eingestellt. Diese muss auf „all“ gesetzt und anschließend gespeichert werden.

## 4.5 Erforderliche Libraries

Es gibt zwei verschiedene Wege um Libraries in das Projekt mit einzubinden.  
Der einfache Weg ist in der pom.xml die Dependencies hinzuzufügen und danach das Projekt über Maven zu aktualisieren.   
Der andere Weg ist die gewünschten JAR-Dateien extern herunterzuladen und über dem Java Build Path unter dem Reiter Libraries mit „Add External JARs“ hinzuzufügen.   
Folgende Libraries werden in dem Projekt verwendet:

Maven Dependencies

* Commons-math3-3.6.1.jar  
  Wird benötigt für Mathematik- und Statistikkomponenten.
* Primefaces-8.0.jar  
  Bibliothek für das User-Interface
* Javaee-api-8.0.jar  
  Bibliothek für die Java Enterprise Edition Applikation
* Javax.faces-2.4.0.jar  
  Wird benötigt für die Benutzung von Webanwendungen
* Javax.el-api-3.0.1-b04.jar  
  Zuständig für die Kommunikation zwischen den Webkomponenten und den Klassen
* javax.el-3.0.1-b04.jar  
  Bietet eine Implementierung zum Erstellen und Auswerten von EL-Ausdrücken.
* javax.mail-1.6.0.jar  
  Bietet Klassen für die Modellierung eines Mailsystems
* activation-1.1.jar  
  Aktivierungsframework zum Verwalten von Java-Beans
* itextpdf-5.5.10.jar  
  Freie PDF-Bibliothek wird verwendet um PDF-Dateien zu generieren

Web App Libraries

* all-themes-1.0.10.jar  
  Mehrere Weboberflächendesigns von Primefaces
* cdi-api-1.2.jar  
  Dienst, die zur Verbesserung der Struktur des Anwendungscodes beiträgt
* commons-math3-3.6.1.jar  
  Wird benötigt für Mathematik- und Statistikkomponenten.
* itext-2.1.7.jar  
  Freie PDF-Bibliothek wird verwendet um PDF-Dateien zu generieren
* javax.inject.jar  
  Wird benötigt zum Injizieren von anderen Klassen
* javax.transaction-api-1.3.jar  
  Zuständig für die Transaktion
* jsf-api-2.2.20.jar  
  Wird benötigt für die Java Server Faces Applikation
* mysql-connector-java-5.1.48.jar  
  Connector für die Verbindung zur Datenbank
* poi-4.1.1.jar  
  Für den Zugriff auf Microsoft-Formatdateien
* primefaces-8.0.jar  
  Bibliothek für das User-Interface

Referenced Libraries

* jsf-api.2.2.20.jar  
  Wird benötigt für die Java Server Faces Applikation
* mysql-connector-java-5.1.48.jar  
  Connector für die Verbindung zur Datenbank
* javax.transaction-api-1.3.jar  
  Zuständig für die Transaktion
* javax.inject.jar  
  Wird benötigt zum Injizieren von anderen Klassen
* cdi-api-1.2.jar  
  Dienst, die zur Verbesserung der Struktur des Anwendungscodes beiträgt
* primefaces-8.0.jar  
  Bibliothek für das User-Interface
* all-themes-1.0.10.jar  
  Mehrere Weboberflächendesigns von Primefaces

Neben den anderen Paketen, benötigt man noch die Eclipse Web Developer Tools 3.19 und Eclipse Enterprise Java and Web Developer Tools 3.19 für die Webentwicklung. Diese befinden sich im Eclipse Marketplace zur freien Installation.

## 4.6 Erstellung der Entities

Um die Entities aus der Datenbank zu generieren, wird eine aktive Verbindung zur Datenbank benötigt. Wenn dies der Fall ist und der JDBC Connection Pool in dem Applikationsserver existiert, wird die persistence.xml geöffnet, welches unter dem Projektbaum unter JPA Content zu finden ist.   
Bevor die Entity Tables aus der Datenbank generiert werden sollte in der persistence.xml die JTA Data Source erstellt werden. In dem Glassfish Server wäre das in dem Fall „ <jta-data-source>jdbc/stundenplanung</jta-data-source>“ und in dem WildFly Server „<jta-data-source>java:/jdbc/stundenplanung</jta-data-source>“.  
Danach wird das Projekt mit einem Rechtsklick ausgewählt und auf den Reiter JPA Tools🡪Generate Entities from Tables… geklickt.

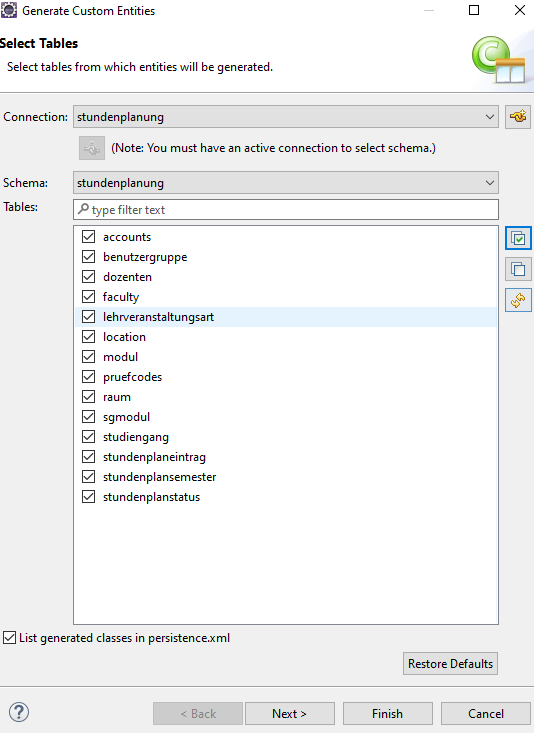


Abbildung 4.13: Auswählen der Tabellen

In diesem Schritt werden die Connection, Schema und die dazugehörigen Tabellen ausgewählt. Der Hacken bei „List generated classes in persistence.xml“ sollte ebenfalls gesetzt sein. Mit Next kann man in weiteren Schritten andere Einstellungen für die Generierung vornehmen, aber diese werden in diesem Fall nicht benötigt und deswegen mit Finish generiert. Die Entity Java Klassen finden sich im Projektbaum unter Java Resources in dem Package „model“ wieder.   
Nach der erfolgreichen Generierung der Entities müssten die Entity Java Klassen in der persistence.xml eingetragen sein. In der persistence.xml gibt es keinen Unterschied zu den NetBeans Projekten.

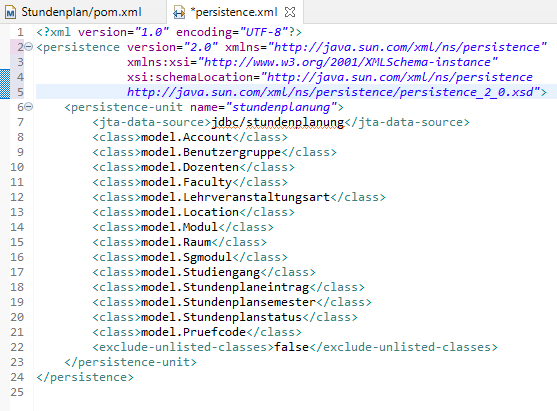


Abbildung 4.14: Persistence.xml Datei

## 5 Javadoc und Internationalisierung

## 5.1 Javadoc

Ein Javadoc ist ein Dokument, welches alle Methoden in einer Übersicht auflistet und schnell die gewünschte Methode zu lokalisieren, wo diese verwendet wird. In dieser Methode kann nachgeschaut werden, zu welchem Zweck und wofür diese gebraucht wird.   
Für ein Javadoc ist es ratsam für jede Methode ein Javadoc Header zu erstellen. Dies geschieht durch einen Rechtsklick auf die Methode 🡪 Source 🡪 Generate Element Comment oder durch die Tastenkombination Alt+Shift+J, wodurch der Javadoc Header generiert wird. Aus der Methode werden die Parameter und die Exeptions automatisch in den Javadoc Header geschrieben. Zusätzlich könnte ein Kommentar zu der Methode geschrieben werden, um diese zu beschreiben. Um ein Javadoc generieren zu können muss auf das Exportverzeichnis zugegriffen werden. Das läuft folgendermaßen ab, es wird ein Rechtsklick auf das Projekt durchgeführt und dann den Pfad Export 🡪 Export… 🡪 Java 🡪 Javadoc gefolgt.

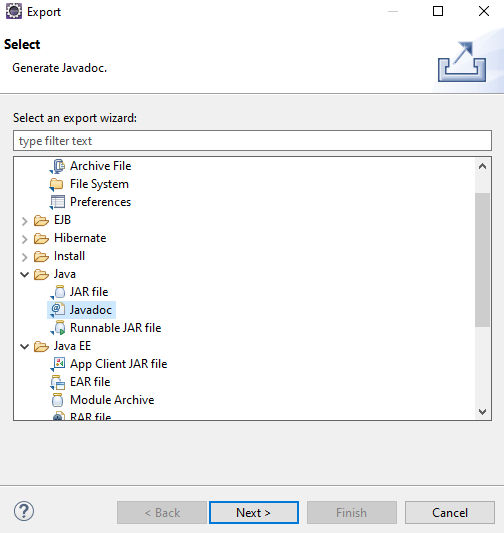


Abbildung 5.1: Auswahl des Javadocs

Wenn der Punkt Javadoc ausgewählt wurde, wird in dem Fenster der Punkt Next gewählt, um weitere Einstellungen zu tätigen.

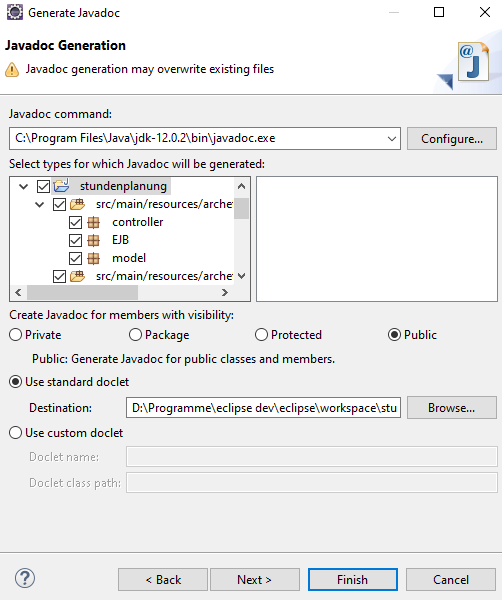


Abbildung 5.2: Javadoc Generierung

In diesem Fenster wird das jeweilige Projekt ausgewählt, dann die Sichtbarkeit des Javadocs und den Pfad, wo die Java Dokumentation gespeichert werden soll. Auf der nächsten Seite wird der Titel von dem Javadoc angegeben und zusätzlich die Basis Optionen und Tags ausgewählt.

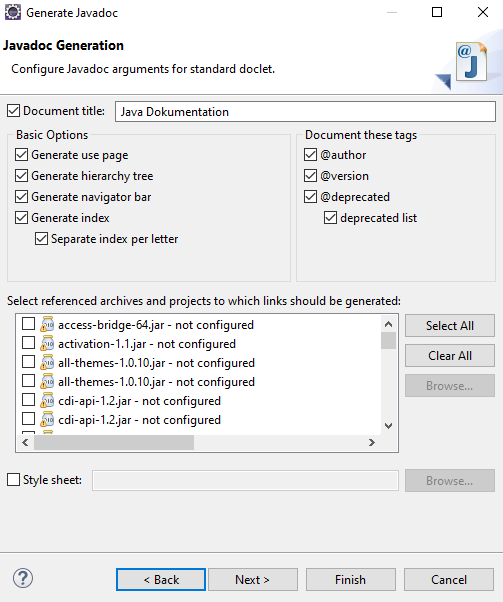


Abbildung 5.3: Javadoc Genereirung für die zweite Seite

Zum Schluss wird die Eingabe mit Finish beendet, das Javadoc wird erstellt und dann in einem Browser geöffnet.

## 5.2 Internationalisierung

Bei der Internationalisierung werden Status Meldungen, Labels und Textfelder als Variablen gespeichert. So können diese Variablen einfach in die jeweilige Sprache umgewandelt werden. In diesem Beispiel ist die Sprache Deutsch als Ausgangssprache und Englisch als Zweitsprache deklariert. Die Variablen sind extern unter src 🡪 main 🡪 resources 🡪 messages gespeichert. In diesem Ordner werden die Files messages.properties und messages\_en.properties erstellt, welche diese mit einem Rechtsklick auf den Ordner messages 🡪 new 🡪 File hinzugefügt werden.

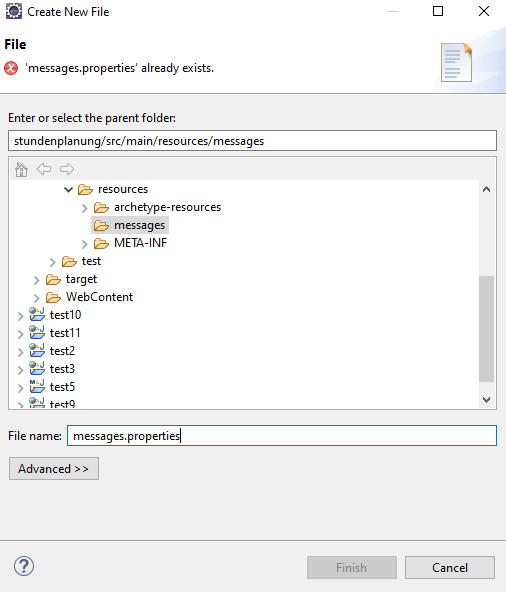


Abbildung 5.4: Erstellung der message.properties Datei

In der Datei können alle Variablen, die zur Internationalisierung verwendet werden hinein geschrieben. Die Variablennamen müssen einmal in dem File messages.properties und messages\_en.properties vorhanden sein. In dem folgenden Beispiel wird dies deutlich.

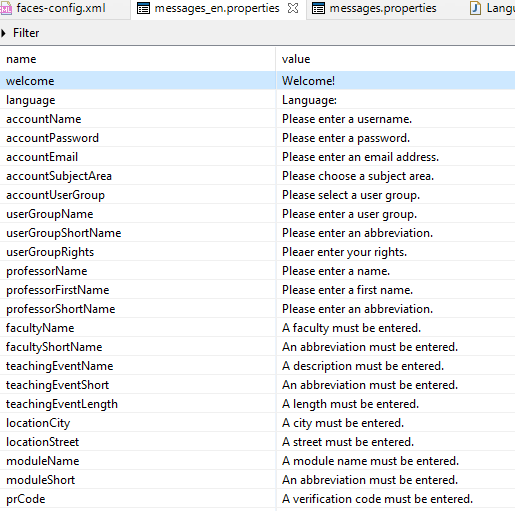
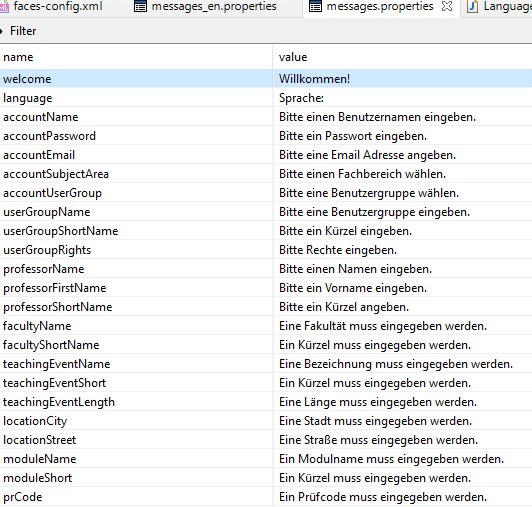


Abbildung 5.5: Message.propierties und Message\_en.propierties Datei

In dem linken und rechten Bild sind Namen der Variablen gleich, somit können diese hinterher angesteuert werden. Der Wert (value) dieser Variablen sind unterschiedlich, da das linke Bild für die deutsche Sprache ist und das rechte Bild für die englische Sprache.

Die faces-config.xml Datei wird um folgende Zeilen erweitert. Diese sind:

<application>

<locale-config>

<default-locale>de</default-locale>

<supported-locale>en</supported-locale>

</locale-config>

<resource-bundle>

<base-name>messages.messages</base-name>

<var>msg</var>

</resource-bundle>

</application>

Die Ausgangssprach wird bei „default-locale“ auf de für Deutsch und die Zweitsprache bei „supported-locale“ auf en für Englisch gesetzt. Das Resource-Bundle steht unter dem Pfad src/main/resources/messages dadurch weiß das Programm, wo es zu finden ist. Mit der Variable msg werden alle Variablen innerhalb der messages.properties und messages\_en.properties angesprochen.

Der LanguageController besteht aus einer privaten String Variable mit dem Namen locale mit setter und getter Funktionen. Außerdem eine private statische HashMap mit den Attributen String und Objekt und den Namen countries. Diese Variable wird neu erzeugt und die Sprachen Deutsch und Englisch hinzugefügt, wie in dem unteren Beispiel gezeigt. Diese werden als eine Liste in der master.xhtml verwendet.

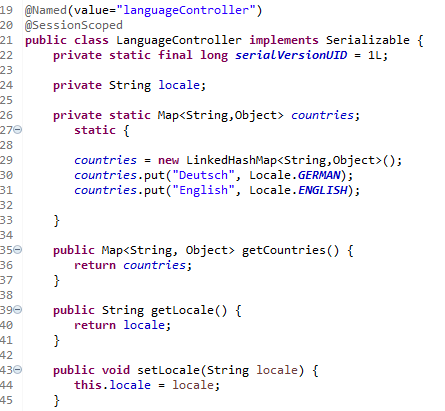


Abbildung 5.6: LanguageController

Zusätzlich hat die Klasse LanguangeController eine Methode namens localeChanged, welche ein ValueChangeEvent Parameter besitzt. Diese Methode schaut, welche Sprache in der master.xhtml ausgewählt worden ist und ändert alle Variablen in der messages.properties bzw. messages\_en.properties auf ihren Wert.

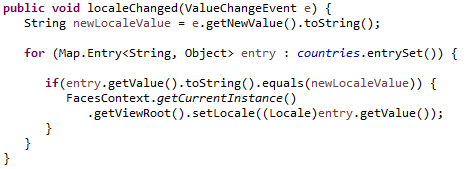


Abbildung 5.7: Methode localeChange

In der master.xhtml steht auf der Ost-Seite des Formulars die Internationalisierung.

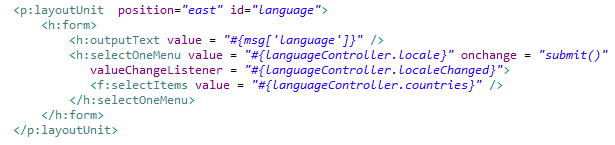


Abbildung 5.8: Ausschnitt aus der master.xhtml

Die Variablen werden zum Beispiel mit <h:outputText value = “#{msg[‘language‘]}“/> aufgerufen. „msg“ steht für die Bundle-Variable und „language“ für Variable in der messages.properties bzw. messages\_en.properties. Es befindet sich zusätzlich noch ein Dropdownmenü mit dabei, um zwischen den Sprachen hin und her zu wechseln. Wenn die jeweilige Sprach ausgewählt worden ist, werden alle Variablen, die eingetragen wurden geändert.

In den Controllern gibt es jeweils eine addInfoMessage Methode, um die Status Meldungen auszugeben. Diese hat einem String Parameter, um die Variablen in der messages.properties anzusprechen. Danach wird die Meldung mit einer FacesMessage ausgegeben.

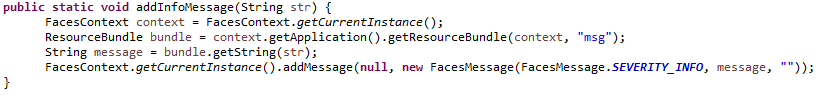


Abbildung 5.9: Methode addInfoMessage

## 6 Master Template und Use Cases

Das Mastertemplate ist eine Weboberflächenvorlage, welche für jede xhtml-Seite benutzt wird. Dieses besteht aus verschiedenen LayoutUnits, die sind in alle Himmelsrichtungen ausgelegt. In dem Nord-Bereich werden der Titel „Stundenplan“, der Name des Benutzers und die jeweilige Benutzergruppe, falls ein Benutzer angemeldet ist, ausgegeben. Die West-Seite besteht aus dem Menü, diese wird für die Navigation verwendet. Die Ost-Seite beinhaltet die Internationalisierung, um zwischen Deutsch und Englisch zu wechseln. Der Hauptteil befindet sich im zentralen Teil, welches „content“ genannt wird, indem sich die Interaktionen der gesamten Applikation wiederspiegeln.   
Je nach Benutzergruppe, ergibt sich ein anderes Layout im Menü. Die vier verschiedenen Benutzergruppen sind Administrator, Raumzeitplaner, Studiengangsleiter und Nobody. Diese haben jeweils unterschiedliche Rechte.

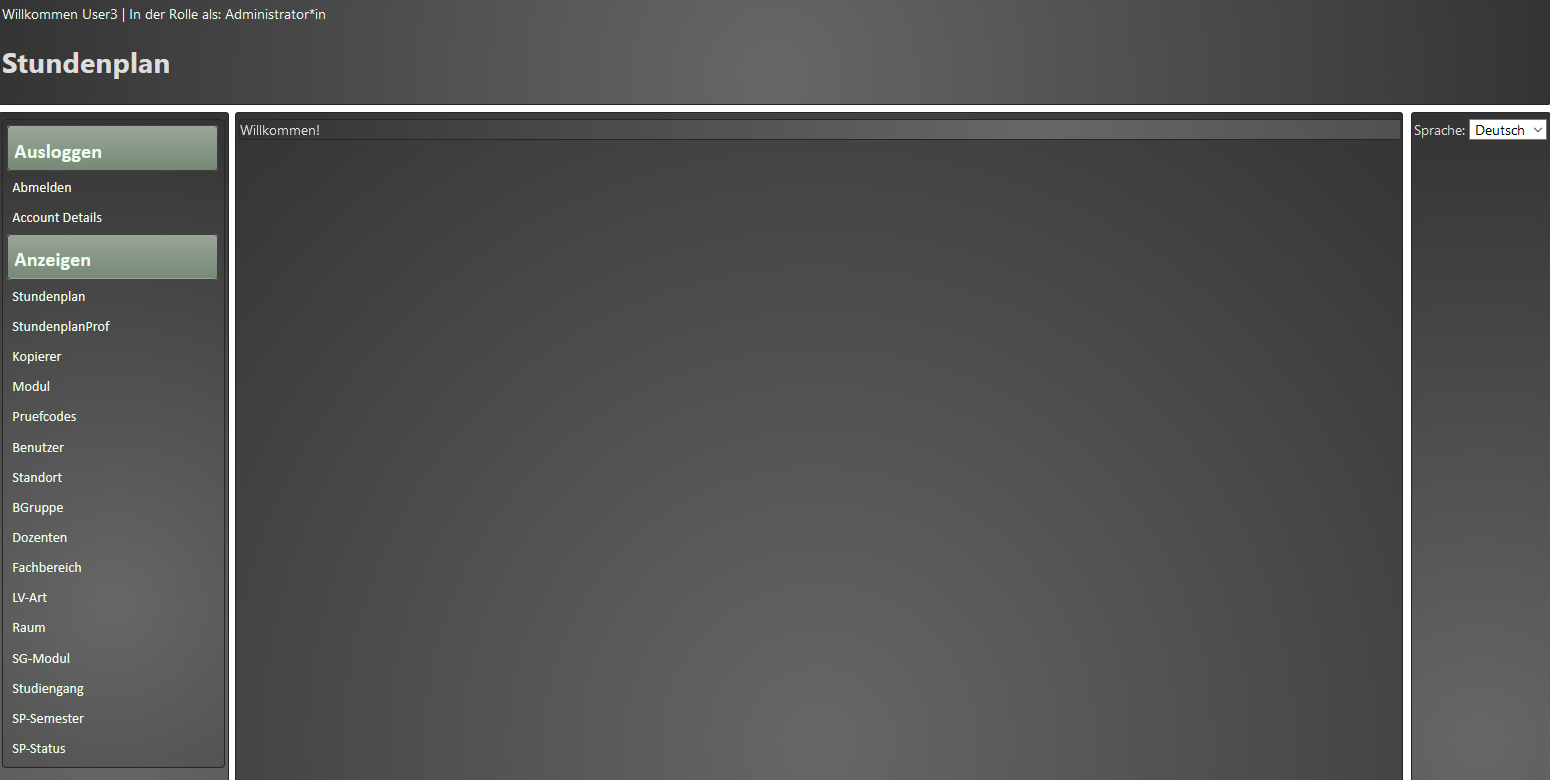
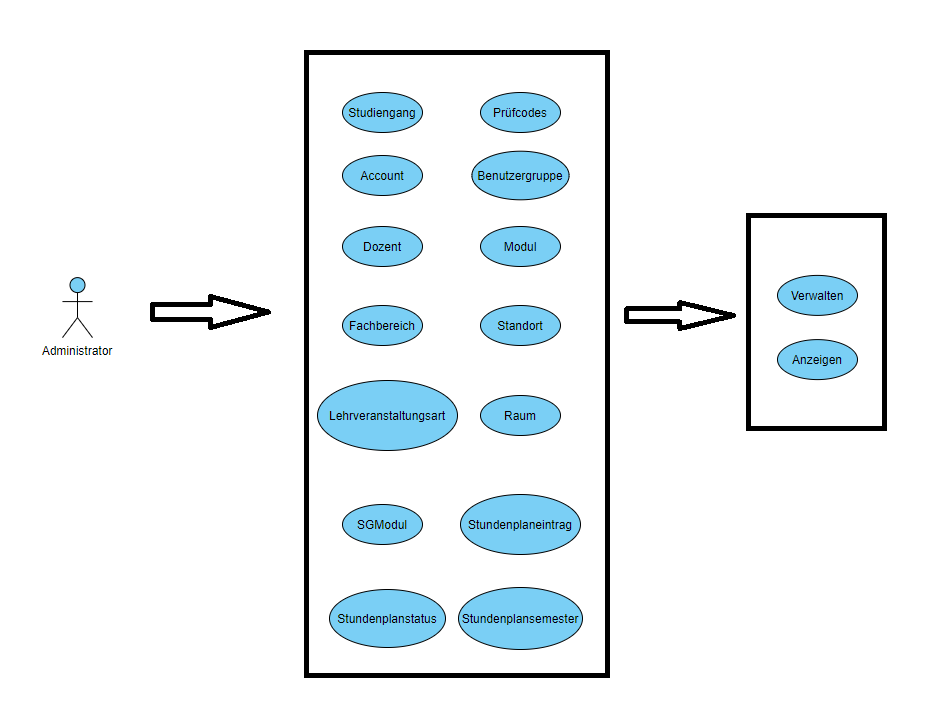


Abbildung 6.1: UseCase Diagramm für den Administrator mit dem entsprechenden Menü

Der Administrator ist derjenige mit den meisten Rechten, er kann im Grunde alle Daten bearbeiten und hat im Menü alle xhtml-Seiten aufgeführt.

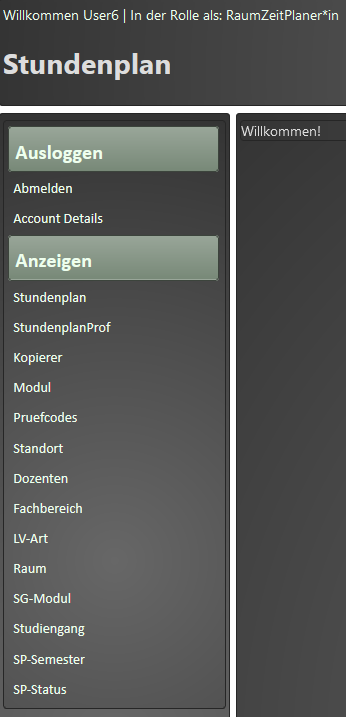
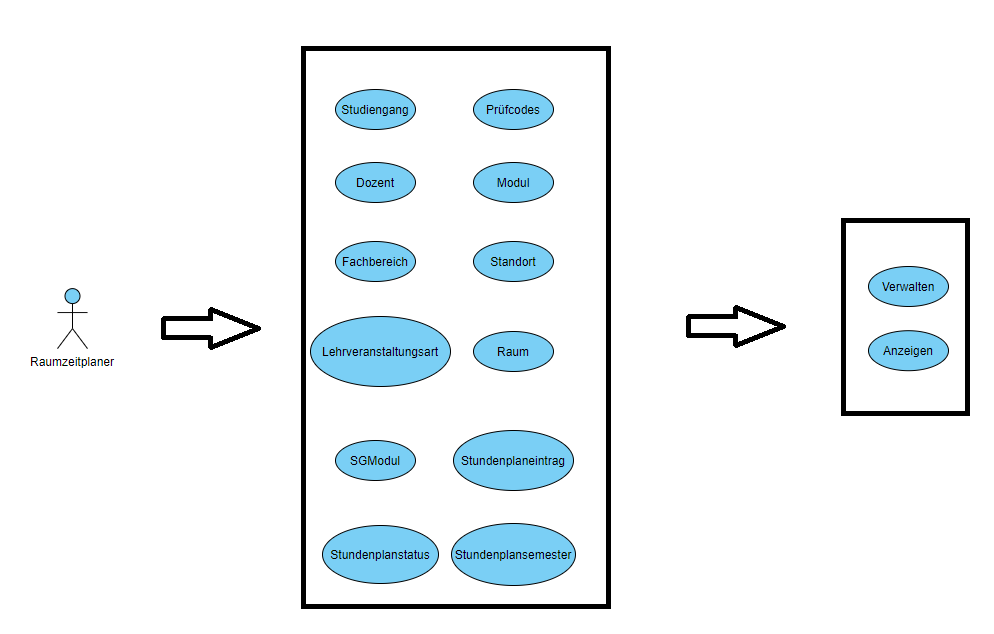


Abbildung 6.2: UseCase Diagramm für den Raumzeitplaner mit dem entsprechenden Menü

Der Raumzeitplaner darf wie der Administrator alles außer Benutzer hinzufügen/bearbeiten und Benutzergruppen erstellen.

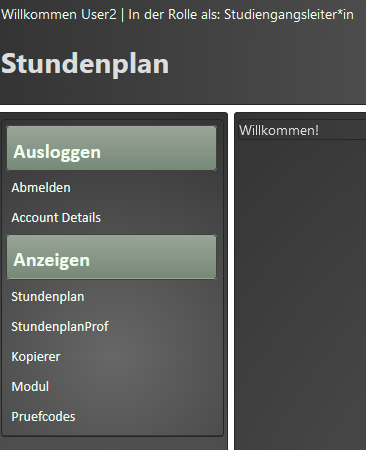
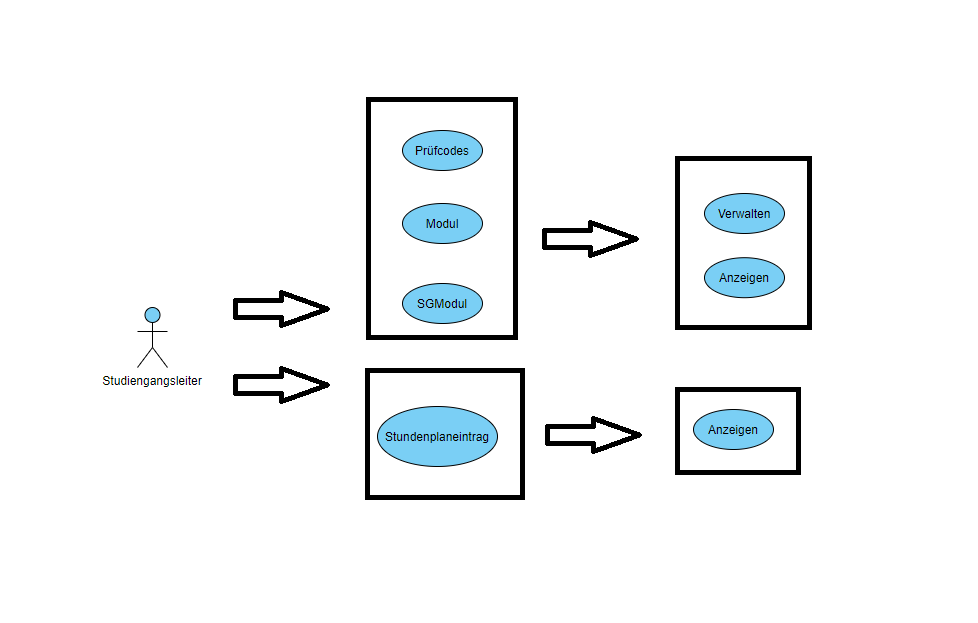


Abbildung 6.3: UseCase Diagramm für den Studiengangsleiter mit dem entsprechenden Menü

Der Studiengangsleiter darf Module und Prüfcodes hinzufügen und bearbeiten. Außerdem Stundenpläne erstellen und diese von vorherigen Jahren auf die aktuellen Semestern kopieren. Auch kann er die Stundenpläne von anderen Dozenten einsehen.

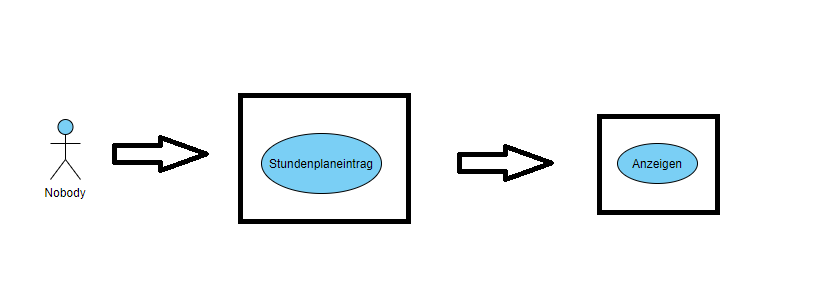


Abbildung 6.4: UseCase Diagramm für den Nobody mit dem entsprechenden Menü

Der Nobody kann nur die Stundenpläne einsehen und seine Account Daten ändern.

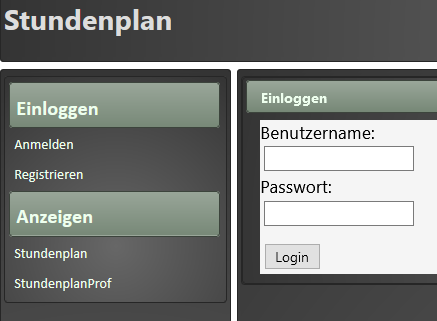
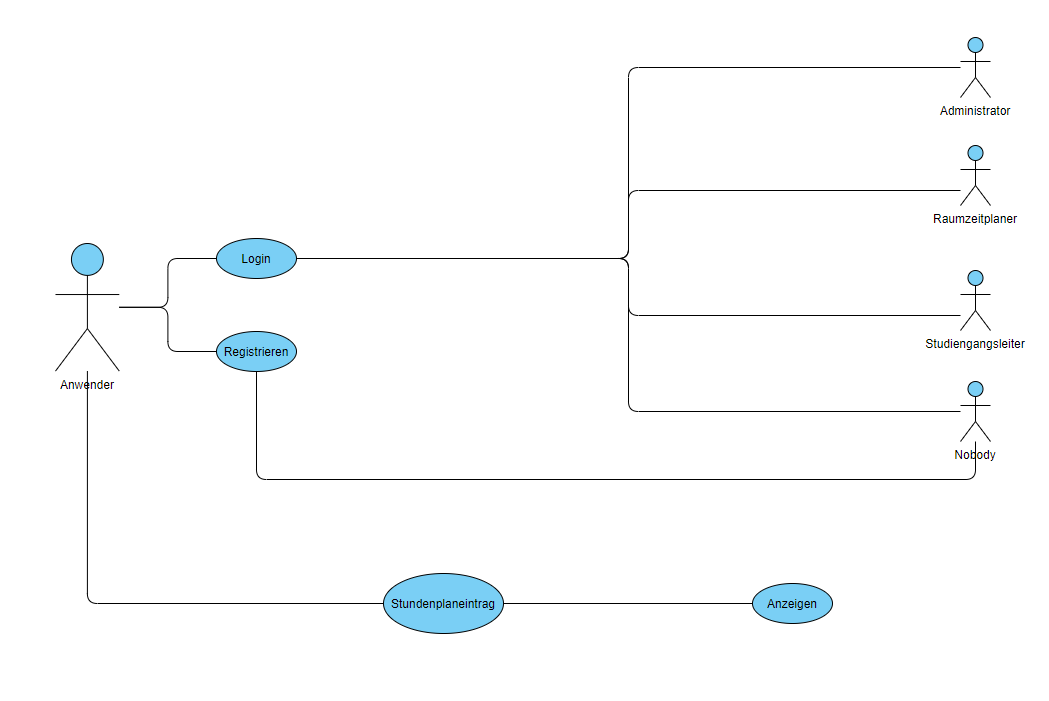


Abbildung 6.5: UseCase Diagramm für den Anwender

Wenn man nicht angemeldet ist, hat der Benutzer die gleichen Rechte wie der Nobody außer das er Account hat und seine Daten ändern kann. Unter dem Menüpunkt Registrieren kann er sich einen Account erstellen. Dieser wird direkt die Benutzergruppe Nobody zugewiesen.

## 7 Verwendung von EJBs

EJBs, auch Enterprise Java Beans genannt, dienen zur Kommunikation zwischen der Datenbank und dem Applikationsserver. Die Transaktionen werden automatisch durch den EJB-Container beim Persistieren gesteuert. Der Container benutzt schreib- und lesesperren, damit es nicht zu Konflikten führt, wenn mehrere Benutzer einen Datensatz bearbeiten.   
Die EJB Klassen wurden alle in Netbeans generiert und in Eclipse implementiert, da in Eclipse die Funktion „SessionBean for Entity Classes“ für die automatische Generierung der Facades nicht vorhanden ist. In Eclipse gibt es zwar die Funktion „SessionBean (EJB 3.x)“, aber diese generiert nur die Klassen ohne Inhalt und die Abstract-Klasse fehlt komplett.  
Es gibt drei Klassenbestandteile für die Funktionalität der EJB-Container. Diese sind folgende:

* AbstactFacade
* EntityFacade
* EntityFacadeLocal

Die AbstractFacade Klasse ist die zentrale EJB Klasse und beinhaltet die Queries für die Kommunikation und Transaktion mit der Datenbank. Die Queries bestehen aus den Methoden create, edit, remove, find Object, findAll, findRange und count.   
Für die Erstellung eines neuen Datensatz wird die create-Methode verwendet. Beim Bearbeiten eines Datensatzes ist die edit-Methode dafür zuständig. Das Löschen eines Datensatzes erfolgt durch die remove-Methode. Wenn ein Objekt durch eine ID gefunden werden soll, findet die Methode das Objekt anhand der ID. Um alle Einträge in der jeweiligen Tabelle zu finden, wird die finAll-Methode verwendet. Durch die findRange-Methode werden nur bestimmte Datensätze aus der jeweiligen Tabelle aufgerufen. Die count-Methode gibt die Anzahl der vorhandenen Datensätze in einer Tabelle wieder.

In der EntitiyFacade Klasse wird auf das Model der Entity Klasse verwiesen und von der AbstractFacade Klasse erweitert. Hinzu kommt, dass der Entitiy Manager in der Klasse erstellt wird. Dieser wird verwendet für das Transaktionsmanagement.   
Die EntityFacadeLocal Klasse ist eine Interface-Klasse, welche eine Schnittstelle zwischen der AbstractFacade Klasse und die jeweilige Entity Klasse ist. Diese übergibt den Methoden, die sich in der AbstractFacade befinden, die Entity Objekte.

In den jeweiligen Controllern wird mit der Annotation @EJB ein EntityFacadeLocal Objekt erstellt, welche auf die Methoden in der AbstractFacade Klasse zugreifen. Der Zugriff geschieht durch den Befehl EntitiyFacadeLocal.methode(Objekt).

## 8 Controller

## 8.1 Entity Controller

Die Entity Controller sind dafür da, um die Verwaltung der Tabellen in der Datenbank zu gewährleisten. Damit kann der Benutzer die einzelnen Datensätze bearbeiten, löschen und neue hinzufügen. Diese Entity Controller befinden sich in dem Controller Package.  
Da in unserem Programm die Entity Controller alle gleich aufgebaut sind, wird nur auf ein Controller Bezug genommen und die einzelnen Unterschiede kurz eingegangen.

Der SgmodulController wird in den folgenden Absätzen beschrieben. Um auf die dazugehörigen Entities zugreifen zu können, müssen diese mit der Annotation @Inject hinzugefügt werden.   
Beim Starten der Applikation wird die mit @PostConstruct annotierte Methode „init()“ ausgeführt.  
In dieser Methode werden alle Listen von den dazugehörigen Entities aus der Datenbank aufgerufen.

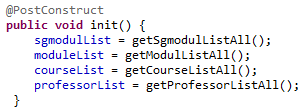


Abbildung 8.1: Methode init()

Die Listen bestehen aus der Elterntabelle „sgmodul“ und den Kindtabellen „modul“, „studiengang“ und „dozenten“.   
Für die Liste der Elterntabelle wird mit der EJB-Methode findAll() alle Datensätze aus der Datenbank hervorgerufen.



Abbildung 8.2: Getter für die Sgmodul Liste

Die Kindtabellen werden auch mit der EJB-Methode findAll() aufgerufen, aber die Liste wird zusätzlich mit einer Sortierfunktion ausgestattet. Die Sortierfunktion ist dafür da, um eine sortierte Liste in der Select-One-Menu Komponente in der jeweiligen xhtml-Seite anzuzeigen.

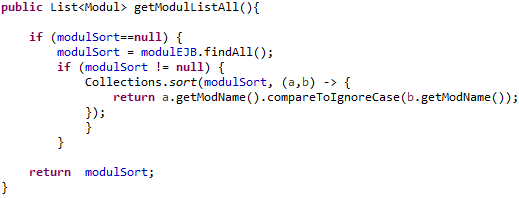


Abbildung 8.3: Getter für die geordnete Modul Liste

Jeder Controller besitzt eine create(), edit() und delete() Methode um die Datensätze in der Datenbank zu verwalten.   
Die create-Methode wird aufgerufen, wenn der Benutzer einen neuen Datensatz hinzufügen möchte.  
Es wird ein neues Entity Objekt erstellt und die eingegebenen Werte von dem Benutzer gespeichert. Dies geschieht mit der EJB-Methode create(). Falls der Benutzer die Eingaben getätigt hat, bekommt dieser eine Informationsmeldung, ob das Objekt in die Datenbank gespeichert worden ist oder nicht.

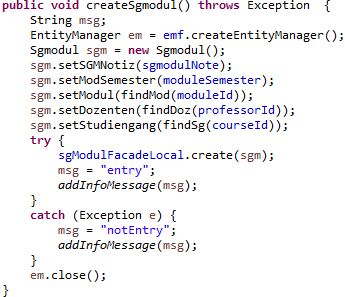


Abbildung 8.3: Methode createSgmodul()

In der createDo()-Methode wird die create()-Methode aufgerufen und die Liste der anzuzeigenden Entity wird aktualisiert.



Abbildung 8.4: Methode createDoSgmodul()

Auf der Weboberfläche sind alle Datensätze in einer Tabelle dargestellt. Durch eine Primefaces-Funktion kann der Benutzer einen Datensatz mit einem entsprechenden Klick auf die jeweilige Zeile bearbeiten oder löschen. Bei der onRowSelect()-Methode wird die entsprechende Zeile aus der Tabelle ausgewählt und das Fenster zum Bearbeiten öffnet sich. Die Fremdschlüssel und die dazugehörigen Datensätze werden durch ein Event mit ihrem Primärschlüssel gefunden. Zudem bekommt der Benutzer eine Informationsmeldung, dass das Objekt ausgewählt wurde.

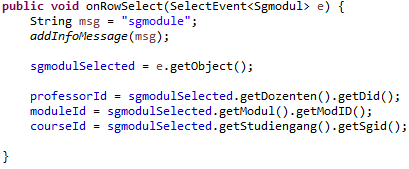


Abbildung 8.5: Methode onRowSelect()

Nachdem öffnen des Fensters zum Bearbeiten eines Datensatzes hat der Benutzer drei Möglichkeiten, um eine Aktion zu tätigen. Dieser kann die alten Attributwerte durch den „Speichern“ Button mit den neuen Werten überschreiben. Mit dem Speichern wird die add()-Methode zum Bearbeiten des jeweiligen Datensatzes aufgerufen und der Benutzer bekommt eine Informationsmeldung, ob das Bearbeiten erfolgreich war oder nicht. Die Methode findet das Objekt durch den EntityManager, in dem die Entity-ID übergeben wird. Das Bearbeiten des Datensatzes in der Datenbank erfolgt durch die EJB-Methode edit(). Anschließend wird die Entity Liste neu geladen und die Seite wird aktualisiert.



Abbildung 8.6: Methode addSgmodul()

Die zweite Möglichkeit ist den Datensatz aus der Liste und dementsprechend aus der Tabelle zu löschen. Falls der Datensatz erfolgreich aus der Datenbank gelöscht worden oder nicht ist, bekommt der Benutzer eine Informationsmeldung. Das Löschen des Datensatzes in der Datenbank erfolgt durch die EJB-Methode remove().

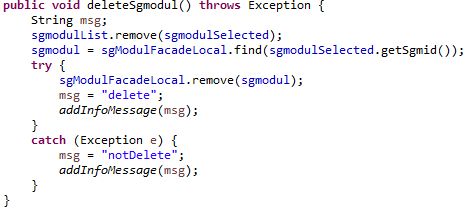


Abbildung 8.7: Methode deleteSgmodul()

Die letzte Möglichkeit ist den Vorgang durch den „Abbrechen“ Button abzubrechen, in dem Fall wird das Fenster geschlossen.

Für die Select-One-Menu Komponenten werden jeweils die find() Methoden verwendet, um das passende Objekt des Fremdschlüssels zu finden und übergeben. Zum Finden wird die EJB-Methode find() verwendet.



Abbildung 8.8: Methode findDoz()

In den vorherigen Versionen des Projekts kam die UserTransaction Methode zum Verwalten und die TypedQueries für die Abfrage der Datensätze aus der Datenbank zum Einsatz.  
Durch den Hinweis des Professors wurde entschieden, dass es zum Verwalten der Datensätze die EJB-Methoden und für die einfachen Abfragen der Datensätze überwiegend die CriteriaQueries benutzt werden sollen. Die EJB-Methoden sind einmal Code sparender und zum anderen sicherer bei der Übertragung, da es automatisch von dem EJB-Container gesteuert wird. Bei komplexeren Abfragen werden weiterhin die TypedQueries genutzt, da es bei diesen Abfragen schwierig ist in die CriteriaQueries umzuwandeln.

Bei den ausgaben der Informationsmeldungen wurden vorher die Meldungen in dem Code integriert und im Nachhinein wurde eine extra Datei für die Internationalisierung erstellt.

## 8.2 ModulController

In dem ModulController wird die create()-Methode ein wenig verändert, da die Anforderung von dem Professor beim Hinzufügen eines Moduls zusätzlich die Auswahl oder Eingabe des Prüfcodes erfordert. Die Auswahl wird mit einem Select-One-Radio Button dargestellt. Die Möglichkeiten sind entweder ein vorhandenen Prüfcode in einer Select-One-Menu Komponente auszuwählen oder durch eine neue Eingabe von einem Prüfcode in einer Input-Text Komponente.   
In der Methode werden mit zwei if-Anweisungen die Auswahl mit einer booelan variable geprüft, falls die boolean Variable false sein sollte, wird der Prüfcode durch die Select-One-Menu Komponente ausgewählt. Andernfalls wird ein neuer Prüfcode mit den eingegebenen Daten in dem PrüfcodeController erstellt und dann das Modul.



Abbildung 8.9: Methode createModul()

Genauso ist es in der add()-Methode des Moduls zum editieren mit aufgebaut, auf die es allerdings nicht mehr weiter eingegangen wird.

## 8.3 RegisterController

Alle Benutzer haben einen einzigartigen Namen, welcher nicht wiederholt in der Datenbank vorkommen darf. Das kommt daher, da der Account durch den Namen unterschieden wird.  
Um bei der Registrierung eines Benutzers diese Regel einzuhalten, wird der Name mit einer TypedQuery nach der Eingabe in der Datenbank überprüft, ob dieser schon vorhanden ist. Dies geschieht mit der checkName()-Methode, welches eine boolean-Variable zurückgibt.

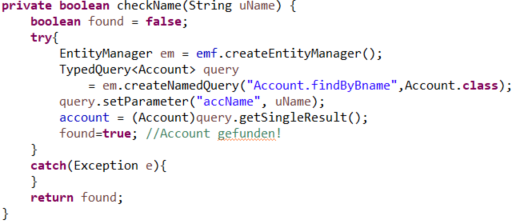


Abbildung 8.10: Methode checkName()

Beim setzten des Benutzernamen wird erst überprüft, ob ein Accountamen eingegeben worden ist. Danach wird der Name an die Methode checkName() übergeben und in dieser wird der Name überprüft. Wenn der Name nicht gefunden wurde, wird der Accountname und eine boolean-Variable auf true gesetzt. Falls der Name gefunden wurde, wird eine Fehlermeldung ausgegeben.

In der registerUser()-Methode wird zuerst geschaut, ob der Name in der Datenbank vorhanden ist, dies geschieht mit einer TypedQuery. Falls die Liste leer ist, wird der Account in der Datenbank hinzugefügt mit den eingegeben Attributwerten. Jeder registrierte Benutzer bekommt die Benutzergruppe „Nobody“ zugewiesen. Wenn der Vorgang fehlschlagen sollte, wird dementsprechend eine Fehlermeldung ausgegeben.



Abbildung 8.11: Methode registerUser()

Die register\_button()-Methode wird ausgelöst, nachdem der Benutzer auf den Button „Registrieren“ geklickt hat. Diese Methode überprüft die eingegebenen Attributwerte auf ihre Richtigkeit und danach wird die registerUser()-Methode aufgerufen. Nach einer erfolgreichen Registrierung gelangt der Benutzer auf die Login-Seite. Andernfalls bekommt die Register-Seite die Fehlermeldungen für die entsprechenden Attributwerte angezeigt.

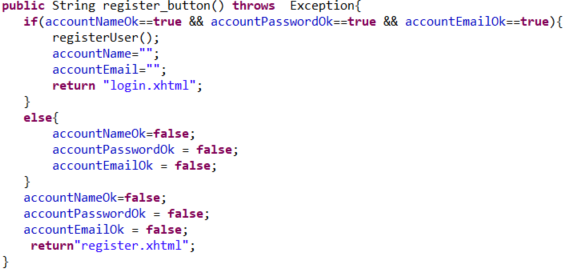


Abbildung 8.12: Methode register\_button()

## 8.4 LoginController

Nachdem registrieren von einem Benutzer kann dieser sich in der Login.xhtml-Seite anmelden und je nachdem welche Berechtigung der Benutzer hat auf das System zugreifen. Beim Anmelden des vorhandenen Accounts, wird der Benutzername auf seine Richtigkeit geprüft und dementsprechend auf seine dazugehörige Benutzergruppe zugeteilt. Der Benutzername wird wie im RegisterController durch eine Methode gefunden. Falls der Accountname nicht gefunden wird, bekommt der Benutzer eine Fehlermeldung ausgegeben.

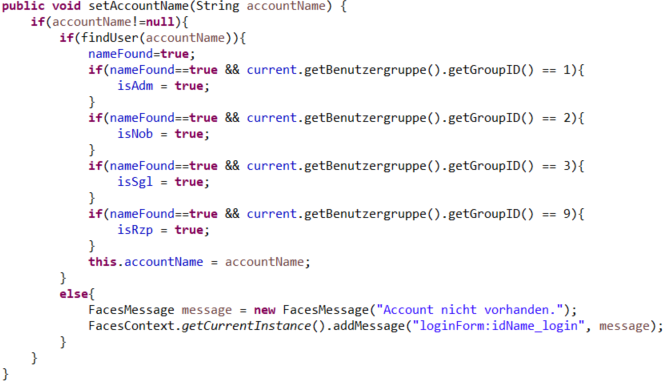


Abbildung 8.13: Setter für AccountName()

Das eigegebene Password wird mit der Methode checkPwd() überprüft, ob einmal der Benutzername vorhanden ist und das dazugehörige Password richtig eingegeben wurde.

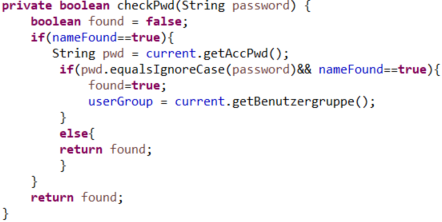


Abbildung 8.14: Methode checkPwd()

Wenn der Login-Button ausgelöst wird und der Benutzername mit dem jeweiligen Password richtig ist, dann gelangt der Benutzer auf die index.xhtml-Seite. Falls das nicht der Fall ist, bleibt der Benutzer auf der Seite und bekommt eine Fehlermeldung.

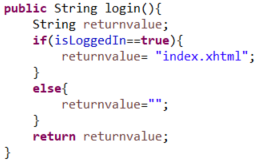


Abbildung 8.15: Methode login()

Nach dem Anmelden sieht der Benutzer auf der linken Seite in dem Menü, die Seiten auf die er berechtigt ist und den „Abmelden“ Button. Beim Abmelden des Benutzers wird die Session beendet und der Benutzer wird wieder auf die Anmeldeseite weitergeleitet.

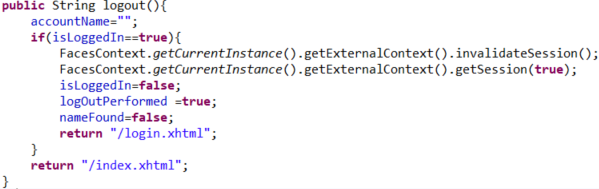


Abbildung 8.16: Methode logout()

Jeder Benutzer kann sein Password auf der Accountdetails-Seite ändern, in dem er das neue Password zweimal eingegeben hat und mit Speichern bestätigt. Beim nächsten Anmelden des Accounts müsste der Benutzer sein neues Password eingeben um sich anzumelden.

## 8.5 ScheduleControllerCopy

Um den Benutzern der Applikation die Erstellungen der Stundenpläne einfacher zu gestalten, ist die Überlegung vorherige Stundenpläne auf das neue Semester zu übertragen. Da sich an den Stundenplänen für das jeweilige Semester nur geringfügig etwas ändert, ist das Bearbeiten der Stundenpläne schneller als neue zu erstellen.   
Der ScheduleControllerCopy dient dazu um vorherige Stundenplaneinträge auf das neue Semester zu kopieren.   
In der init()-Funktion wird die selection()-Methode aufgerufen, in der die Listen beim Starten der Applikation geladen werden.   
In der copy()-Methode werden zwei Listen aus der Datenbank geladen, diese sind einmal der Stundenplan von den zu kopierenden Einträgen und falls vorhanden in dem Ziel Stundenplan.  
Danach werden die Einträge in dem Ziel Stundenplan gelöscht, da ansonsten die Einträge doppelt vorhanden wären, sofern Einträge existieren. Anschließend wird der Quell-Stundenplan in dem Ziel-Stundenplan übernommen. Zum Schluss bekommt der Benutzer eine Informationsmeldung, ob der Vorgang erfolgreich durchgeführt wurde oder nicht.



Abbildung 8.17: Methode copy()

Die delete()-Methode löscht alle Einträge aus dem Ziel Stundenplan, falls dort Einträge vorhanden sind.

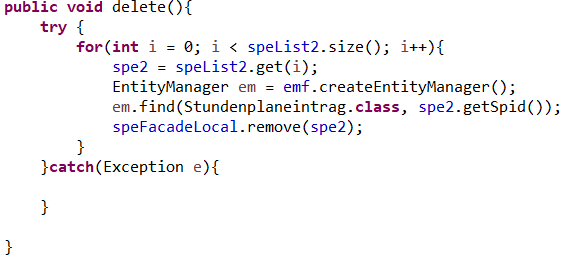


Abbildung 8.18: Methode delete()

Die create()-Methode holt sich die Liste, in der die Einträge vom Quell Stundenplan geladen wurde, und alle Daten, außer das Semester, werden aus dem jeweiligen Eintrag auf das ausgewählte Semester kopiert.



Abbildung 8.19: Methode create()

## 8.6 ScheduleControllerProf

In der aktuellen Version des Stundenplans von der Fachhochschule Bielefeld hat der Benutzer nur einen Überblick über alle Stundenpläne, die nach Studiengängen und den Semestern gefiltert sind.   
Die Überlegung in dem neuen Stundenplan ist, dass alle Professoren einen Einblick auf den Stundenplan von einem beliebigen Professor haben. Durch diese Funktion können alle Professoren profitieren und sich den eigenen Stundenplan anzeigen lassen. Es muss nicht unbedingt der eigene Stundenplan angezeigt werden, sondern es kann auch nach anderen Professoren gesucht werden, um zum Beispiel einen freien Zeitraum mit einem Professor auszumachen.  
  
In der init()-Funktion wird die selection()-Methode aufgerufen, in der die Listen beim Starten der Applikation geladen werden. Zusätzlich werden alle Events für das entsprechende Jahr, Semester und Professor geladen. Ein Event ist ein Eintrag in einem Stundenplan.

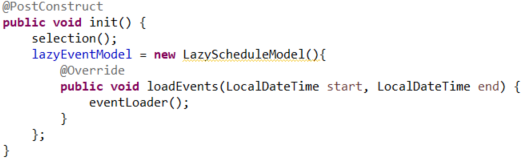


Abbildung 8.20: Methode init()

Die eventLoader()-Methode lädt alle Events mit der Abhängigkeit von der Stundenplansemester-ID und Professor-ID in eine Liste. Diese erfolgt durch eine Named-Query, die sich in dem Model Stundenplaneintrag befindet. Danach wird die Liste komplett durch iteriert, wodurch jeder Eintrag in ein Event umgewandelt wird. Jedes Event wird in einem LazyEvent Model hinzugefügt und in dem Stundenplan angezeigt.



Abbildung 8.21: Methode eventLoader()

Bei Veränderungen der Suchkrieterien bzw. beim Betätigen des Update-Buttons werden alle Einträge in dem LazyEvent Model gelöscht und die eventLoader()-Methode aufgerufen. Dadurch werden die Events mit den neuen Kriterien geladen.

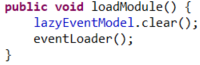


Abbildung 8.22: Methode loadModule()

In einem Stundenplan ist es in den meisten Fällen üblich, dass die Woche nur von Montag bis Freitag angezeigt wird. Da die Verbunds-Studiengänge auch am Samstag Unterricht haben, kann der Benutzer mit dem Button „Wochenenden“ zusätzlich noch die Wochenendtage anzeigen lassen. Dies geschieht durch die weekendChange()-Methode.

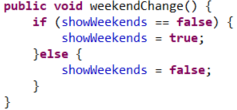


Abbildung 8.23: Methode weekendChange()

## 8.7 ScheduleController

Der ScheduleController ist das Kernstück der Stundenplan Applikation. Mit diesem Controller werden die Events bzw. Stundenplaneinträge für die jeweiligen Stundenpläne erstellt und bearbeitet. Beim Starten der Seite wird zuerst die init()-Methode aufgerufen, diese aktualisiert die Datenbank Einträge für die Stundenplaneintrag Tabelle. Danach werden die ganzen Listen geladen, die für den ScheduleController gebraucht werden. Anschließend werden acht neue Stundenpläne erstellt, welche für die einzelnen Semester gelten. Zum Schluss werden alle Events bzw. Stundenplaneinträge in die jeweiligen Stundenpläne geladen.

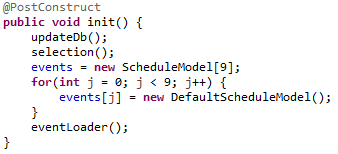


Abbildung 8.24: Methode init()

Wenn die Applikation neu gestartet wird, dann wird die Datenbank aktualisiert. Das heißt, dass bei allen Stundenplaneinträgen die Zeit auf die aktuelle Woche gesetzt werden. Das kommt daher, dass bei der „schedule“-Komponente in Primefaces die Events eigentlich fest verankert sind. Jedes Event hat ein festes Datum mit der dazugehörigen Uhrzeit. Mit der updateDB()-Methode werden die Events von Woche zu Woche verschoben, sodass es aussieht, dass es nur ein Abbild einer Woche ist.



Abbildung 8.25: Methode updateDB()

In der Stundenplaneintrag Tabelle werden alle Einträge mit der EJB-Methode „findAll“ gefunden und werden in der Liste loadDB gespeichert. Jeder Eintrag aus der Liste wird durch gegangen, um die Zeit anzupassen. Der Wochentag aus der Tabelle wird in einer String Variable gespeichert und danach in einer Enum Variable umgewandelt. Daraus wird zum Schluss der aktuellen tag berechnet. Dies geschieht mit der Methode getWeekdayDate(), hier werden alle Wochentage durchgegangen, bis der gewünschte gefunden wurde. Danach werden die Tabellenspalten SPEStartZeit und SPEEndZeit neu gesetzt. Falls irgendetwas an den Einträgen verändert wird, dann wird dementsprechend auch der Zeitstempel neu gesetzt.   
Die selection()-Methode lädt alle Listen für den „ScheduleController“ aus der Datenbank sowie die eventloader()-Methode alle Events aus der Datenbank für die einzelnen Stundenpläne lädt. Darauf wird in diesem Abschnitt nicht eingegangen, da diese in dem „ScheduleControllerProf“ beschrieben sind.   
Die Hauptfunktionen des ScheduleController sind neue Events erstellen, bearbeiten, löschen, verschieben und vergrößern bzw. verkleinern, damit wird die Zeit des Events angepasst.   
Wenn der Benutzer ein freies Feld in einem der Stundenpläne anklickt, wird ein Dialog aufgerufen und somit schon ein vorausgewähltes Event erstellt. Die Vorauswahl besteht darin, dass die Zeit übernommen wird, wo der Benutzer in den Stundenplan geklickt hat als Startzeit und eine Stunde später die Endzeit. Zusätzlich wir ein neues Objekt „eventSelected“ erstellt.

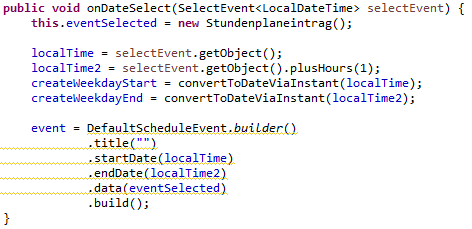


Abbildung 8.26: Methode onDateSelect()

In der addEvent()-Methode wird ein neues Event erstellt und anschließend dieses in der Stundenplaneintrag Tabelle hinzugefügt. Die Daten werden in einem Dialogfenster eingegeben und mit der EJB()-Methode „create“ in die Datenbank gespeichert. Danach werden alle Events gelöscht und neu geladen, dies geschieht mit der loadModule()-Methode. Falls ein neues Event erfolgreich hinzugefügt wurde, bekommt der Benutzer eine Informationsmeldung, ob der Vorgang Erfolgreich war oder nicht.



Abbildung 8.27: Methode addEvent()

Die loadModule()-Methode löscht alle vorhanden Events, jede Liste für das Exportieren eines Stundenplans und anschließend werden alle Einträge für die jeweiligen Stundenpläne neu geladen, dies geschieht mit der eventLoader()-Methode.

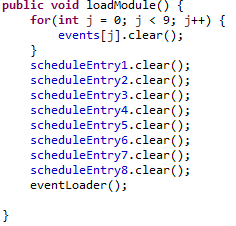


Abbildung 8.28: Methode loadModule()

Wenn ein Eintrag in einem Stundenplan ausgewählt wurde, dann werden alle Daten in das Objekt „eventSelected“ gespeichert. Danach werden alle Fremdschlüssel und die Uhrzeiten separat in Variablen gespeichert, das dient dazu, dass diese Daten in dem Dialogfenster zum Bearbeiten eines Eintrags angezeigt werden.

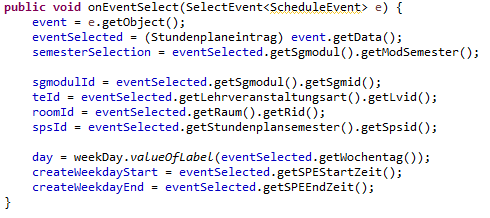


Abbildung 8.29: Methode onEventSelect()

Zum Bearbeiten eines Datensatzes wird in dem Dialogfenster für das bearbeiten die Daten geändert und anschließend auf den Button „Bearbeiten“ geklickt. In der Methode editEvent() werden alle geänderten Daten in das Objekt „eventSelected“ gespeichert und danach in die Datenbank mit der EJB-Methode „edit“ geladen. Zum Schluss wird eine Informationsmeldung ausgegeben, ob der Vorgang Erfolgreich war oder nicht.



Abbildung 8.30: Methode editEvent()

Mit der deleteEvent()-Methode wird das betroffene Event anhand der ID gefunden und dann gelöscht. Dies geschieht zuerst in der Datenbank, danach bekommt der Benutzer von der Applikation eine Informationsmeldung, ob der Vorgang erfolgreich war oder nicht. Zum Schluss wird der Eintrag aus der Liste „events“ herausgenommen.

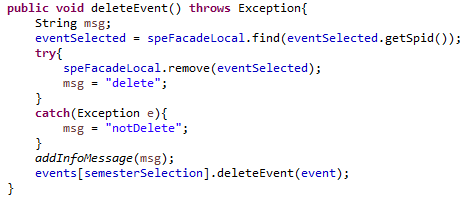


Abbildung 8.31: Methode deleteEvent()

Durch ziehen von Events können Events auf die gewünschte Position verschiebt werden, dies geschieht mit der onEventMove()-Methode. Die Methode speichert die vorhandenen Daten in die Variable eventSelected. Dann werden die Startzeit und Endzeit in extra Variablen gespeichert, diese werden mit der Verschiebungsdauer addiert, um so die neuen Zeiten zu berechnen. Die Zeiten werden hinterher wieder in Datums-Variablen umgewandelt, damit die Einheiten in der Datenbank und dem Programm gleich sind. Nachdem die neuen Zeiten berechnet worden sind, wird der entsprechende Eintrag anhand der ID gefunden und der Eintrag in der Tabelle aktualisiert. Es wird die Startzeit, Endzeit, der Wochentag und der Zeitstempel aktualisiert. Wenn Vorgang abgeschlossen ist, wird eine Informationsmeldung ausgegeben, ob der Vorgang Erfolgreich war oder nicht.



Abbildung 8.32: Methode onEventMove()

Die Methode onEventResize() hat die Funktion Events zu vergrößern und verkleinern, indem man auf den unteren Teil des Events klickt und nach oben oder unten zieht. Dadurch wird die Endzeit von einem Event beeinflusst. Die Methode berechnet sich aus dem Event die Verschiebungsdauer und rechnet diese auf die Endzeit. Die Endzeit wird danach in dem jeweiligen Event geändert, sowie auch in der Datenbank. Nachdem der Vorgang abschlossen ist bekommt der Benutzer eine Informationsmeldung, ob dies funktioniert hat oder nicht.

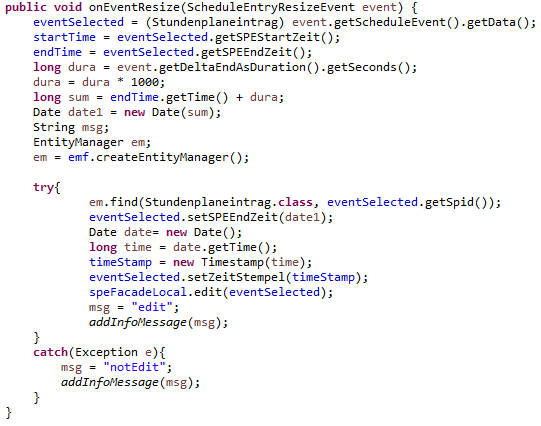


Abbildung 8.33: Methode onEventResize()

Die Benutzer können den jeweiligen Stundenplan ausdrucken oder in einer PDF Datei exportieren, dass können sie mit dem Button „Drucken“ oder „Exportieren“ erreichen. Hier wird ein neues Dialogfenster aufgerufen, dadurch bekommt der Benutzer die Auswahl, welchen Stundenplan derjenige von dem jeweiligen Studiengang gerne haben möchte.

## 9 Vergleiche zwischen Eclipse und Netbeans

In der folgenden Tabelle werden die Unterscheidungen von Netbeans und Eclipse aufgelistet.  
Auf der rechten Seite steht das besser benutzbare Programm.

|  |  |
| --- | --- |
| Unterscheidungen von Netbeans und Eclipse | Vorteil bei: |
| Generierung der Entities | Netbeans |
| Server Installation | Netbeans |
| Projekt Erstellung | Netbeans |
| Arbeitsfreundlich | Netbeans |
| MariaDB Connector | Netbeans |
| Neukompilierung | Netbeans |
| Einfügen von Maven Dependencies | Netbeans |
| Clean&Build des Projekts | Eclipse |
| Umfangreiche Projektarten | Eclipse |
| Übersichtliche Nutzungsweise | Netbeans |
| Generierung der EJBs | Netbeans |

Abbildung 9.1: Unterschiede zwischen Eclipse und Netbeans

Bei der Generierung der Entities aus der Datenbank ist Netbeans im Gegensatz zu Eclipse deutlich besser, denn die automatisch erstellten Java-Klassen haben alle NamedQueries aus den Attributen.   
In Eclipse wird nur die FindAll-NamedQuery generiert. Hinzu kommt, dass der Mapping Generator ausführlichere Annotationen und Attribute erstellt.   
Die Server Installation ist wieder bei Netbeans von Vorteil, weil in Netbeans die Installation des Servers nach dem Auswählen automatisch geschieht. Jedoch in Eclipse müssen die Pakete für die Server extra manuell installiert werden.   
Eclipse bietet eine Möglichkeit umfangreiche Projektarten zu erstellen, was für einige Nutzer von Vorteil sein könnte.   
Unter Netbeans ist die Projektauswahl viel übersichtlicher, da die einzelnen Projektarten zusammengefasst worden sind.   
Nach der Projekterstellung werden die persistence.xml, beans.xml, web.xml und facesconfig.xml von Netbeans automatisch erstellt. Im Gegensatz zu Eclipse, wo diese xml-Dateien in der Eigenschaft des Projekts extra ausgewählt und einige Einstellungen vorgenommen werden muss.   
Netbeans ist im Gegensatz zu Eclipse arbeitsfreundlicher, da Eclipse öfter einfriert und dementsprechend öfter warten muss oder das Programm des Öfteren abstürzt.   
Da das Projekt auf der Maria Datenbank basiert, wird in Netbeans der MariaDB Connector verwendet. Wobei in Eclipse der dazu kompatible MySQL Connector benutzt wird.   
Wenn das Projekt auf dem Server kompiliert wurde, kann bei Netbeans Änderungen an den xhtml-Seiten ohne neukompilieren des Projekts vorgenommen und auf dem Browser aktualisiert werden.   
Hingegen bei Eclipse jegliche Änderungen neu auf dem Server kompiliert werden muss.  
Beim Hinzufügen der Maven-Dependencies in der pom.xml reicht unter Netbeans das Speichern vollkommen aus um die Änderungen wirksam zu machen. Bei Eclipse muss dieser Vorgang noch mit einem Update des Projekts unter Maven abgeschlossen werden.  
Der Clean&Build Vorgang des Projekts funktioniert in Netbeans nur unter der Voraussetzung, dass der Server noch nicht gestartet worden ist. Wenn der Server aktiv ist, kann dort nur der Build Vorgang ausgeführt werden. In Eclipse ist der Vorgang nicht abhängig vom Server und kann immer durchgeführt werden.   
Bei der Generierung der EJBs ist Netbeans deutlich im Vorteil, da unter Netbeans der Grundaufbau aus den Entities automatisch generiert wird und diese darauf aufbauend mit Methoden erweitert werden kann. Bei Eclipse kann man die EJB Klasse erstellen, aber ohne Inhalt. Das heißt, dieser muss manuell Implementiert werden.

## 10 Vergleichen und Testen der Server

In der Projektvorbesprechung wurde vom Professor festgelegt, dass die Stundenplanapplikation auf den beiden Servern WildFly und GlassFish unter Eclipse getestet und verglichen werden soll. Um dies zu verwirklichen, wurde in den Abschnitten des Programms eine Zeitmessung für die Methoden zum Hinzufügen, Bearbeiten, Vergrößern, Verschieben und Löschen implementiert. Diese Methoden wurden jeweils mit den beiden Servern getestet und anschließend verglichen. Zusätzlich wurde auch gemessen, wie lange Eclipse braucht um den jeweiligen Server zu starten.

Für jede Aktion wurden jeweils zehn Messungen durchgeführt und in Jupyter Notebook ausgewertet. Die blauen Punkte stellen die Messungen von dem WildFly-Server und die roten für die Messungen des Glassfish-Servers dar.

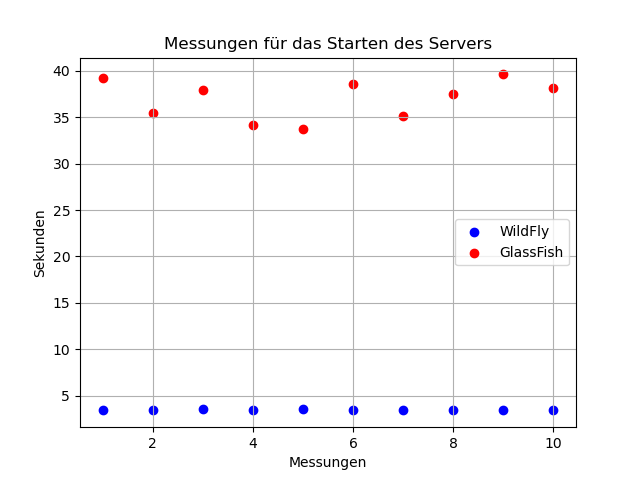


Abbildung 10.1: Messungen für das Starten des Servers

In der obigen Abbildung ist zu erkennen, dass sich die beiden Server beim Starten unter Eclipse in der Geschwindigkeit deutlich unterscheiden. Der WildFly-Server schwankt nur in einem kleineren Bereich zwischen einer Sekunde, dagegen der Glassfish-Server im größeren Bereich zwischen fünf Sekunden.

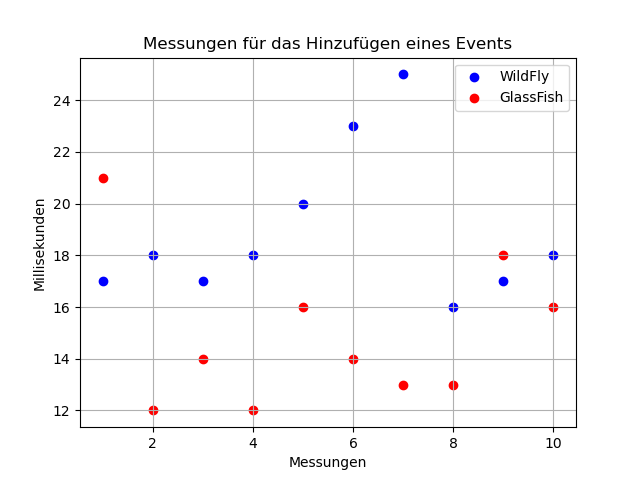


Abbildung 10.2: Messungen für das Hinzufügen eines Events

Beim Hinzufügen eines Events bei den jeweiligen Applikationsservern wird beobachtet, dass sich die beiden Server bei der Geschwindigkeit nicht deutlich unterscheiden. Da sich die Punkte oftmals überschneiden und die Messungen in Millisekunden stattfinden.

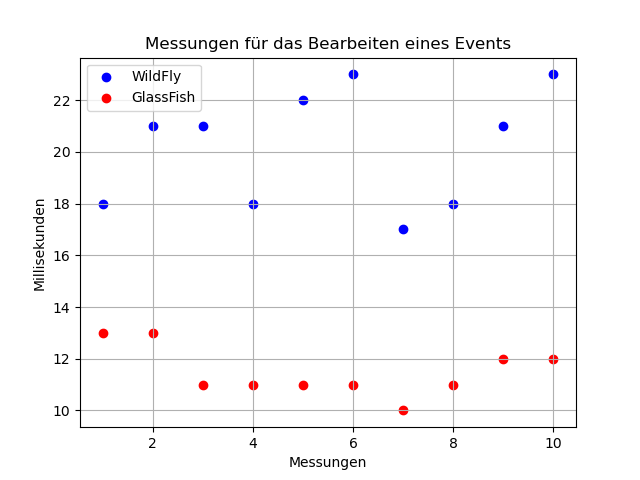


Abbildung 10.3: Messungen für das Bearbeiten eines Events

Bei dem Bearbeiten des Events wird deutlich sichtbar, dass der Glassfish-Server zwischen 10 und 13 Millisekunden und der WildFly-Server zwischen 17 und 23 Millisekunden schwankt.

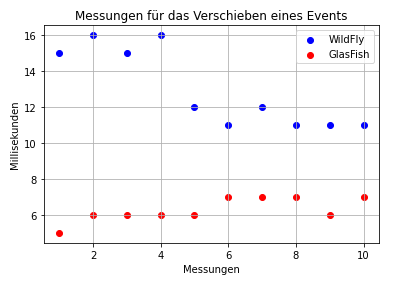


Abbildung 10.4: Messungen für das Verschieben eines Events

Auf dieser Abbildung wird auch verdeutlicht, dass sich die Server in der Geschwindigkeit beim Verschieben eines Events unterscheiden. Der GlassFish-Server verhält sich wieder in einem kleinen Bereich zwischen fünf und sieben Millisekunden und der WildFly-Server hingegen zwischen 11 und 16 Millisekunden.

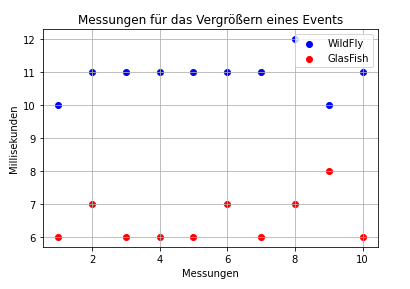


Abbildung 10.5: Messungen für das Vergrößern eines Events

Wie in der obigen Abbildung zu sehen ist, verhalten sich die Server in beiden Bereichen in zwei Millisekunden. Beim Verschieben und Vergrößern verhalten sich die Messungen ungefähr gleich, weil in beiden Fällen nur die Zeiten des Events verändert werden.

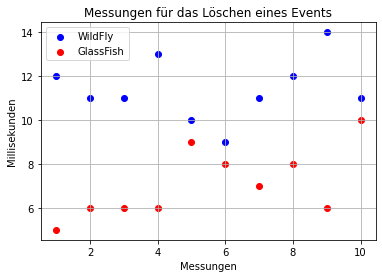


Abbildung 10.6: Messungen für das Löschen eines Events

Das Löschen eines Events verläuft ungefähr gleich wie das Hinzufügen eines Events, da sich die Server in der Geschwindigkeit wieder überschneiden und das Verhalten in größeren Bereichen abgebildet wird.

Aus den Messungen können wir schließen, dass die Geschwindigkeit bei den Methoden für die Kommunikation mit der Datenbank in der Applikation beim GlassFish-Server im Grunde schneller sind als die vom WildFly-Server, auch wenn es sich hierbei nur um Millisekunden handelt. Jedoch beim Starten des Servers und Hinzufügen der Applikation auf dem Server ist der WildFly-Server deutlich schneller als der GlassFish-Server. Da bei der Entwicklungszeit neue Implementierungen stattfinden, muss das Projekt immer wieder neu kompiliert und neu auf dem Server draufgespielt werden. Dies kostet bei dem GlassFish-Server-Nutzer viel mehr Zeit als bei dem WildFly-Server-Nutzer. Wo es bei dem WildFly-Server um die vier Sekunden dauert, dauert es bei dem GlassFish-Server 45 Sekunden, bis man auf die Weboberfläche zugreifen kann.

Bei der Konfiguration des WildFly-Servers ist diese ein wenig umfangreicher als bei dem GlassFish-Server. Der WildFly-Server muss vor der Konfiguration noch ein Benutzer angelegt werden und dann kann dieser auf die Administrator ebene zugreifen. Ansonsten müssen zusätzlich noch die Konnektoren hinzugefügt werden, da es nur wenige vorhanden sind. Bei dem GlassFish-Server ist es nur etwas umständlich den Server auf Eclipse zu installieren, da man dort zusätzliche Pakete installieren muss. Nachdem Starten des Servers kann man dann einfach auf die Administrations ebene zugreifen. Die Verwaltungsebene bei dem GlassFish-Server ist einfacher und übersichtlicher gehalten als bei dem WildFly-Server.  
  
Aus den genannten Argumenten wird daher für die Entwicklung eines Projekts ist der WildFly-Server empfohlen und für die Ausführung der Applikation der GlassFish-Server.

## 11 Fazit

In diesem Projekt wurde der Grundstein für die Webapplikation der Stundenpläne für die Fachhochschule Bielefeld erstellt, damit langfristig die Erstellung der Stundenpläne und somit der Zugriff von den Studenten einfacher als die vorhandene Version geschehen soll. Als erstes wurde eine neue Datenbank für den Stundenplan konzipiert und erstellt. Danach wurden die beiden Server aufgesetzt und beides in Eclipse verbunden. Zum Schluss wurde die Webapplikation programmiert und getestet, damit sich die aus den Aufgabenstellungen des Projekts ergebenen Problematiken gelöst werden konnte.

Nach der Fertigstellung der Webapplikation des Stundenplans wurden die Laufzeiten der beiden Server für das Starten des Servers und die Interaktionen zwischen dem Server und der Datenbank, die vom Benutzer in der Webapplikation ausgelöst worden sind, getestet. Daraus schließen wir, dass für die Entwicklung der Webapplikation der WildFly-Server besser geeignet ist und für die Benutzung der Webapplikation der GlassFish-Server.

Um auf die Fragen, die sich aus der Projekt-Vorbesprechung ergeben haben, zurückzukommen werden diese nachfolgend beantwortet.

*Sind die Stundenpläne anschließend inhaltlich gleichwertig?*  
Die Stundenpläne sind inhaltlich gleichwertig, jedoch können in dieser Webapplikation schneller und einfacher die Informationen für eine Veranstaltung abgerufen werden.  
  
*Werden bei der Nutzung des neuen Systems bisher nicht entdeckte Bugs gefunden?*Bisher sind keine Bugs der neuen Webapplikation aufgefallen.   
  
*Gibt es eine Effizienzsteigerung?*  
Nein, es gibt keine Effizienzsteigerung, weil es in dieser Webapplikation keine Konfliktbehandlung für die Erstellung einer Veranstaltung implementiert wurde.   
  
*Welche der beiden Server ist am sinnvollsten zu nutzen?*  
Unserer Meinung nach ist die Benutzung der WildFly-Server sinnvoller.

Die Verbesserungsvorschläge, die sich aus dem Projekt ergeben haben, sind einmal die komplexeren TypedQueries in EJB-CriteriaQueries umzuschreiben. Sonst könnte noch die Export- und Druckversion des Stundenplans verschönert werden. Das wichtigste wäre die Konfliktbehandlung, um schnell und einfach Stundenpläne zu erstellen ohne das Fehler auftreten. Beim Umschalten der Webapplikation von Deutsch auf Englisch muss das Bundle vervollständigt werden, da bisher nur die Messages drin sind. Zusätzlich könnte man beim Löschen eines Events eine Bestätigung implementieren, ob das Event wirklich gelöscht werden soll. Was noch fehlt sind die Fehlermeldungen in den Dialogkomponenten, diese könnte man ebenfalls programmieren. Für die Inhaltliche Verbesserung der Webapplikation wäre zum Beispiel das Pflegen der ganzen Portfolios und das Erstellen der Stundenpläne ratsam.

## 

## 12 Anhang

## 12.1 Konfiguration des WildFly Servers

Der WildFly Server wird für die Stundenplanapplikation verwendet, sodass diese auf dem Server gestartet werden kann und um auf die Applikation zugreifen zu können. Im Folgenden wird beschrieben, was nötig ist, um den Server einzubinden und hinterher auch starten zu können.

Zu allererst wird der WildFly Server heruntergeladen, dieser steht kostenlos im Internet zur Verfügung. Der WildFly Server braucht keine zusätzliche Installation, das heißt, dass dieser einfach an den gewünschten Ort abgespeichert werden kann.

Um auf die Konfiguration des Servers zugreifen zu können, wird ein Administrator Account benötigt. Dazu wird die Eingabeaufforderung (cmd) geöffnet und der Befehl „add-user“ eingegeben. Dadurch wird eine Batch-Datei ausgeführt, um ein Administrator Account zu erstellen.

Die folgenden Eingaben sind ein Beispiel:

What type of user do you wish to add?

a) Management User

b) Application User

Benutzereingabe: a

Username:

Benutzereingabe: dba

Password:

Benutzereingabe: dba

The password should be different from the username

Are you sure you want to use the password entered yes/no?

Benutzereingabe: yes

Re-enter Password:

Benutzereingabe: dba

What groups do you want this user to belong to? (Please enter a comma seperated list, or leave blank for none) [ ]:

Benutzereingabe: [Enter]

About to add user ‘dba‘ for realm ‘ManagementRealm‘

Is this correct yes/no?

Benutzereingabe: yes

Is this new user going to be used for one AS process to connect to another AS process?

e.g. for a slave host controller connecting tot he master or for a Remoting connection for server to server EJB calls.

yes/no?

Benutzereingabe: no

Benutzereingabe: [Enter]

Der WildFly Server wird mit dem Kommando „standalone“ gestartet, um auf die Administrator Konsole zugreifen zu können.Mit „localhost:8080“ wird die Startseite aufgerufen und mit „localhost:9990/console/index.html“ die Administrator Konsole.

Falls der gewünschte Connector für die Datenbank nicht vorhanden sein sollte, muss dieser um den erweitert werden. Unter dem Ordnerverzeichnis „C:\Program Files\wildfly-20.0.1.Final\modules\system\layers\base\com“ wird eine Ordnerstruktur in Form von mysql/mysql/main erzeugt. In diesem Ordner wird der Connector (mysql-connector-java-5.1.48.jar) und eine module.xml hinzugefügt. Der Inhalt der module.xml sieht wie folgt aus:

<?xml version="1.0" encoding="UTF-8"?>

<module xmlns="urn:jboss:module:1.5" name="com.mysql.mysql">

<resources>

<resource-root path="mysql-connector-java-5.1.48.jar"/>

</resources>

<dependencies>

<module name="javax.api"/>

<module name="javax.transaction.api"/>

<module name="javax.servlet.api" optional="true"/>

</dependencies>

</module>

Für das Erstellen eines Treiber und Datenquelle auf dem WildFly Server wird der Reiter Configuration/Subsystems/Datasource & Drivers geöffnet. Der Treiber wird mit dem Button „Add JDBC Driver“ hinzugefügt. Folgende Eingaben und Schritte sind zu tun, um einen Treiber hinzuzufügen:

Driver Name\*: stundenplanung

Driver Module Name\*: com.mysql.mysql

Module Slot:

Driver Class Name: com.mysql.jdbc.Driver

Driver Datasource Class Name:

Driver XA Datasource Class Name:

Für die Datenquelle wird der Reiter „Datasources“ aufgerufen und auf den Button „Add Datasource“ geklickt. Folgende Eingaben sind nötig um dies zu tun:

Choose Template:

Benutzereingabe: Custom

Benutzereingabe: [Next]

Attributes:

Benutzereingabe: Name\*: stundenplanung

Benutzereingabe: JNDI Name\*: java:/jdbc/stundenplanung

Benutzereingabe: [Next]

JDBC Driver:

Benutzereingabe: Driver Name\*: stundenplanung

Benutzereingabe: Driver Module Name:

Benutzereingabe: Driver Class Name:

Connection:

Benutzereingabe: Connection URL: jdbc:mysql://localhost:3306/stundenplanung

Benutzereingabe: User Name: [Name des Benutzers in der Datenbank]

Benutzereingabe: Password: [Passwort des Benutzers in der Datenbank]

Benutzereingabe: Security Domain:

Benutzereingabe: [Next]

Test Connection:

Benutzereingabe: Test Connection

Benutzereingabe: [Next]

Review:

Benutzereingabe: [Finish]

Um den WildFly Server in Eclipse einzubinden, wird die zusätzliche Bibliothek „JBoss Tools 4.17.0.Final“ benötigt. Diese steht in dem „Marketplace“ zur freien Verfügung. Unter dem Reiter „Servers“ wird der WildFly Server hinzugefügt, dieser steht bei der „JBoss Community“. Dort wird der passende WildFly Server ausgewählt und den Anweisungen gefolgt. Es werden keine zusätzlichen Eingaben gebraucht.

## 12.2 Installation des Glassfish Servers in Eclipse

Um den Glassfish Server in Eclipse mit einzubinden, wird ein extra Paket zum Installieren benötigt.   
Diese „Eclipse Sapphire“ Installation geschieht über dem Menüpunkt Help🡪Install New Software…  
Danach öffnet sich das folgende Fenster:

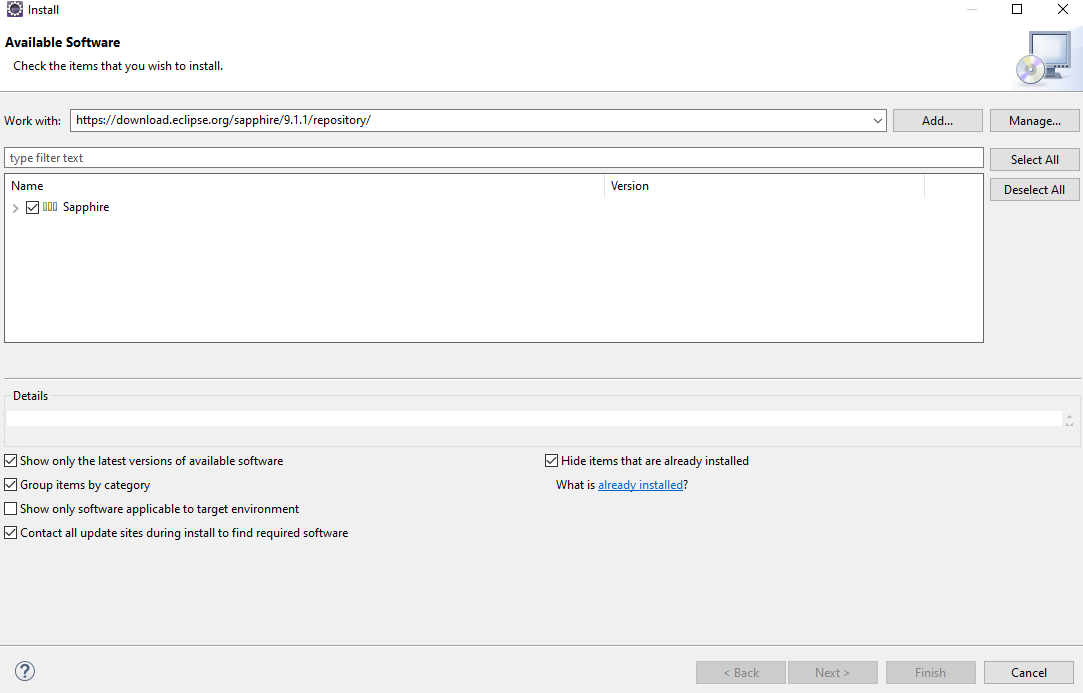


Abbildung 12.1: Sapphire Software

In dem Feld rechts neben Work with sollte die URL „https://download.eclipse.org/sapphire/9.1.1/repository/“ eingegeben und mit Enter bestätigt werden. Es sollte nach der Bestätigung nach dem Paket gesucht und automatisch in der Liste angezeigt werden. Wie auch im Obigen Bild zu sehen, erscheint dort die Auswahl für die „Sapphire“ Installation. Da in unserem Fall das Paket schon installiert wurde, sind die Next und Finish Balken nicht anzuklicken. Wenn das Paket installiert werden soll betätigt man dies mit Finish und den Rest macht Eclipse automatisch.

Nachdem das Sapphire auf Eclipse installiert wurde, wird nun auf gleicherweise das Eclipse GlassFish Tools installiert. Also wieder im Menüpunkt Help🡪Install New Software…

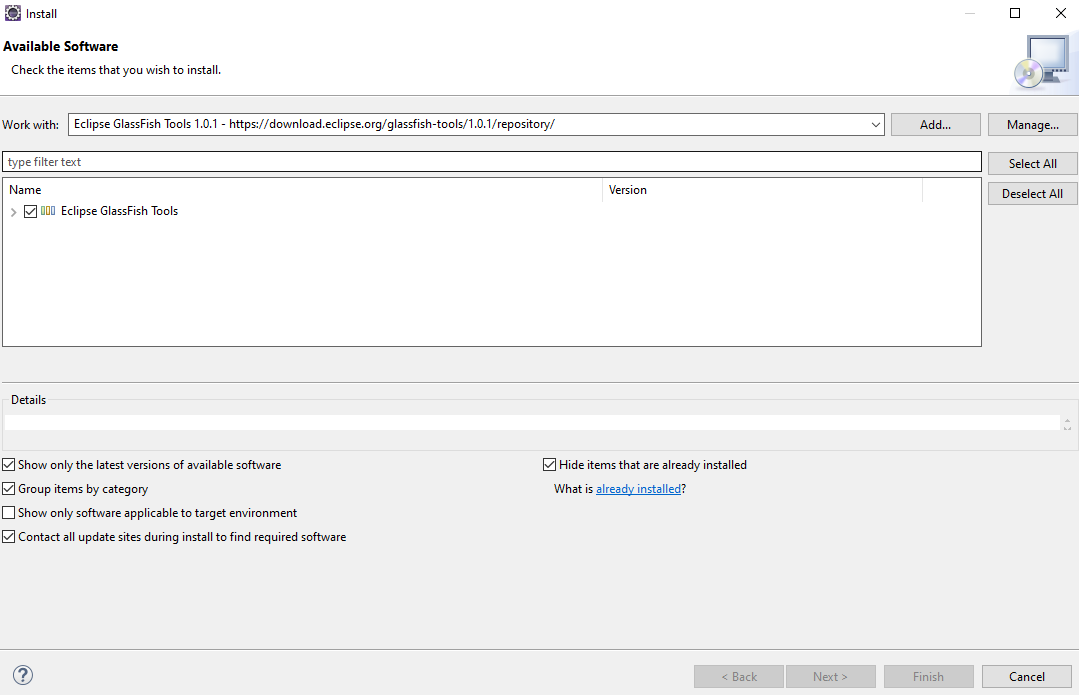


Abbildung 12.2: Eclipse GlassFish Tools Software

In dem Feld Work With wird die URL(„https://download.eclipse.org/glassfish-tools/1.0.1/repository/“) für das Installieren des Applikationsservers eingegeben und mit Enter bestätigt. Das zu installierende Pakete wird gesucht und in der Liste angezeigt. Mit Finish wird das Paket direkt Installiert, aber hier sollte der Weg mit Next genommen werden, denn wir sollten den Pfad für die Installation des Servers bestimmen um ihn nach der Installation für die Konfiguration wieder zu finden.

## 12.3 Einrichtung GlassFish Server

Nachdem die erforderlichen Pakete Installiert sind, wird nun der GlassFish Server in Eclipse eingerichtet. In Eclipse sollte unten ein Reiter Namens „Servers“ befinden, falls nicht ist es im Menüpunkt Window🡪Show view🡪Servers zu finden. Dort wird in dem leeren weißen Feld ein Rechtsklick ausgeführt und New🡪Server ausgewählt.

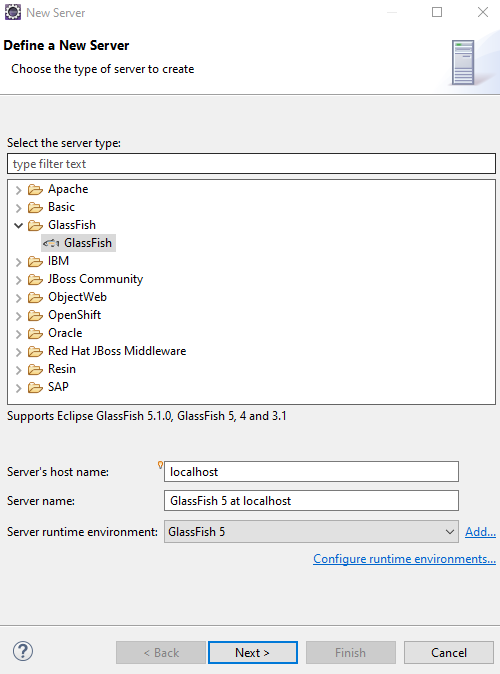


Abbildung 12.3: Server Definieren

Wie im obigen Fenster zu sehen sind viele Server Typen aufgelistet.Aus der Lister wird GlassFish ausgewählt und auf Next klicken.

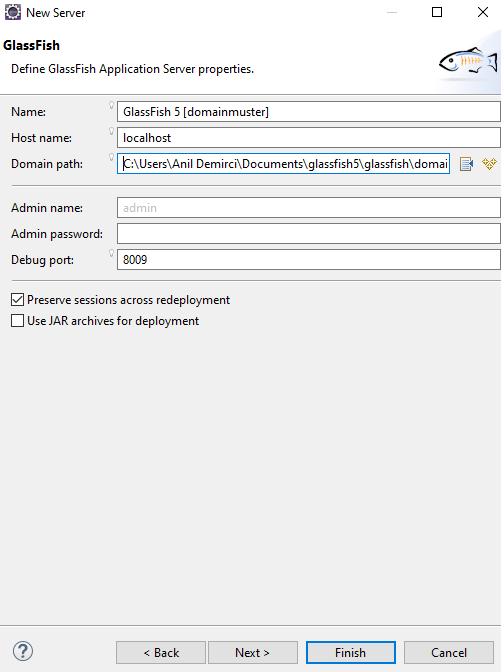


Abbildung 12.4: GlassFish Server Eigenschaften

Nachdem klicken auf Next werden einige Eigenschaften zum GlassFisch Server angezeigt, wie der Name des Servers, Hostname, der Domain-Pfad und Verbindungszugangsdaten. Im Normalfall wird hier nichts geändert und mit Finish wird der Server bestätigt und erstellt.  
Bevor der Server gestartet wird um die Verbindung zur Datenbank herzustellen, muss der MySQL Connector in dem Server Pfad unter glassfish🡪lib platziert werden.  
Wenn die Datenbank zur Verfügung steht, bzw. der Apache und MySQL server im XAMPP Control Panel gestartet sind, wird auch nun der Glassfish Server gestartet.

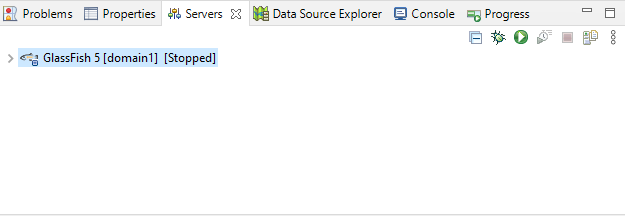


Abbildung 12.5: Server Fenster

In dem Reiter Servers befindet sich der erstellte GlassFish Server. Dieser wird mit dem grünen Play Symbol oder mit einem Rechtsklick Start ausgewählt und gestartet. Nach einem erfolgreichen Start wird der Server wieder mit der rechten Maustaste angeklickt und Glassfish🡪View Admin Console ausgewählt.

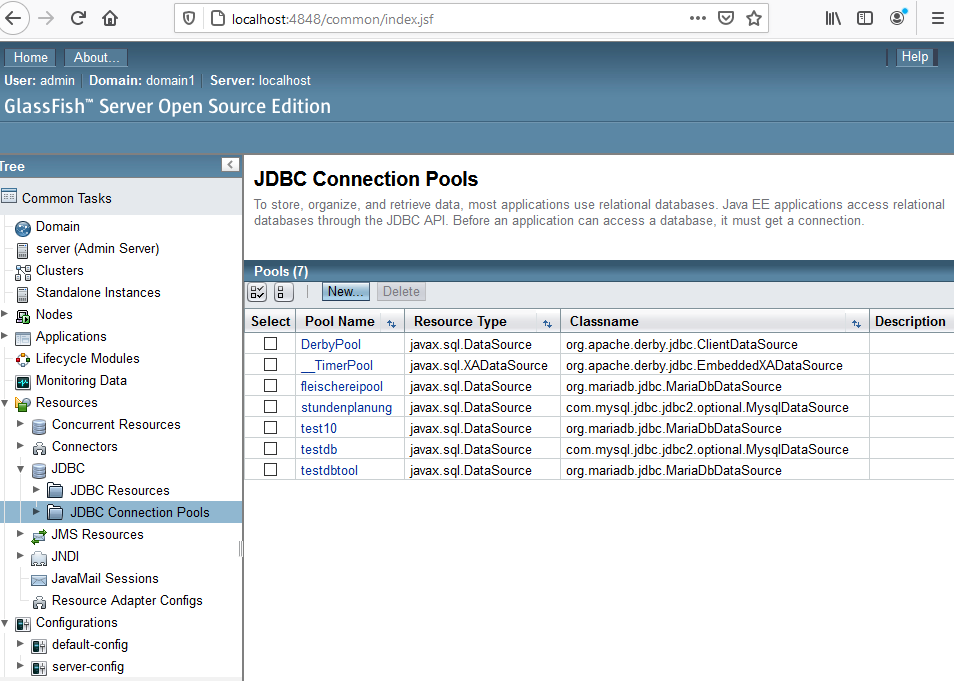


Abbildung 12.6: JDBC Connection Pools

Es öffnet sich die Administrations-Console in dem Browser. In dem Reiter Resources🡪JDBC🡪JDBC Connection Pools wird mit New… ein neues Verbindungs-Pool eingerichtet.

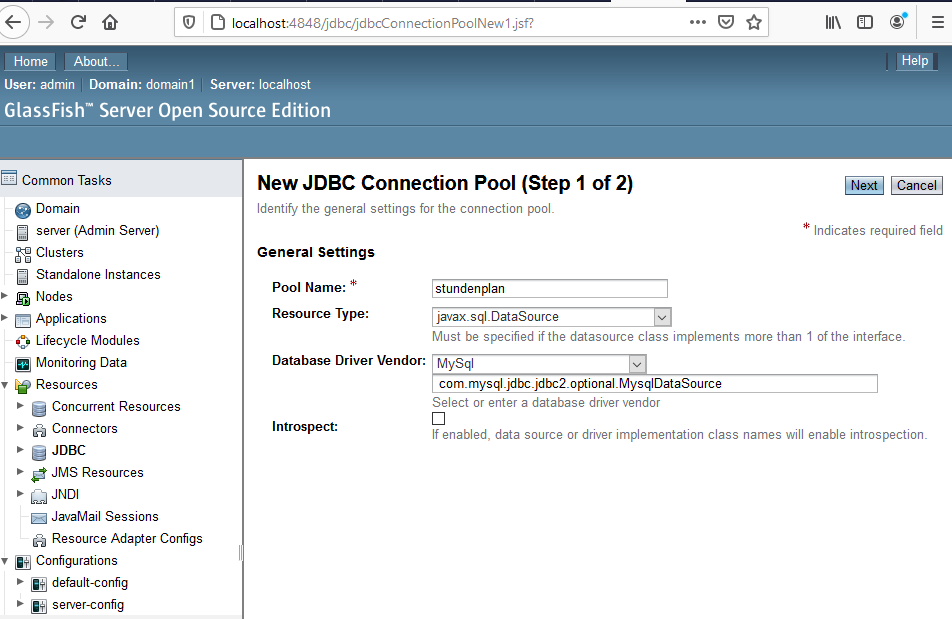


Abbildung 12.7: Neue JDBC Verbindung herstellen Seite 1

Im Pool Name wird ein beliebiger Name eingegeben, aber um Namenskonflikte zu vermeiden wird hier standartgemäß der Datenbankname eingetragen. Als Resource Type wird javax.sql.DataSource ausgewählt. Der Database Drive Vendor ist MySQL und dazu unten in dem Feld com.mysql.jdbc.jdbc2.optional.MysqlDataSource eingegeben und weiter mit Next angeklickt.

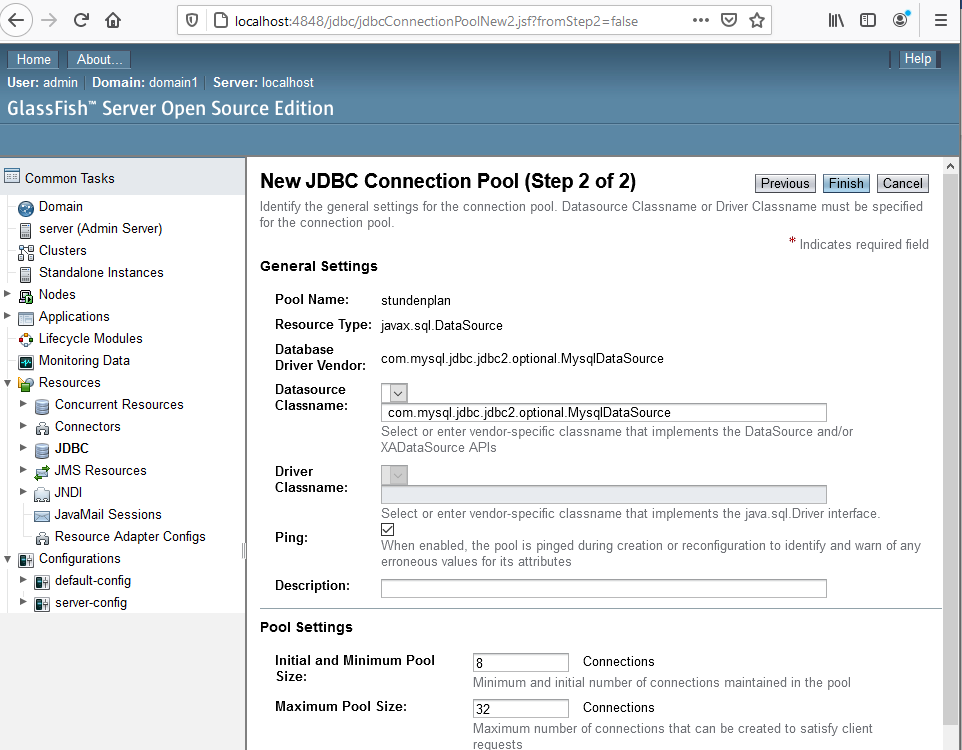


Abbildung 12.8: Neue JDBC Verbindung herstellen Seite 2

In dem zweiten Schritt wird bei Datasource Classname wieder die „com.mysql.jdbc.jdbc2.optional.MysqlDataSource“ Data Source eingetragen und die Seite nach unten zu Additional Properties gescrollt.

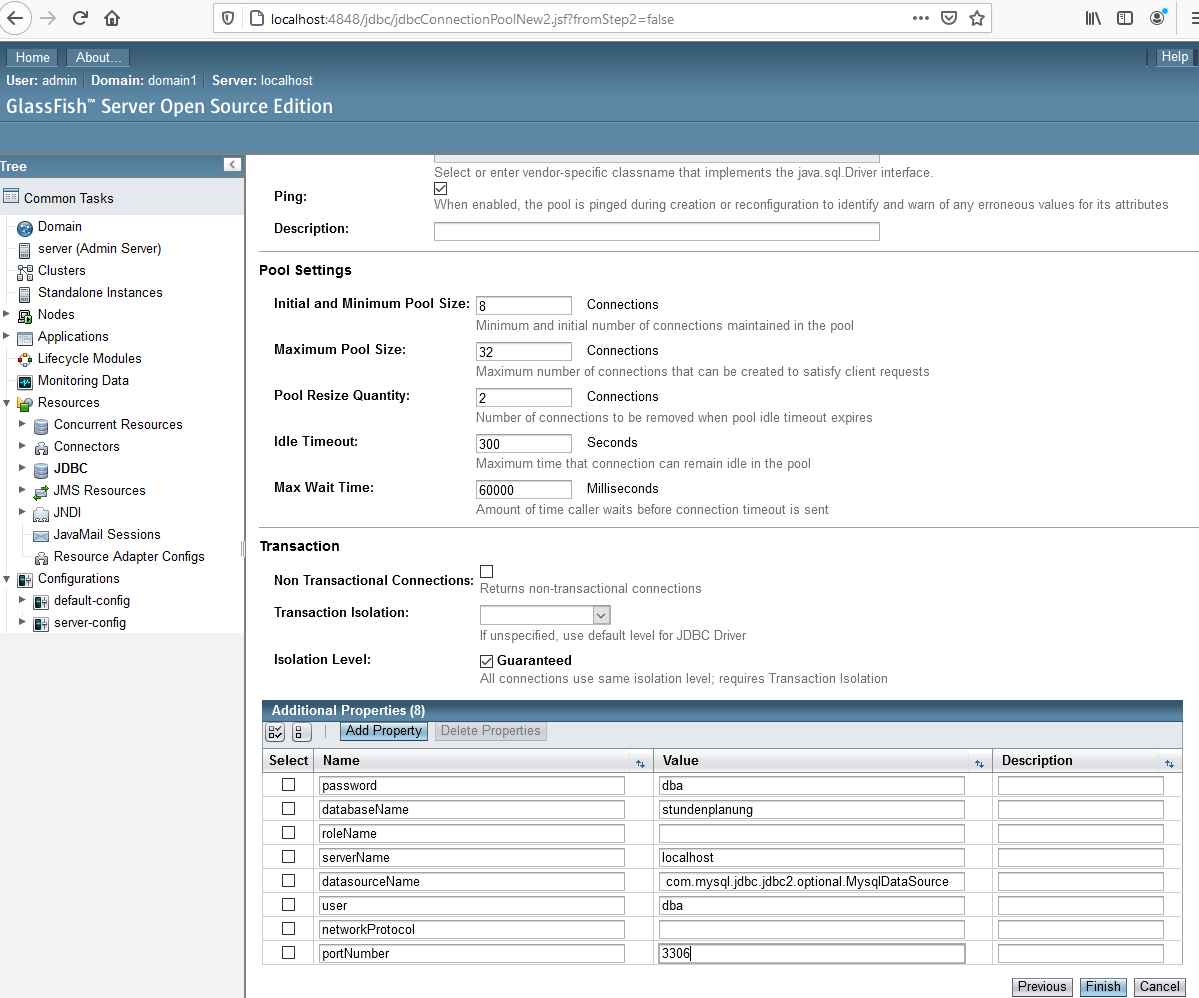


Abbildung 12.9: JDBC Eigenschaften

In den Additional Properties werden die Verbindungsdaten zur Datenbank eingegeben die im phpMyAdmin erstellt wurden.

Password🡪dba  
databaseName🡪stundenplanung  
serverName🡪localhost  
datasourceName(optional)🡪 com.mysql.jdbc.jdbc2.optional.MysqlDataSource  
user🡪dba  
portNumber🡪3306

Mit Finish wird der JDBC Connection Pool eingerichtet und wenn man den Hacken bei Ping gesetzt hat wird direkt versucht eine Verbindung mit der Datenbank herzustellen.

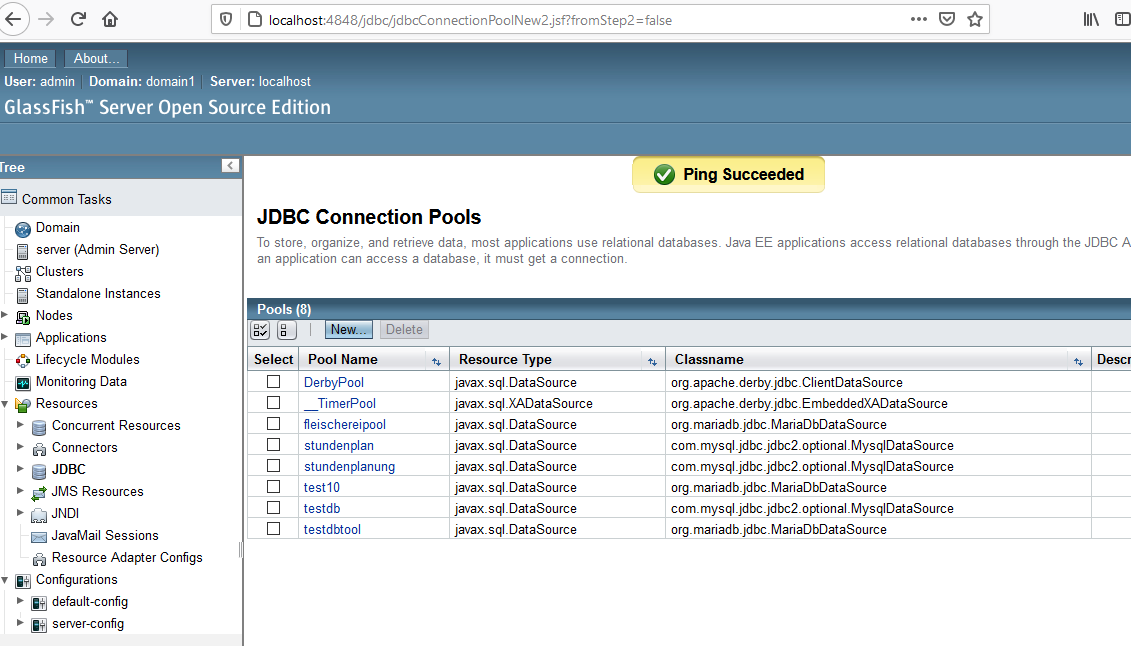


Abbildung 12.10: JDBC Ping

Nachdem alles richtig eingetragen wurde, sollte wie im obigen Bild der grüne Hacken mit Ping Succeded stehen für die Meldung, dass die Verbindung erfolgreich war.

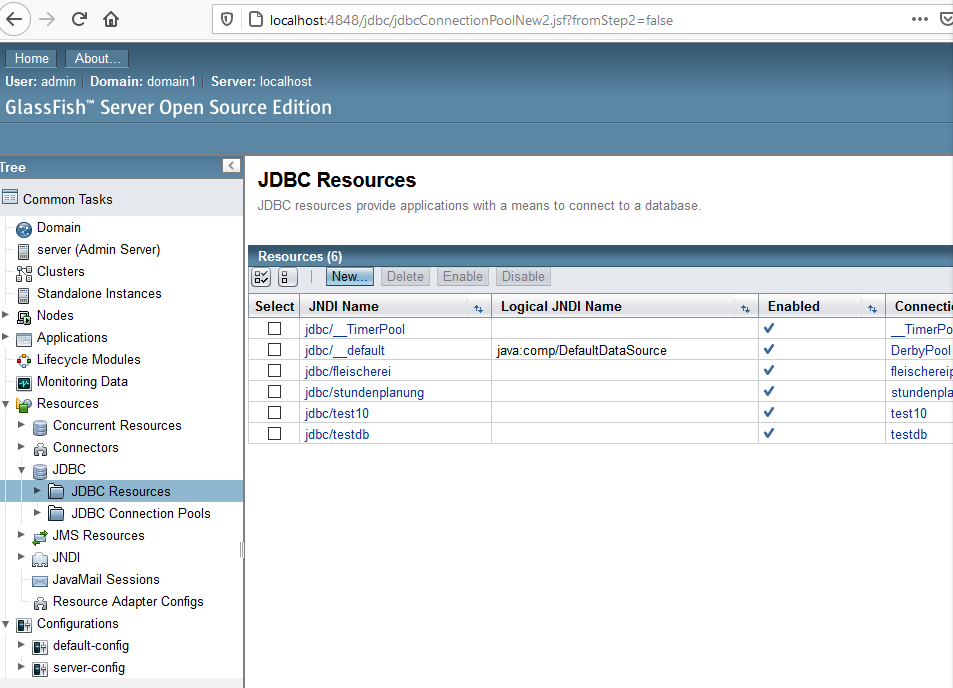


Abbildung 12.11: JDBC Ressourcen

Wenn dies der Fall ist wird in dem Reiter Resources🡪JDBC🡪JDBC Ressources mit New.. eine JDBC Verbindung für den Connection Pool eingerichtet.

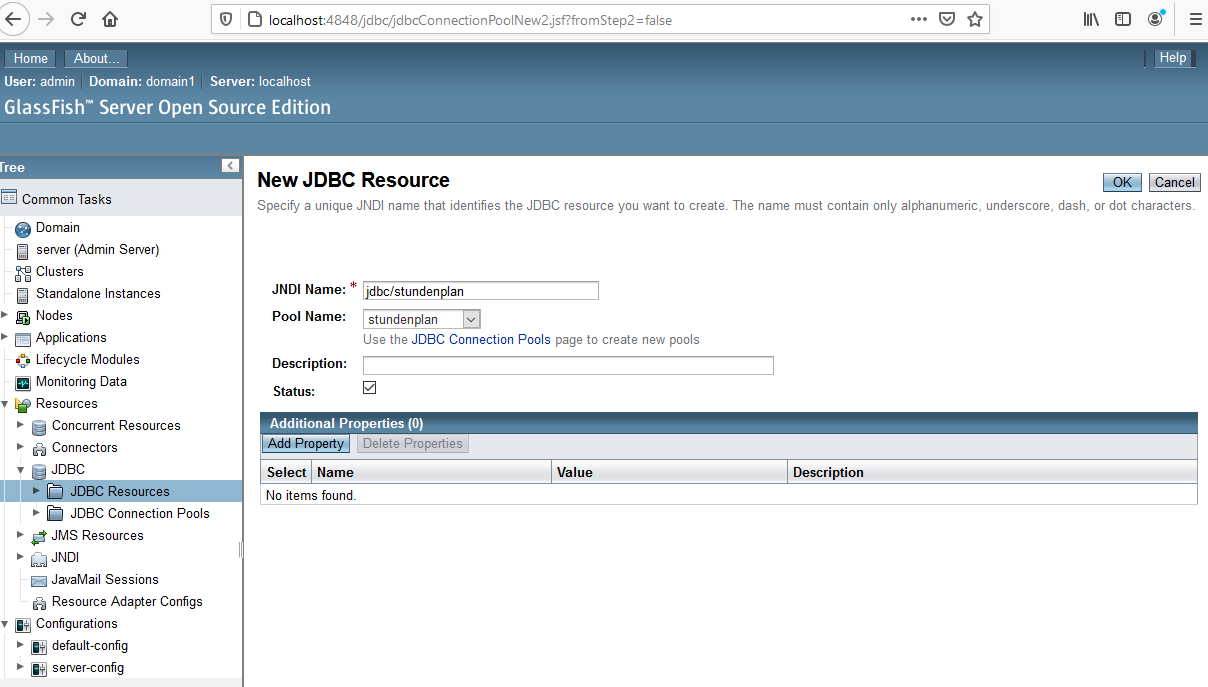


Abbildung 12.12: JDBC JNDI Name

Im JNDI Name kann ein beliebiger Name eingegeben, aber standartgemäß wird jdbc/\*Datenbankname\* eingetragen, also in dem Fall jdbc/stundenplan. Diese JNDI wird nachher auch in der persistence.xml unter JTA-Data-Source eingetragen. Es ist daher wichtig dies zu merken oder zu notieren. Als Pool Name wird der im vorherigen Schritt erstellte Connection Pool ausgewählt und mit OK erstellt.

1. Vergleichen/Testen:  
   -Unterschiede Wildfly und Glassfish (Darauf eingehen was anders ist in sätzen und dann auf konfig. verweisen)  
   -Wildfly/Glassfish: Zeiten: Compilieren, Debuggen, Laufzeiten

-Eclipse/Netbeans: Entities, JPA, EJB, ...

2) Konfigurationen der Server in Eclipse und deren Verbindungen durch JDBC  
 - Wildfly und Glassfish

3) MySQL Connector und Einstellungen fürs Verbinden mit der Datenbank durch Eclipse

4) Wie man einen Projekt in Eclipse erstellt und Extra einstellungen   
 - Librarys die wir benuten z.b. Eclipselink und wofür die Sind  
 - Java Build Path ( Java version )   
 - JPA version aktiv bzw in den Properties ausgewählt um die Persistence.xml zu generieren  
 - Welche pakete müssen in Eclipse installiert werden z.b. für xhtml datein  
 - Java Compiler auf 1.8 einstellen  
 - Project Facets beschreiben mit den jeweiligen Versionen  
 - bean erstellen  
 - wie bekommen wir die Entitys aus der Datenbank in das Projekt (persistence.xml)

5) Einleitung und Ziel des Projekt

- Was möchten wir haben?   
 - Datenbank-Struktur und beschreibung

6) Hauptteil (Punkt 1-4)

7) Diagramm für Use Cases (wer hat welche berechtigung) und die ausgabe der menüpunkte. Bzw aufbau Template

8) Methoden beschreibung   
 - Verwendung von EJBs  
 - Welche möglichkeiten gibt es zum persistieren und abfragen (usertransaction und EJB) welchen weg haben wir benutzt (z.b. einfache querys mit EJB, komplexe mit typedqueries) EJB wird automatischgesteurt 🡪 sicherer

- Methoden in andere Controller beschreiben (Modul,Benutzergruppe,Account etc)  
 - Welche funktionen wir geschrieben haben  
 - ScheduleController und xhtml Beschreibung   
 - web.xml beschreiben (welcome file (Startseite), session time, primefaces theme jars in pom.xml und in web-inf library)  
  
9) Bundle und Javadoc verwendung

10) Fazit  
 - Was haben wir erreicht und was fehlt uns noch bzw welche Verbesserungen man machen könnte.