1. Vergleichen/Testen:  
   -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 Projekts  
 - Was möchten wir haben?   
 - Datenbank-Struktur und beschreibung

6) Hauptteil (Punkt 1-4)

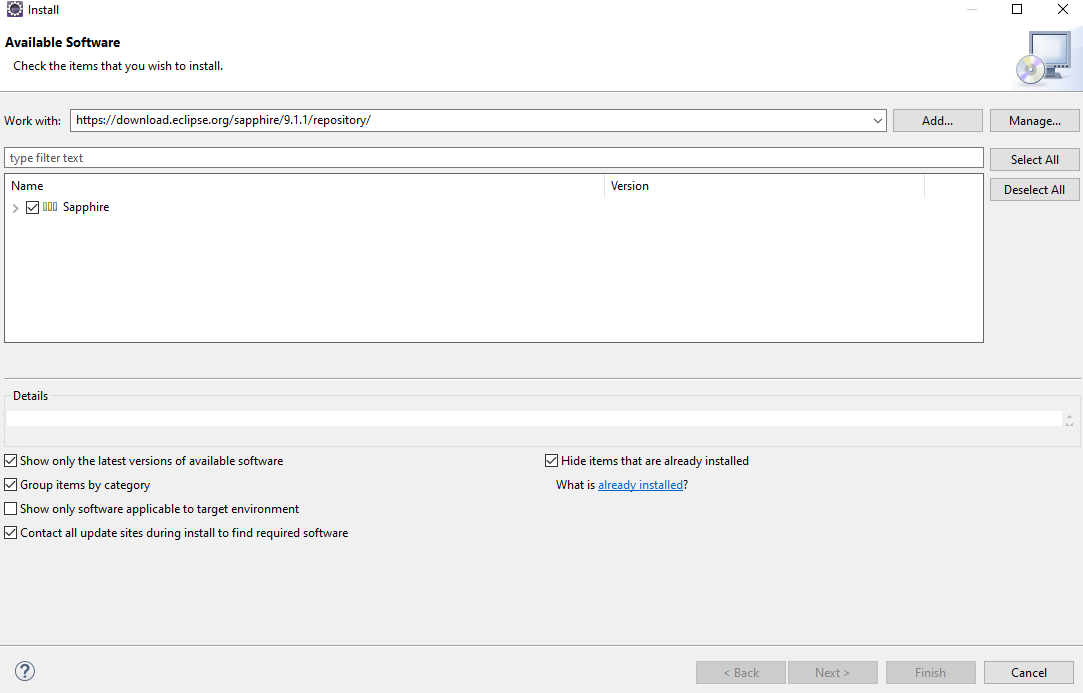
7) Use Cases (wer hat welche berechtigung) und die ausgabe der menüpunkte. Bzw aufbau Template

8) Methoden beschreibung   
 - Verwendung von EJBs  
 - Welche funktionen wir geschrieben haben  
 - Controller 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.

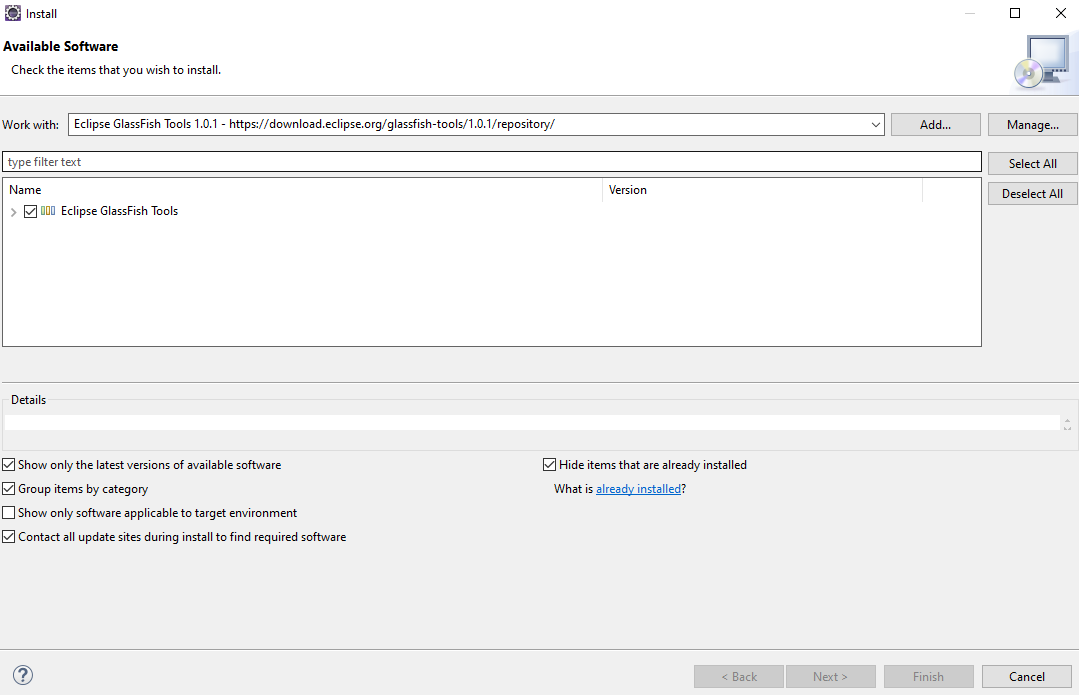
Installation des Glassfish Servers in Eclipse

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:



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.

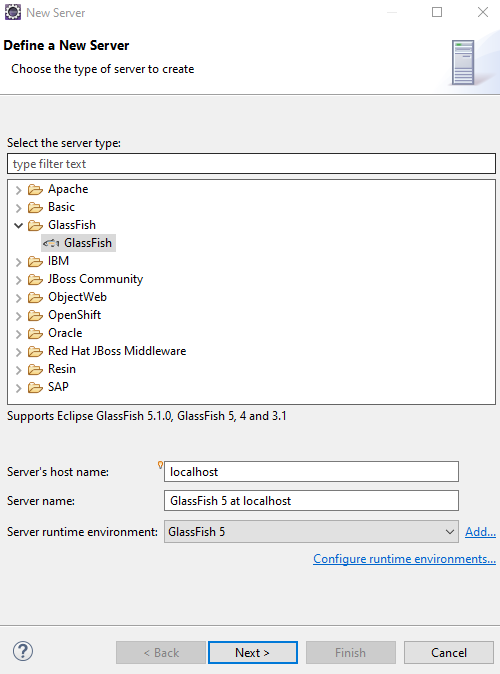
Nachdem das Sapphire auf Eclipse installiert wurde, wird nun auf gleicherweise das Eclipse GlassFish Tools installiert. Also wieder im Menüpunkt Help🡪Install New Software…



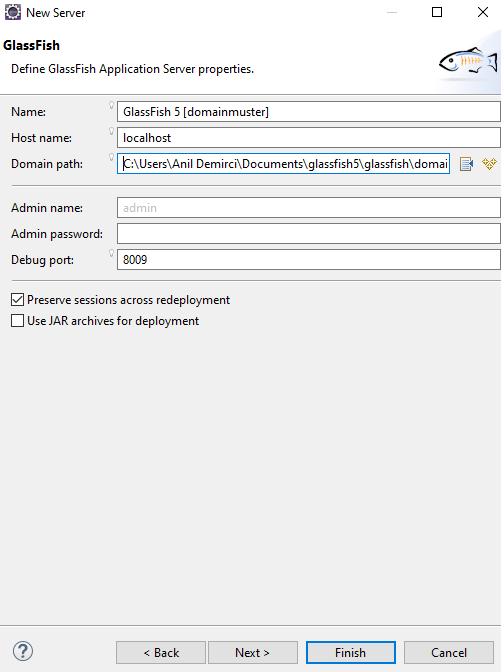
In dem Feld Work With wird die URL(„https://download.eclipse.org/glassfish-tools/1.0.1/repository/“) für das Installieren des Applikationsservers eingegeben und mit Enter bestätigt. Das zu installierende Pakete wird gesucht und in der Liste angezeigt. Mit Finish wird das Paket direkt Installiert, aber 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.

Einrichtung GlassFish Server

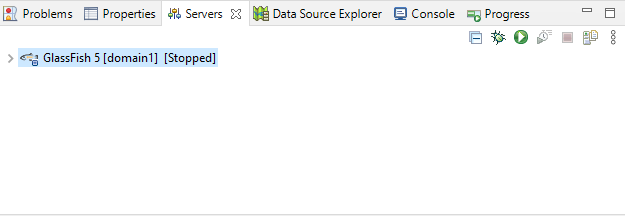
Nachdem die erforderlichen Pakete Installiert sind, wird nun der GlassFish Server in Eclipse eingerichtet. In Eclipse sollte unten ein Reiter Namens „Servers“ befinden, falls nicht ist es im Menüpunkt Window🡪Show view🡪Servers zu finden. Dort wird in dem leeren weißen Feld ein Rechtsklick ausgeführt und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



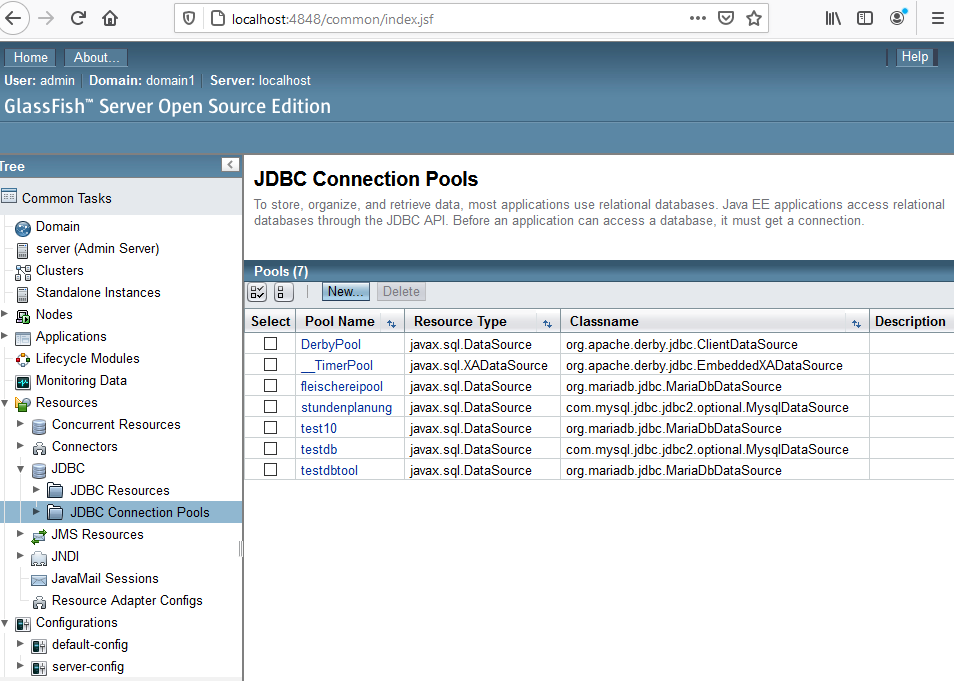
Wie im obigen Fenster zu sehen sind viele Server Typen aufgelistet.Aus der Lister wird GlassFish ausgewählt und auf Next klicken.



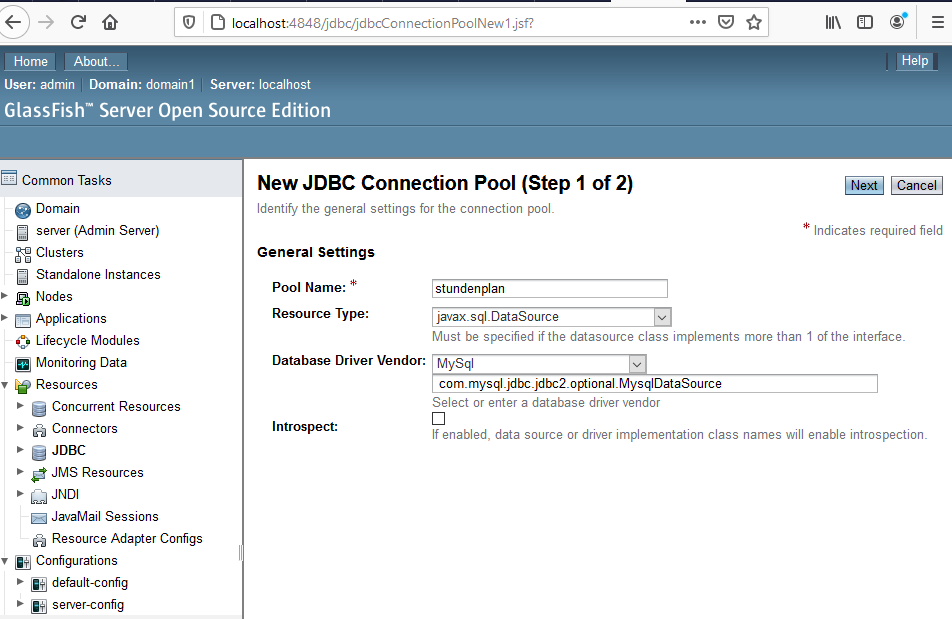
Nachdem klicken auf Next werden einige Eigenschaften zum GlassFisch Server angezeigt, wie der Name des Servers, Hostname, der Domain-Pfad und Verbindungszugangsdaten. Im Normalfall wird hier nichts geändert und mit Finish wird der Server bestätigt und erstellt.  
Bevor der Server gestartet wird um die 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.



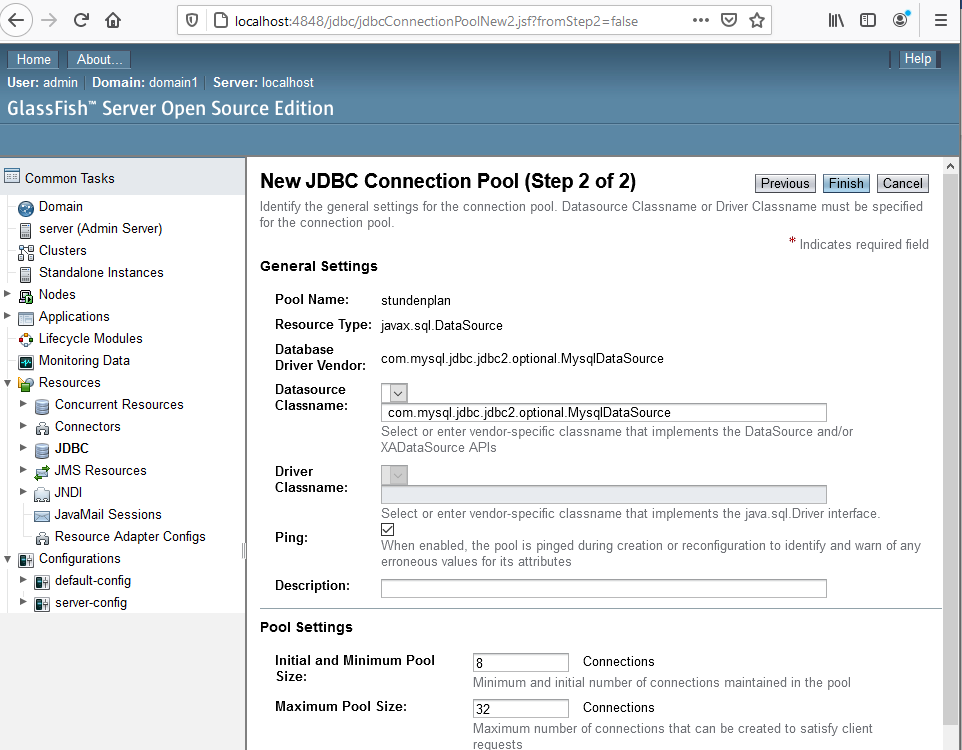
In dem Reiter Servers befindet sich der erstellte GlassFish Server. Dieser wird mit dem grünen Play Symbol oder mit einem Rechtsklick Start ausgewählt und gestartet. Nach einem erfolgreichen Start wird der Server wieder mit der rechten Maustaste angeklickt und Glassfish🡪View Admin Console ausgewählt.



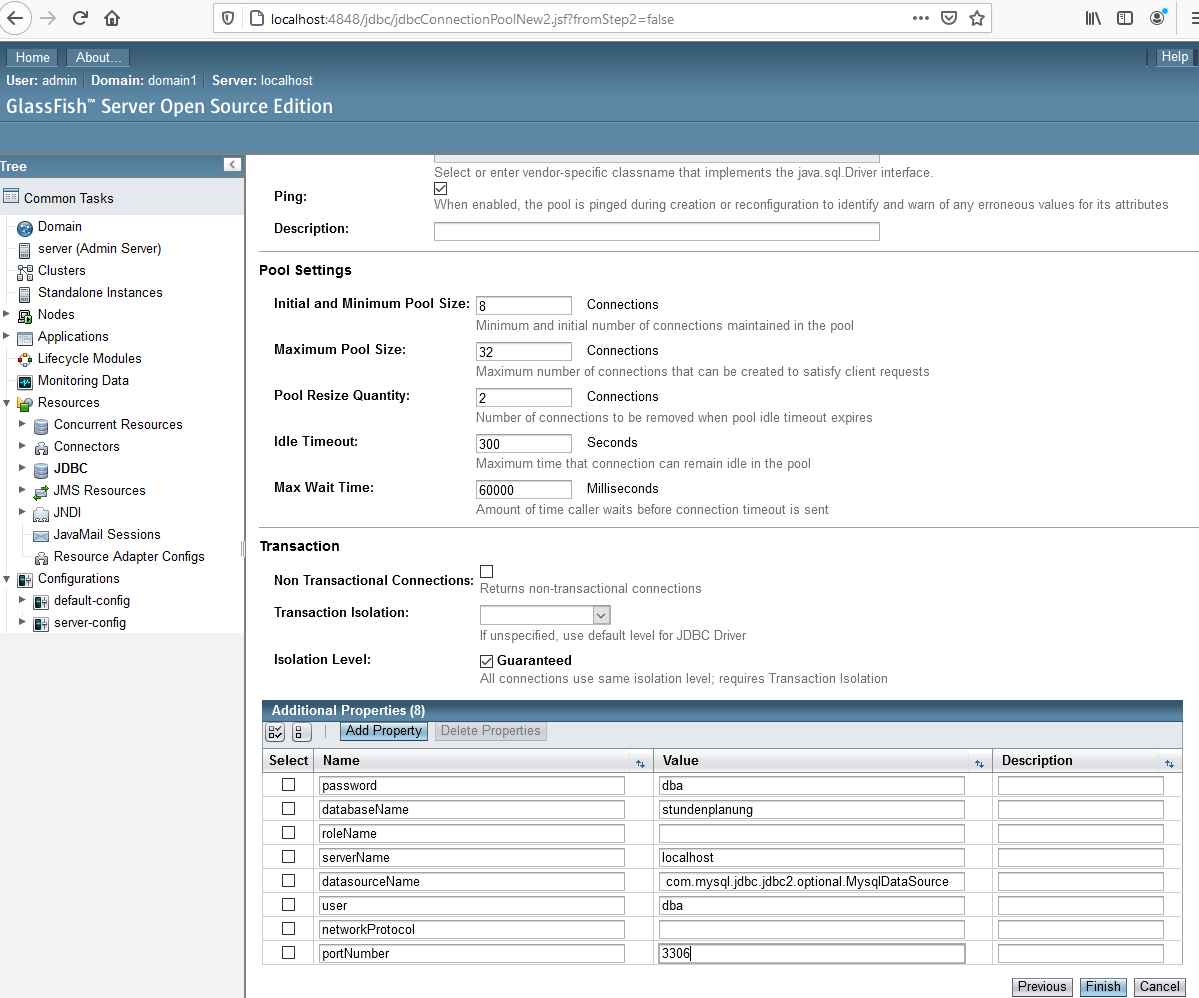
Es öffnet sich die Administrations-Console in dem Browser. In dem Reiter Resources🡪JDBC🡪JDBC Connection Pools wird mit New… ein neues Verbindungs-Pool eingerichtet.



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.

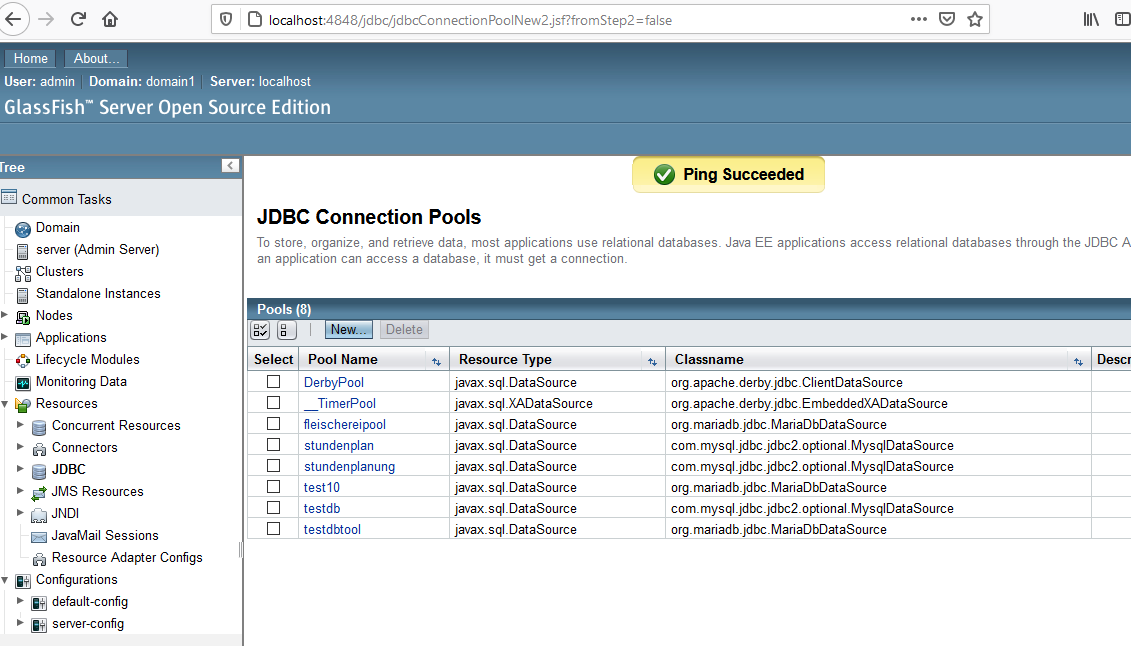


In dem zweiten Schritt wird bei Datasource Classname wieder die „com.mysql.jdbc.jdbc2.optional.MysqlDataSource“ Data Source eingetragen und die Seite nach unten zu Additional Properties gescrollt.

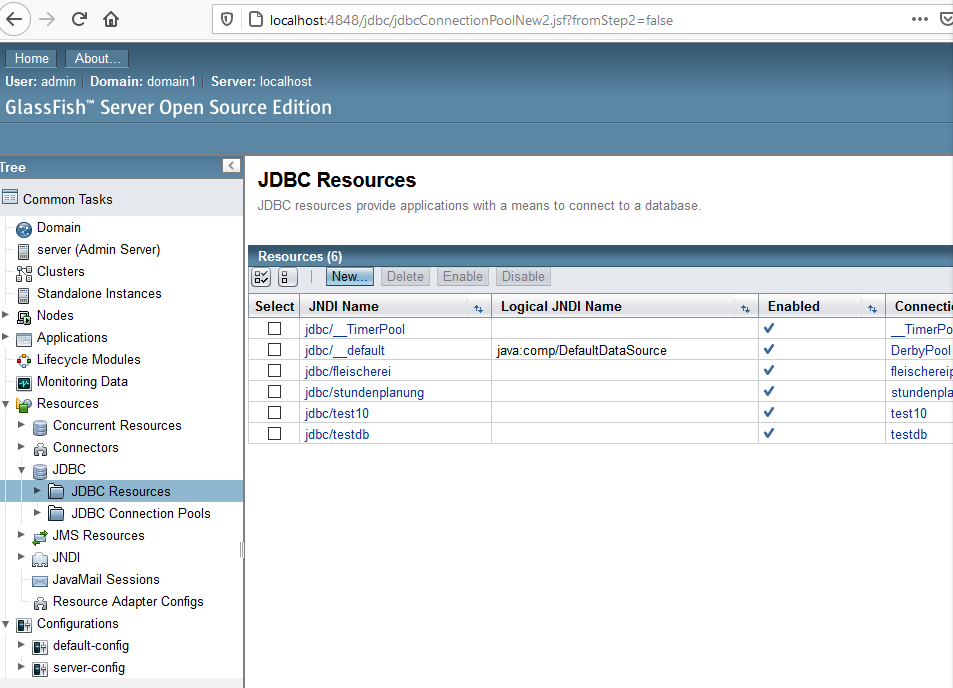


In den Additional Properties werden die Verbindungsdaten zur Datenbank eingegeben die im phpMyAdmin erstellt wurden.   
Password🡪dba  
databaseName🡪stundenplanung  
serverName🡪localhost  
datasourceName(optional)🡪 com.mysql.jdbc.jdbc2.optional.MysqlDataSource  
user🡪dba  
portNumber🡪3306

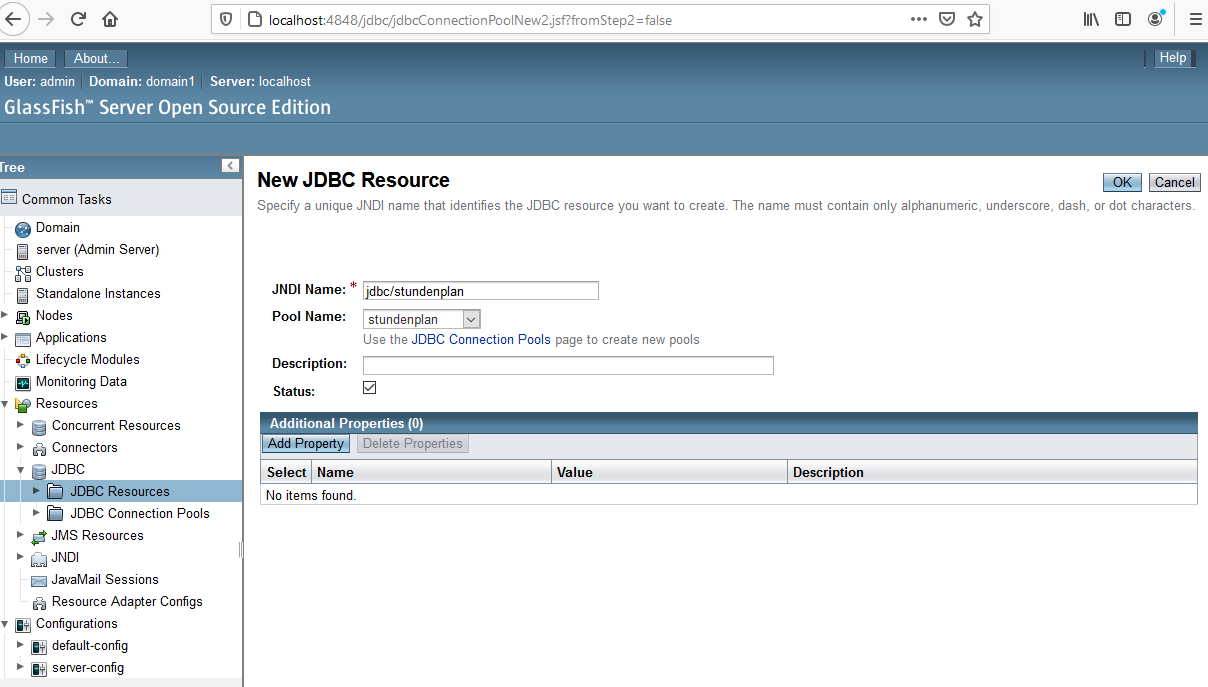
Mit Finish wird der JDBC Connection Pool eingerichtet und wenn man den Hacken bei Ping gesetzt hat wird direkt versucht eine Verbindung mit der Datenbank herzustellen.



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.



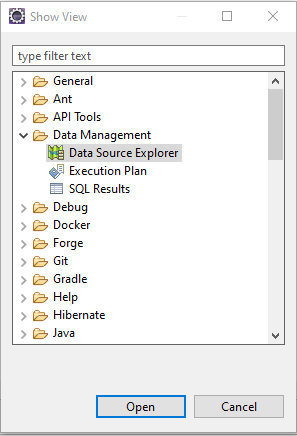
Wenn dies der Fall ist wird in dem Reiter Resources🡪JDBC🡪JDBC Ressources mit New.. eine JDBC Verbindung für den Connection Pool eingerichtet.



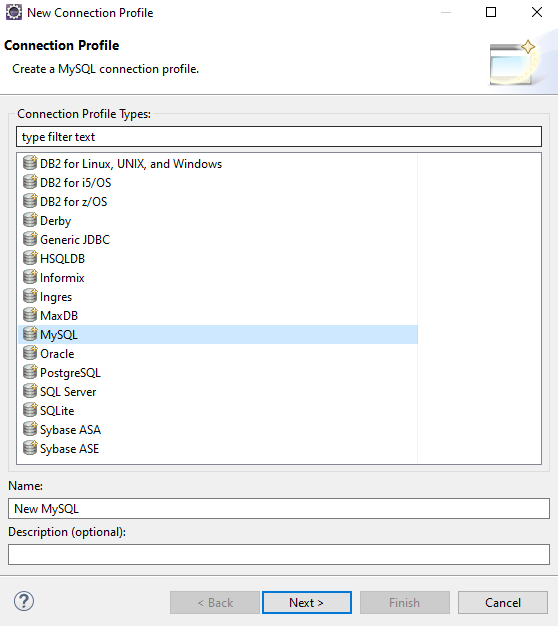
Im JNDI Name kann ein beliebiger Name eingegeben, aber standartgemäß wird jdbc/\*Datenbankname\* eingetragen, also in dem Fall jdbc/stundenplan. Diese JNDI wird nachher auch in der persistence.xml unter JTA-Data-Source eingetragen. Es ist daher wichtig dies zu merken oder zu notieren. Als Pool Name wird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

Verbindung zur Datenbank

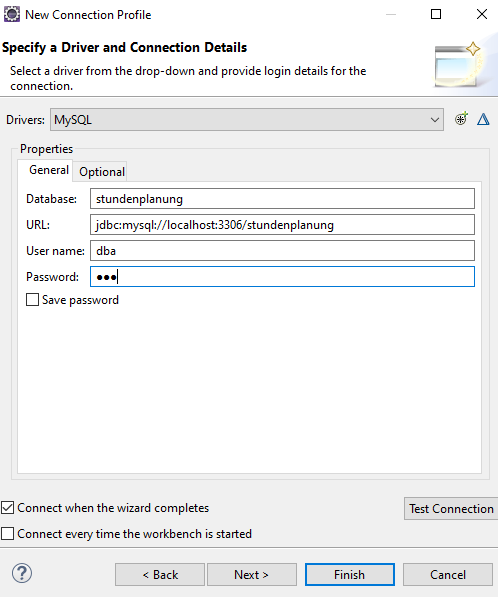
Um eine Verbindung zur Datenbank herzustellen wird ein MySQL Connector benötigt.  
Dieser wird über den Data Source Explorer eingestellt, falls dieser nicht in der Oberfläche zu finden ist, befindet sich diese unter dem Menüpunkt 🡪 Window 🡪 Show View 🡪Other… unter Data Management. Danach erscheint der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



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.

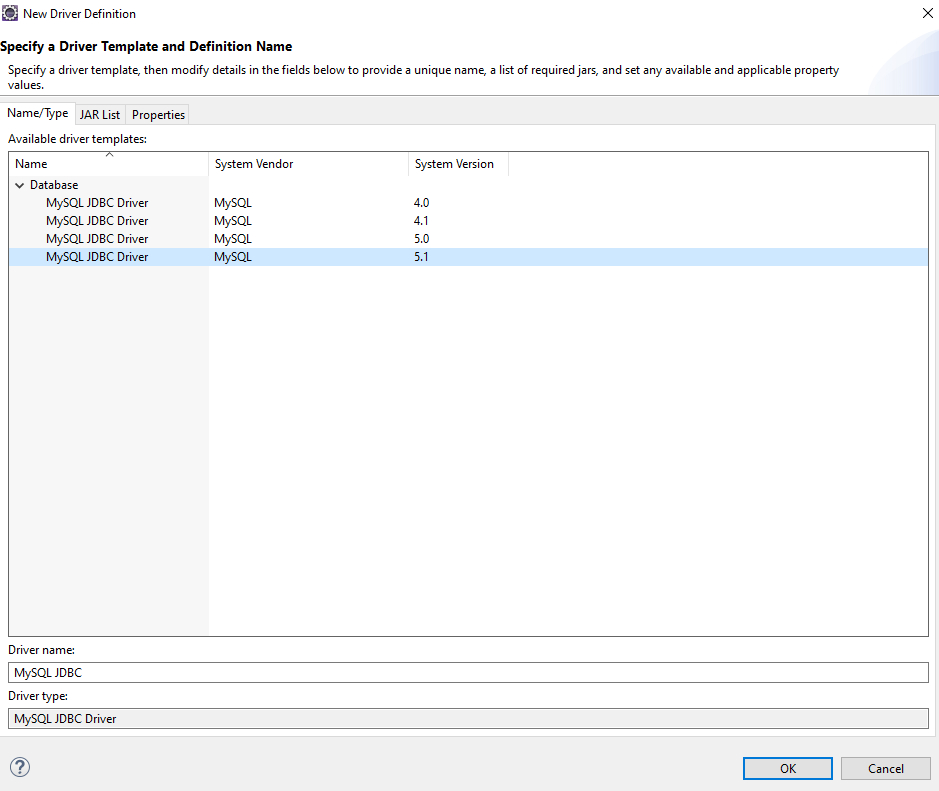


Da die Maria Datenbank auf MySQL basiert und Eclipse nicht mit dem MariaDB Connector arbeiten kann, wird der dazu kompatible MySQL Connector mit der Version 5.1.48 verwendet. Deswegen wird in der Liste MySQL ausgewählt, dazu einen passenden Namen für das Connection Profil eingetragen und anschließend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

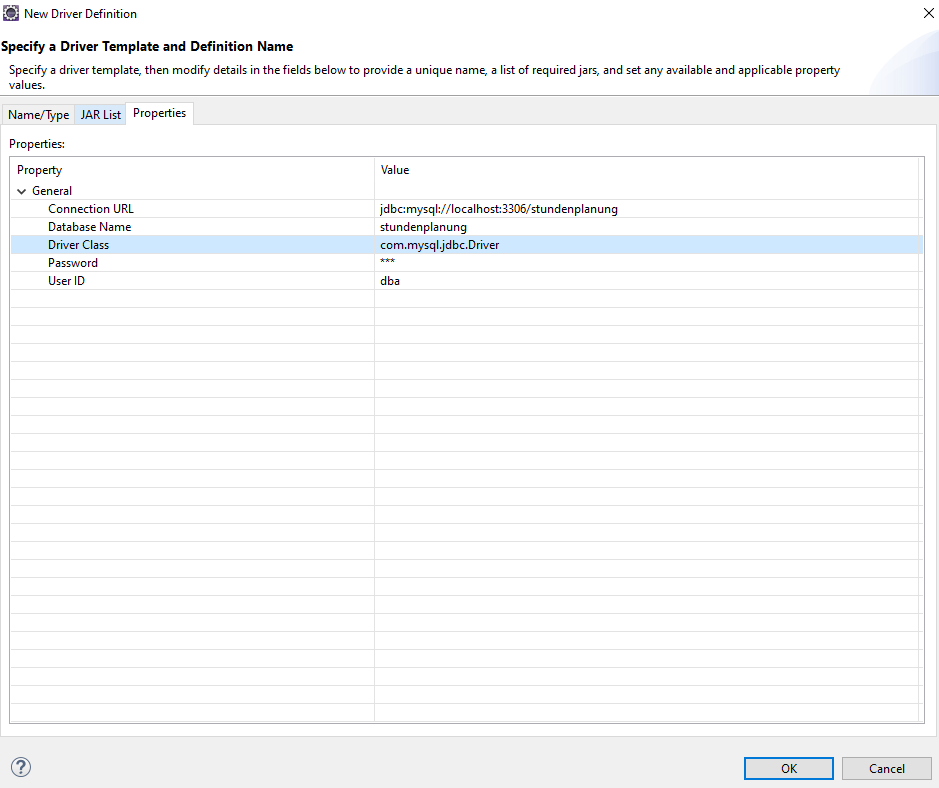


Dann erscheint das obrige Fenster, wo die Daten zur Datenbank gepflegt werden soll.  
In dem Feld neben Database sollte die erstellte Datenbank, bzw. die Datenbank eingetragen werden zu der die Verbindung hergestellt werden soll. Die URL ist standartgemäß als „jdbc:mysql://localhost:3306/database“ einzutragen, 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.



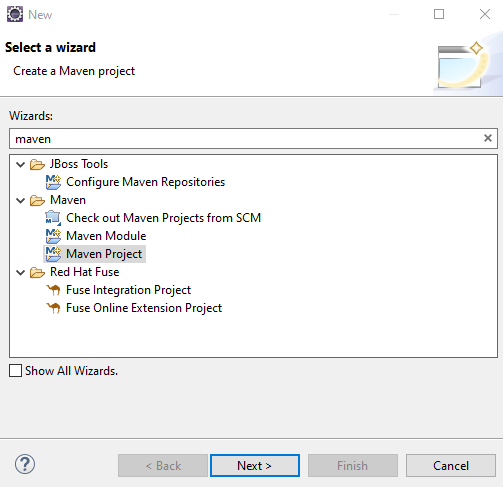
Nun wird eine Liste mit allen MySQL JDBC Driver angezeigt. Dort wird aus der Liste der kompitable Driver ausgewählt, welcher meistens mit der höchsten System version gekennzeichnet ist. Unten im Driver name muss ein eindeutiger Name vergeben werden, sonst entstehen Namenskonflikte. In dem Reiter JAR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



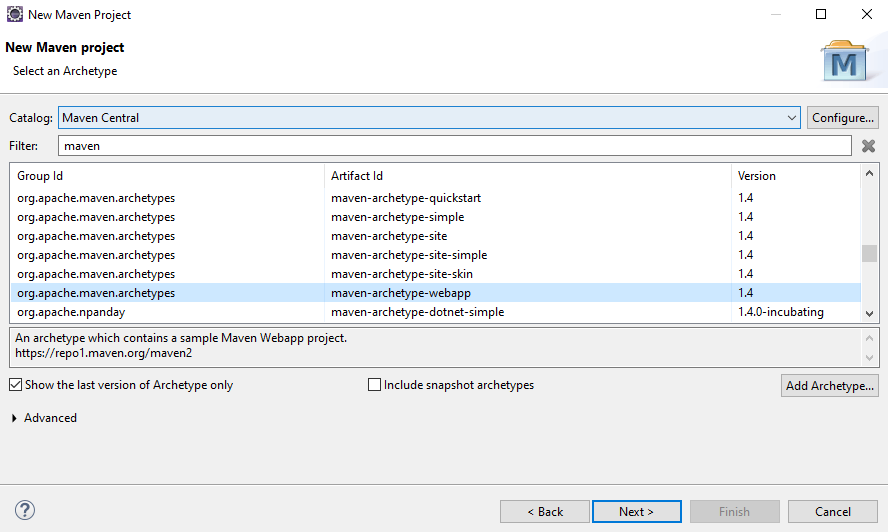
In dem letzten Reiter Properties werden die Daten für den MySQL Driver eingetragen und anschließend nochmal getestet, ob die Verbindung hergestellt werden kann.

Erstellung eines Maven-Projekts

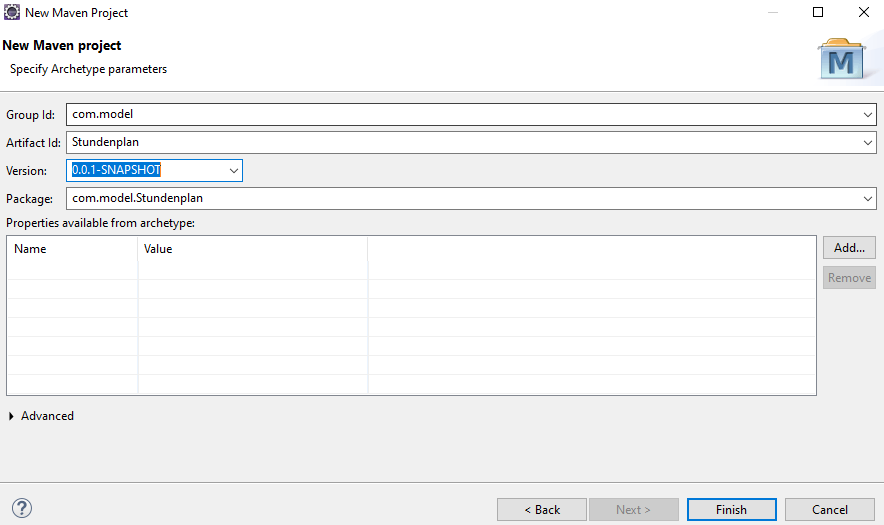
Für die Erstellung eines Maven-Projekts wählt man im Menüpunkt unter File🡪New🡪Other…  
und sucht nach dem Maven-Project.



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.



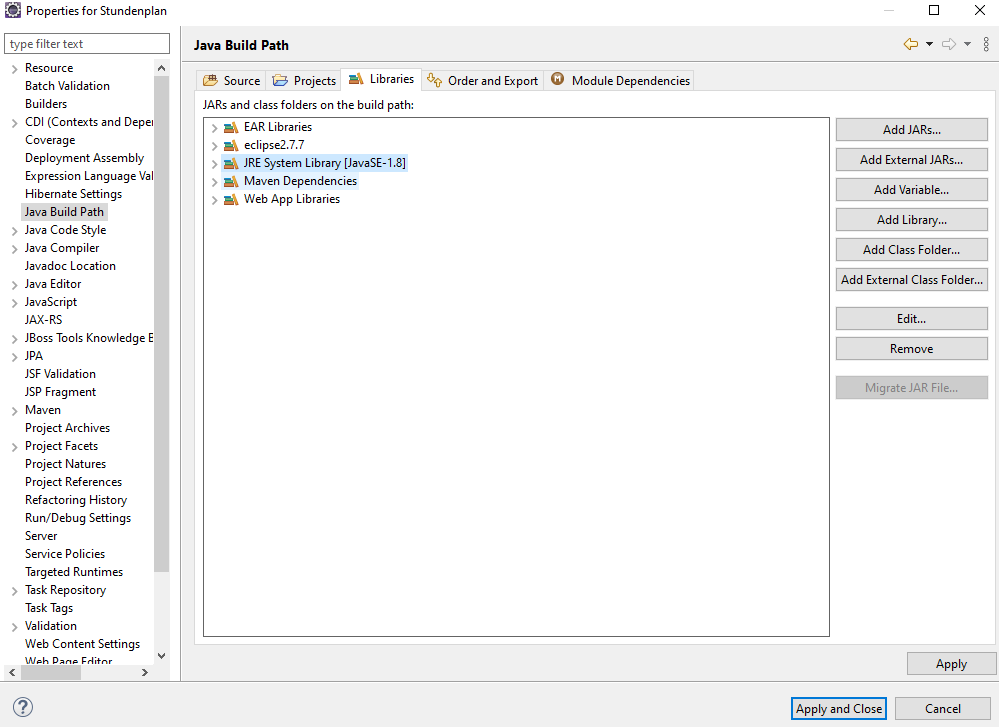
Wenn der Archetype in der Liste gefunden wurde, wird dieser selektiert und anschließend auf Next geklickt.



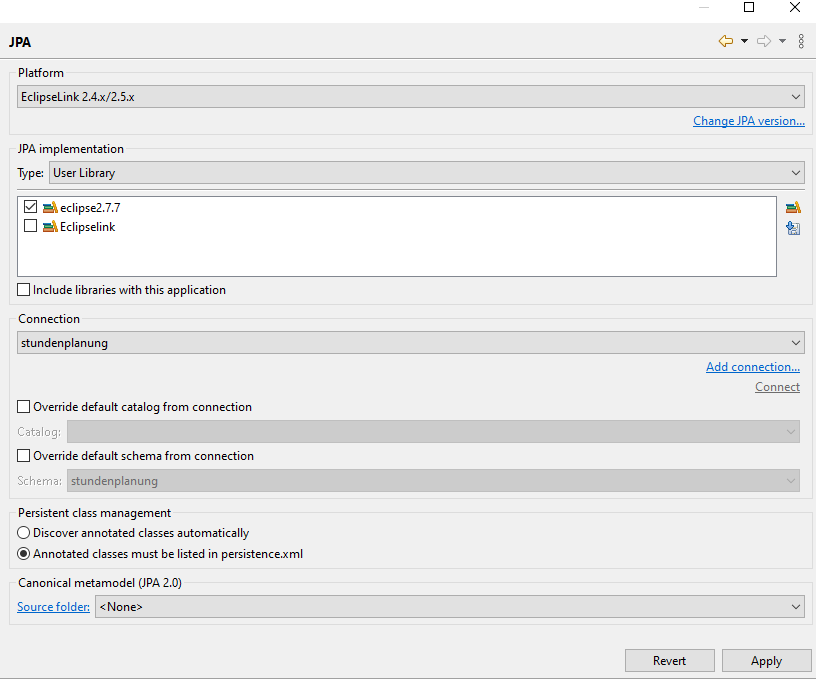
Für den Namen des Projektes wird die Artifact ID mit einem passenden Namen versehen und zuletzt mit Finish beendet.  
Nach der Erstellung des Projekts erscheint das Projekt im Project Explorer. Um ein lauffähiges Projekt zu bekommen, sind noch andere Schritte notwendig.

X.1 Einstellungen des Projekts

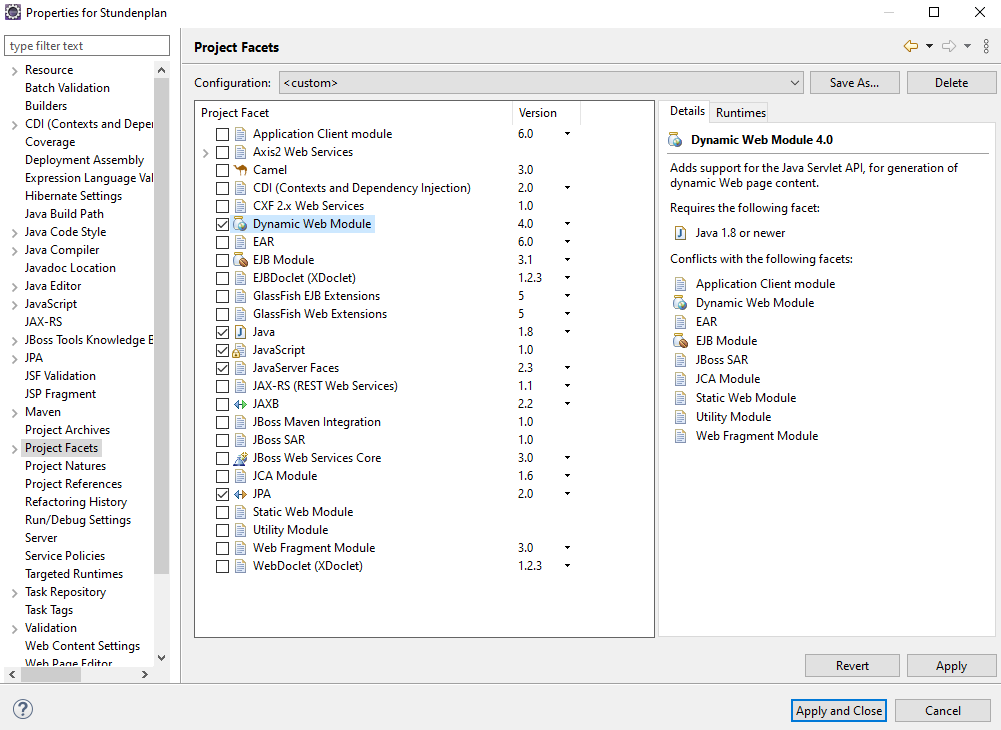
Als aller erstes werden die Eigenschaften des Projekts aufgerufen. Dies geschieht durch rechtsklick auf das Projekt🡪Properties.   
Danach wird der Menüpunkt Java Build Path aufgerufen und unter dem Reiter Libraries die JRE System Library editiert, sodass man die Java Version JDK 1.8 auswählen kann.



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.



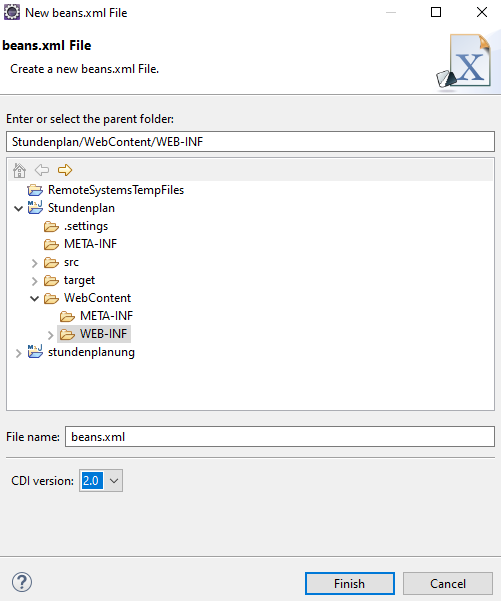
Für die Platform wird EclipseLink 2.4.x/2.5.x ausgewählt und eine User Library mit EclipseLink eingebunden. Unter Connection wird die Datenbank selektiert, zu der man die Verbindung hergestellt hat. Anschließend wird die Eingabe mit Apply bestätigt. Danach sollte das Fenster wieder die Project Facets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



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.

X.2 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.



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.

X.3 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

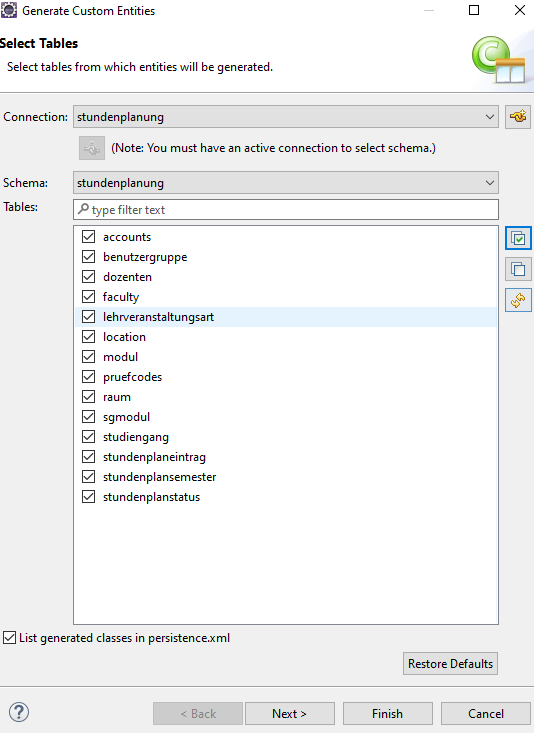
* all-themes-1.0.10.jar  
  Mehrere Weboberflächendesigns von Primefaces
* cdi-api-1.2.jar  
  Dienst, die zur Verbesserung der Struktur des Anwendungscodes beiträgt
* commons-math3-3.6.1.jar  
  Wird benötigt für Mathematik- und Statistikkomponenten.
* itext-2.1.7.jar  
  Freie PDF-Bibliothek wird verwendet 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

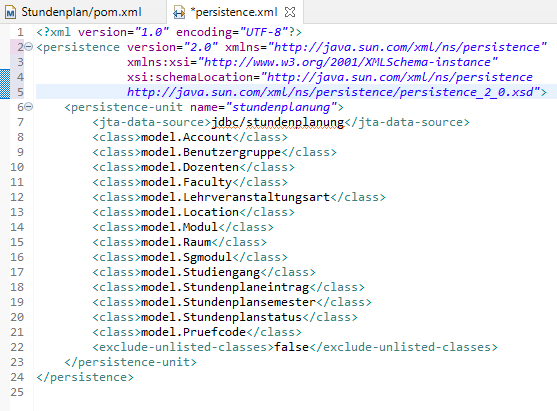
* jsf-api.2.2.20.jar  
  Wird benötigt für die Java Server Faces Applikation
* mysql-connector-java-5.1.48.jar  
  Connector für die Verbindung zur Datenbank
* javax.transaction-api-1.3.jar  
  Zuständig für die Transaktion
* javax.inject.jar  
  Wird benötigt zum Injizieren von anderen Klassen
* cdi-api-1.2.jar  
  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

X.4 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.



In diesem Schritt wird 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.



x.5 Zusätzliche Pakete in Eclipse

Neben den Paketen die für die Server Installationen gebraucht werden, 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.

Eclipse und Netbeans

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

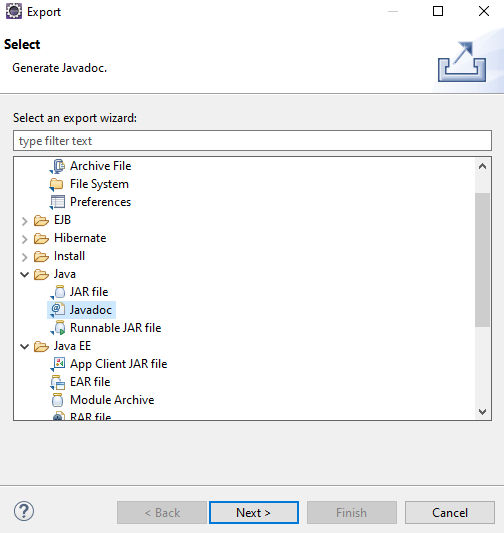
|  |  |
| --- | --- |
| Unterscheidungen von Netbeans und Eclipse | Vorteil bei: |
| Generierung der Entities | Netbeans |
| Server Installation | Netbeans |
| Projekt Erstellung | Netbeans |
| Arbeitsfreundlich | Netbeans |
| MariaDB Connector | Netbeans |
| Neukompilierung | Netbeans |
| Einfügen von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

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.

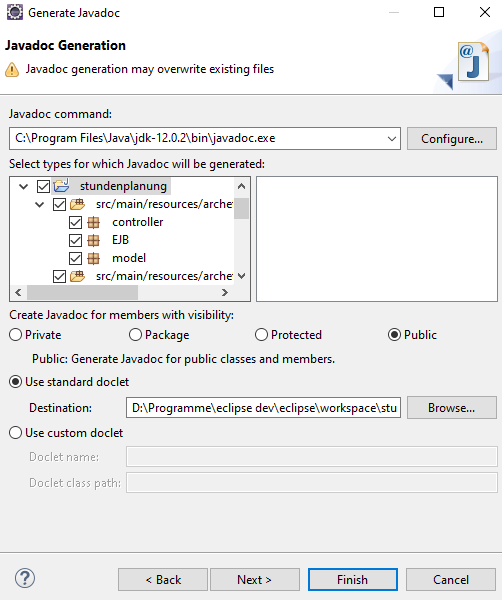
Javadoc und Internationalisierung

Javadoc

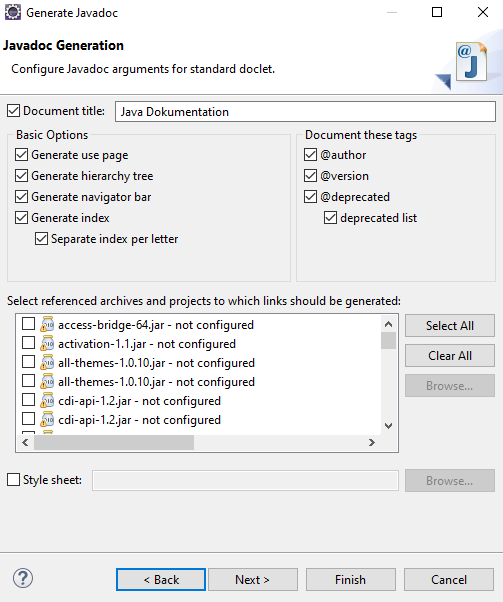
Für ein Javadoc ist es ratsam für jede Methode ein Javadoc Header zu erstellen. Dies geschieht durch einen Rechtsklick auf die Methode 🡪 Source 🡪 Generate Element Comment oder durch die Tastenkombination Alt+Shift+J, wodurch der Javadoc Header generiert wird. Aus der Methode werden die Parameter und 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.



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



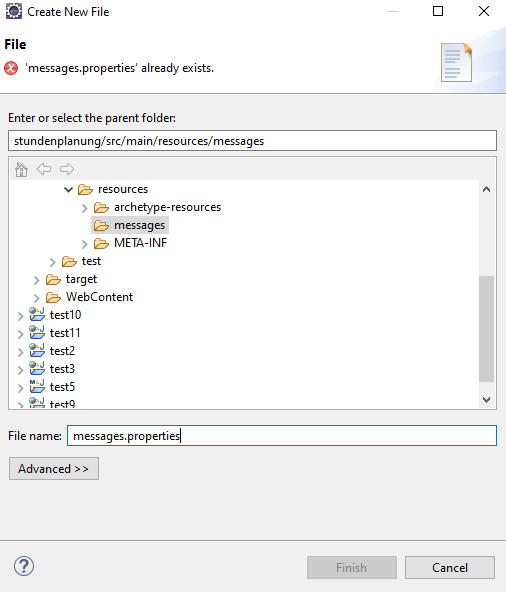
In diesem Fenster wird das jeweilige Projekt ausgewählt, dann die Sichtbarkeit des Javadocs und den Pfad, wo die Java Dokumentation gespeichert werden soll. Auf der nächsten Seite wird der Titel von dem Javadoc angegeben und zusätzlich die Basis Optionen und Tags ausgewählt.



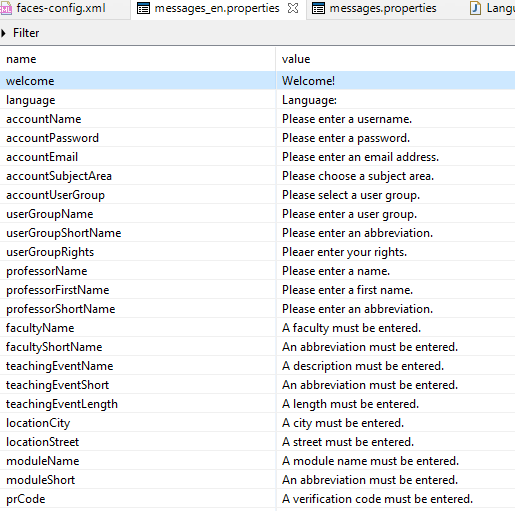
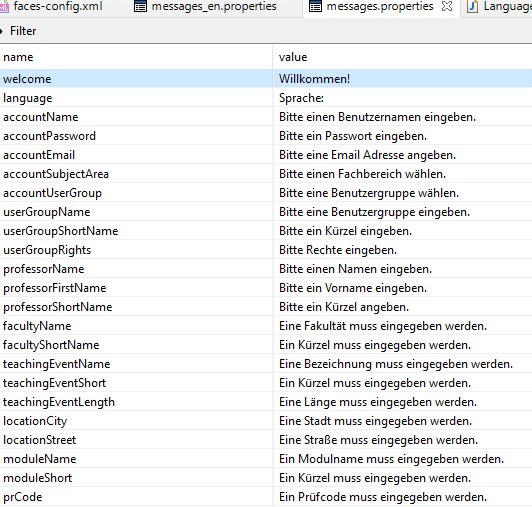
Zum Schluss wird die Eingabe mit Finish beendet, das Javadoc wird erstellt und dann in einem Browser geöffnet.

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.



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.



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>

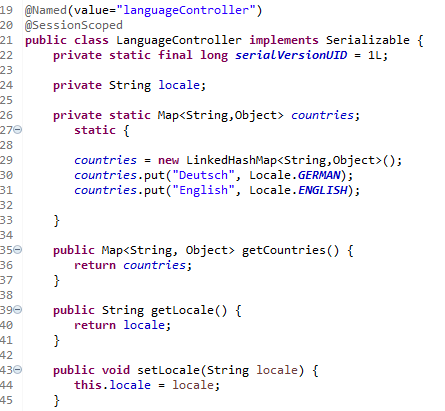
<var>msg</var>

</resource-bundle>

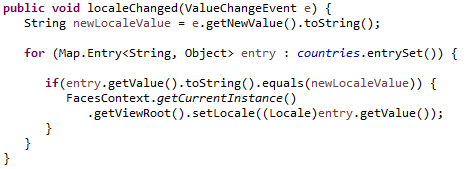
</application>

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.

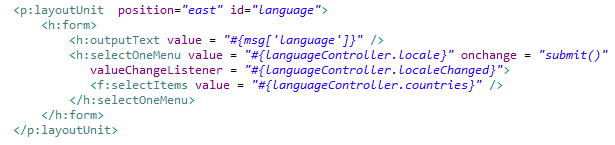
Der LanguageController besteht aus einer privaten String Variable mit dem Namen locale mit setter und getter Funktionen. Außerdem eine private statische HashMap mit den Attributen String und Objekt und den Namen countries. Diese Variable wird neu erzeugt und die Sprachen Deutsch und Englisch hinzugefügt,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



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.

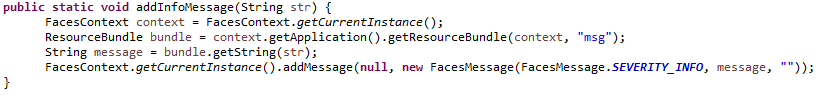


In der master.xhtml steht auf der Ost-Seite des Formulars die Internationalisierung.



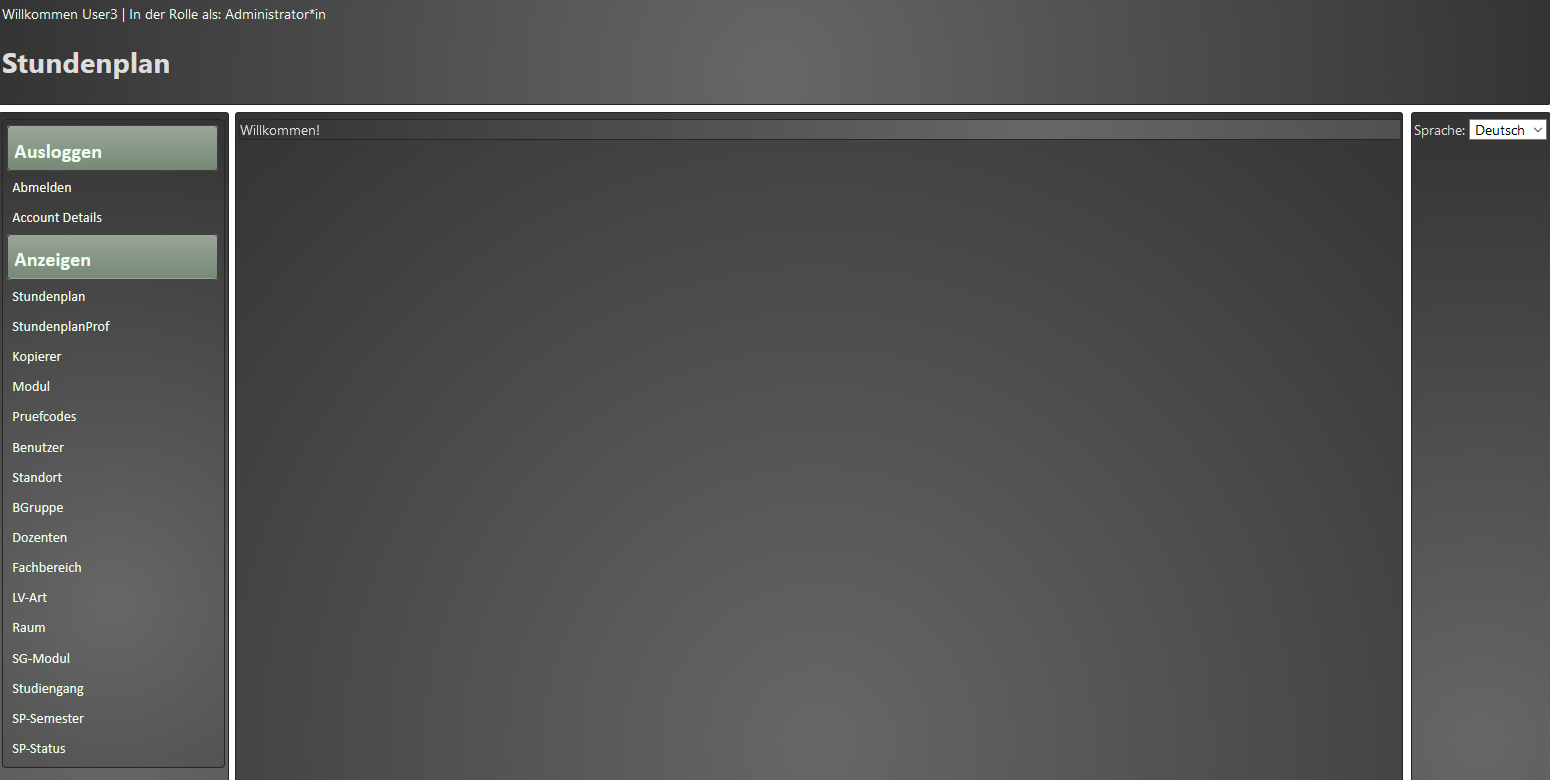
Die Variablen werden zum Beispiel mit <h:outputText value = “#{msg[‘language‘]}“/> aufgerufen. „msg“ steht für die Bundle-Variable und „language“ für Variable in der messages.properties bzw. messages\_en.properties. Es befindet sich zusätzlich noch ein Dropdownmenü mit dabei, um zwischen den Sprachen 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.

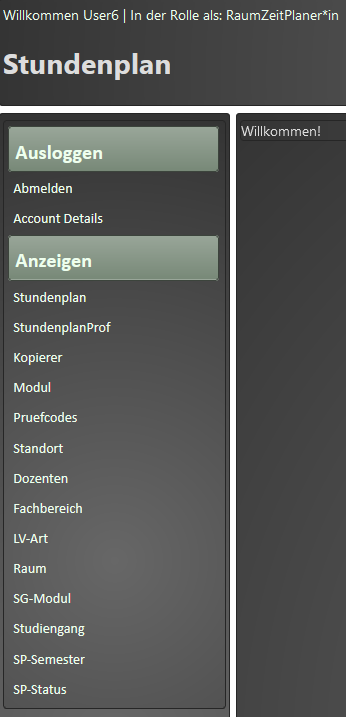


Master Template und Use Cases

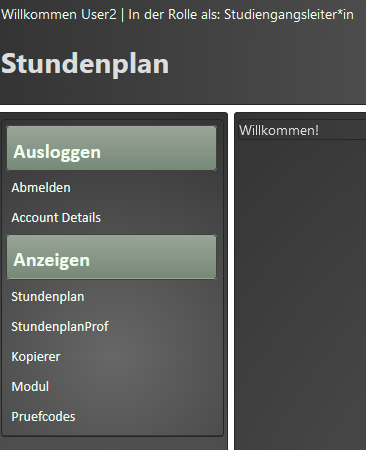
Das Mastertemplate ist eine Weboberflächenvorlage, welche für jede xhtml-Seite benutzt wird. Dieses besteht aus verschiedenen LayoutUnits, die sind in alle Himmelsrichtungen ausgelegt. In dem Nord-Bereich werden der Titel „Stundenplan“, der Name des Benutzers und die jeweilige Benutzergruppe, falls ein 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.



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



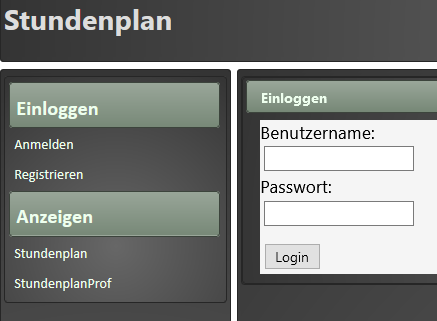
Der Raumzeitplaner darf wie der Administrator alles außer Benutzer hinzufügen/bearbeiten und Benutzergruppen erstellen.



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.



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



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.