

**Installation**

Semesterprojekt SS 2017

3D-Scanner

# Inhaltsverzeichnis

Inhaltsverzeichnis 2

1 Einleitung 3

1.1 Projektstruktur 3

1.2 Überblick der verwendeten Software 3

2 Backend 5

2.1 Datenbank-Server 5

2.1.1 Installation von MySQL 5

2.1.2 Aufsetzen der Datenbank 5

2.2 Webserver 7

2.2.1 Einrichten von WildFly 7

2.2.2 Einrichten der Spring Tool Suite 8

3 Frontend 11

3.1 Gulp – Automatisierungstool 11

3.2 Bower – Verwaltung und Aktuellhaltung von Frameworks 11

3.3 Dependencies installieren 11

3.3.1 NPM 11

3.3.2 Bower 11

4 Start-Programm 12

5 Debugging 13

5.1 Backend debuggen 13

5.2 Frontend debuggen 13

6 Release 15

6.1 Frontend 15

6.2 Backend 16

6.3 Start-Programm erstellen 16

6.4 MiniServer MySQL konfigurieren 17

6.5 Erstellen eines Installationsverzeichnisses 17

# Einleitung

## Projektstruktur

Dieses Dokument dient dazu, projektfremden Personen das Aufsetzen der Entwicklungsumgebung für das Notenarchiv-Programm Archivist zu ermöglichen.

Archivist ist in drei Softwareteilbereiche gegliedert:

* **Backend**: ein Spring Boot Webservice, der eine HTTP-Schnittstelle darstellt und Daten in einer MySQL-Datenbank verwaltet; dieser Webservice wird auf einem WildFly Application Server angeboten
* **Frontend**: eine Angular JS Web-Applikation, die das Notenarchiv grafisch aufbereitet und dargestellte Daten vom Backend bezieht; das Frontend wird ebenfalls im WildFly Application Server integriert
* **Start-Programm**: eine für den Endbenutzer vorgesehene Java-Applikation für Windows, welche bei Aufruf den WildFly Application Server startet, die Frontend-Oberfläche im Standard-Browser öffnet sowie ein Desktop-Verknüpfung für dieses Programm erstellt

Genaueres zur Systemarchitektur finden Sie im Dokument /Documentation/TechnicalDocumentation/**SystemArchitecture.docx**.

## Überblick der verwendeten Software

Zur Entwicklung der drei erwähnten Teilbereiche sind einige Software-Werkzeuge notwendig, um Source-Code zu verfassen, testen sowie eine Release-Version zu erstellen.

Systemvoraussetzung für dieses Projekt ist **Windows 10**.

Folgende Software wird zur Entwicklung von Archivist verwendet (wie und wo eine Software jeweils verwendet wird, wird in späteren Abschnitten erläutert):

|  |  |
| --- | --- |
| **Software und Version** | **Download-Link** |
| **Java JDK 1.8.0\_131 (32 bit)** | [http://www.oracle.com/technetwork/java/ javase/downloads/jdk8-downloads-2133151.html](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) |
| **Spring Tool Suite (STS) 3.8.4**  Bildergebnis für spring tool suite | <https://spring.io/tools/sts> |
| **Wildfly 10.1.0 Final**  Bildergebnis für wildfly | <http://wildfly.org/downloads/> |
| **MySQL Server 5.7.18 (64 bit)**  Bildergebnis für mysql | <https://dev.mysql.com/downloads/installer/> |
| **MySQL Workbench 6.3.9 CE (64 bit)**  Bildergebnis für mysql workbench logo | <https://dev.mysql.com/downloads/installer/> |
| **MiniServer MySQL 5.0.67** | [https://sourceforge.net/projects/miniserver/files/ MiniServer/MiniServer\_ %20MySQL%205.0.67%20Portable/](https://sourceforge.net/projects/miniserver/files/MiniServer/MiniServer_%20MySQL%205.0.67%20Portable/)  mini\_server\_11.zip |
| **Node.js 6.10.3 (64 bit)**  C:\Users\Paul\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png | <https://nodejs.org/en/download/> |
| **Brackets 1.9 (32 bit)**  C:\Users\Paul\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.jpg | <http://brackets.io/> |
| **Eclipse Neon 3 (32 bit)**  Home | <http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/neon3> |

# Backend

Für das Backend müssen einerseits ein MySQL-Server und andererseits ein Spring Boot Webservice aufgesetzt werden.

## Datenbank-Server

### Installation von MySQL

Bei der Installation vom MySQL Server werden der Einfachheit halber folgende Zugangsdaten definiert:

User: **root**

Password: **admin**

### Aufsetzen der Datenbank

Für das Einrichten des Datenbank-Servers bestehen zwei Wege:

#### A) Aufsetzen mit Command Line

Voraussetzungen:

* MySQL Server
* SQL-Skripts:
  + /Implementation/SourceCode/init\_db.sql
  + /Implementation/SourceCode/TestData/archivist\_genre.sql
  + /Implementation/SourceCode/TestData/archivist\_instrument.sql
  + /Implementation/SourceCode/TestData/archivist\_musicpiece.sql
  + /Implementation/SourceCode/TestData/archivist\_score.sql

Bei dieser Variante werden die SQL-Skripts zur Erstellung der Datenbank sowie der Testdaten per Command Line eingespielt.

1. Windows Command Line Prompt im bin-Verzeichnis von MySQL öffnen (zB D:\Program Files\MySQL\MySQL Server 5.7\bin)
2. mysql --user=admin --password=admin
3. Erstellen der Datenbank und Tabellen:  
   mysql>source init\_db.sql
4. Einfügen von Testdaten und Neuerzeugung von Tabellen (Skripts können jederzeit erneut ausgeführt werden, um alle vorhandenen Testdaten zu löschen und die in den Skripts definierten Testdaten wieder in die leere Tabellen einzufügen):  
   mysql>source archivist\_genre.sql  
   mysql>source archivist\_instrument.sql  
   mysql>source archivist\_musicpiece.sql  
   mysql>source archivist\_score.sql

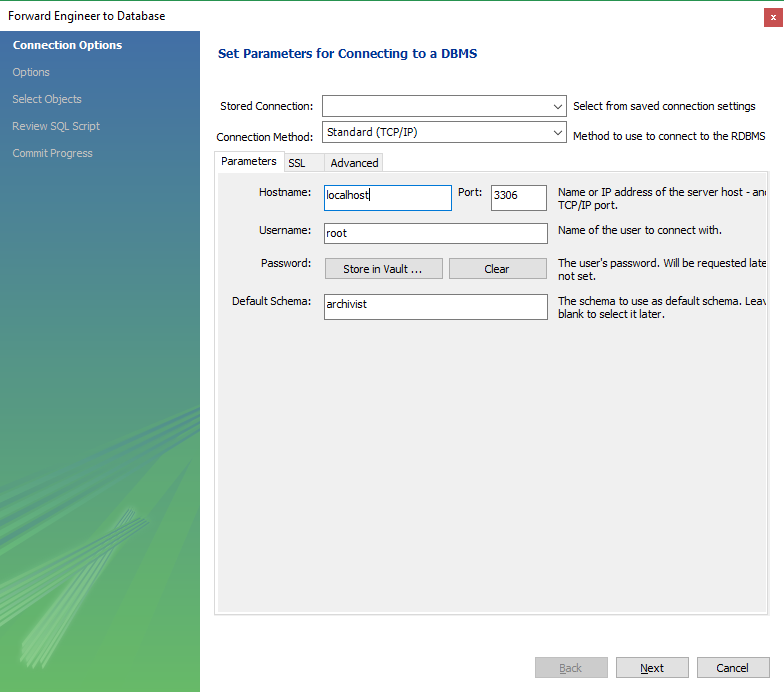
#### B) Aufsetzen mit MySQL Workbench

Voraussetzungen:

* MySQL Server
* MySQL Workbench
* /Implementation/ProjectFiles/archivist\_erd.mwb
* SQL-Skripts:
  + /Implementation/SourceCode/TestData/archivist\_genre.sql
  + /Implementation/SourceCode/TestData/archivist\_instrument.sql
  + /Implementation/SourceCode/TestData/archivist\_musicpiece.sql
  + /Implementation/SourceCode/TestData/archivist\_score.sql

Mithilfe der MySQL Workbench wird mittels Forward Engineer die Datenbank erstellt und die Testdaten mit dem Data Import Tool eingefügt.

1. Öffnen des Datenbankmodells **archivist\_erd.mwb** in der MySQL Workbench
2. Datenbank erzeugen: Database>Forward Engineer



1. Lokale Instanz öffnen und Daten importieren: Server>Data Import>Auswählen des Ordners TestData (SQL-Skripts für Testdaten)

## Webserver

Voraussetzungen:

* Installation von Java
* WildFly
* Spring Tool Suite

### Einrichten von WildFly

Voraussetzungen:

* Download von MySQL Connector für Java 5.1.42 <https://dev.mysql.com/downloads/connector/j/>
* Download von Dom4j 1.6.1 <https://mvnrepository.com/artifact/dom4j/dom4j/1.6.1>

#### Umgebungsvariable JAVA\_HOME

Falls noch nicht vorhanden, wird eine neue Windows-Umgebungsvariable angelegt:

JAVA\_HOME=C:\Program Files (x86)\Java\jdk1.8.0\_101

#### MySQL Treiber

Mittels MySQL Connector für Java wird der MySQL-Treiber für WildFly eingerichtet.

1. Im WildFly-Ordner ist folgende Ordner-Struktur zu erstellen:  
   <wildlfy\_home>/modules/system/layers/base/com/mysql/main
2. Die Datei **mysql-connector-java-5.1.42-bin.jar** aus dem MySQL Connector Packet wird in diesem Ordner eingefügt.
3. Eine Datei **module.xml** mit folgendem Inhalt wird angelegt:

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

<module xmlns=**"urn:jboss:module:1.0"** name=**"com.mysql"**>

<resources>

<resource-root path=**"mysql-connector-java-5.1.42-bin.jar"**/>

</resources>

<dependencies>

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

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

</dependencies>

</module>

1. Weiters ist in der Datei standalone.xml folgender Inhalt sicherzustellen  
   (<wildlfy\_home>/standalone/configuration/standalone.xml):

...

<datasources>

...

<datasource jta=**"true"** jndi-name=**"java:/archivist"** pool-name=**"archivist"** enabled=**"true"** use-ccm=**"true"**>

<connection-url>**jdbc:mysql://localhost:3306/archivist**</connection-url>

<driver-class>**com.mysql.jdbc.Driver**</driver-class>

<driver>**com.mysql**</driver>

<security>

<user-name>**root**</user-name>

<password>**admin**</password>

</security>

<validation>

<valid-connection-checker class-name=**"org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"**/>

<background-validation>**true**</background-validation>

<exception-sorter class-name=**"org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"**/>

</validation>

</datasource>

<drivers>

<driver name=**"com.mysql"** module=**"com.mysql"**/>

...

</drivers>

</datasources>

...

#### Dom4j

Durch Fehlermeldungen während der Entwicklung wurde die Library Dom4j aktualisiert. Auf WildFly ist standarmäßig Dom4j 1.6.1 installiert. Dieselbe Version von Dom4j wird unter obenstehendem Link (siehe 2.2.1 Einrichten von WildFly) heruntergeladen.

Die Datei **dom4j-1.6.1.jar** wird im Verzeichnis <wildfly\_home> /modules/system/layers/base/org/dom4j/main gespeichert.

#### Benutzer anlegen

Um die Konfiguration von WildFly abzuschließen, wird ein WildFly-User angelegt. Dazu wird das Skript **add-user.bat** ausgeführt (Verzeichnis <wildfly\_home>/bin).

Beispielsweise werden folgende Daten festgelegt:

User: **admin**

Password: **WildFly123!**

### Einrichten der Spring Tool Suite

#### Market Place Extensions

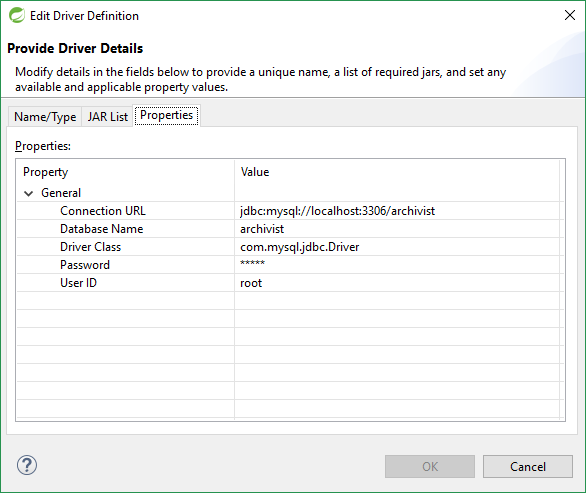
Zu Beginn werden nach dem Start von STS über Help>Eclipse Marketplace… zwei Pakete installiert:

* **Eclipse Data Tools** (für die Data Tools Platform Enablement for MySQL)
* **JBoss Tools** (zum Hinzufügen der Server Ansicht für den WildFly-Server)

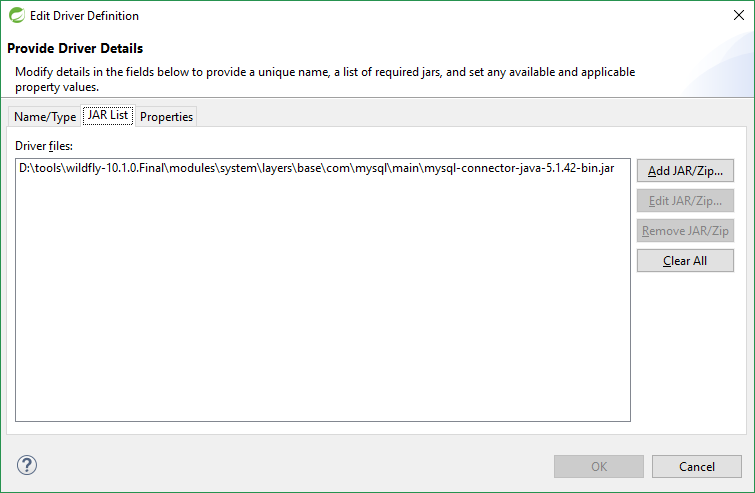
#### Data Source Explorer

Optional kann im Data Source Explorer die Verbindung zur Datenbank „archivist“ hergestellt werden, um beispielsweise Testdaten einzupflegen.

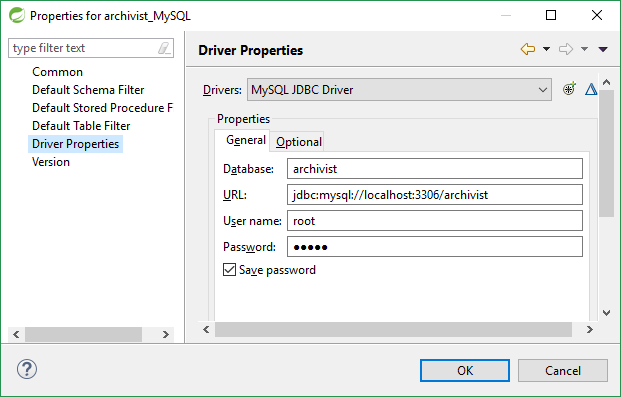
Für den MySQL Driver werden folgende Informationen eingegeben:



Password: **admin** (siehe 2.1.1 Installation von MySQL)



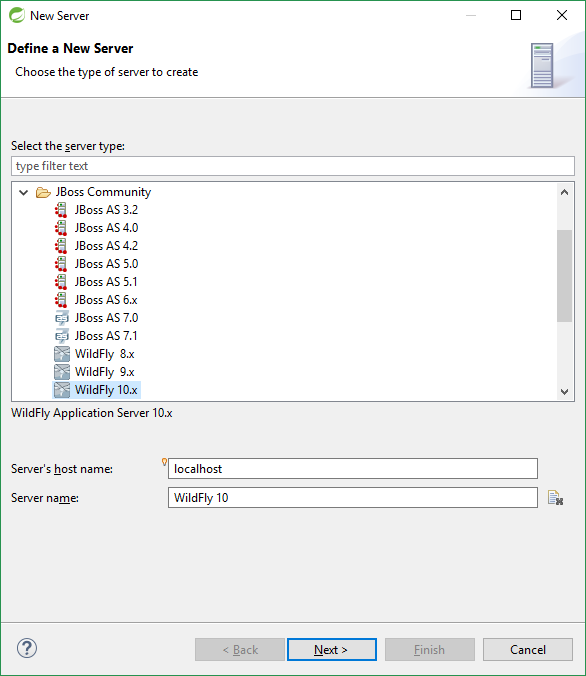
Demnach wird die Verbindung zur Datenbank festgelegt:



Password: **admin** (siehe 2.1.1 Installation von MySQL)

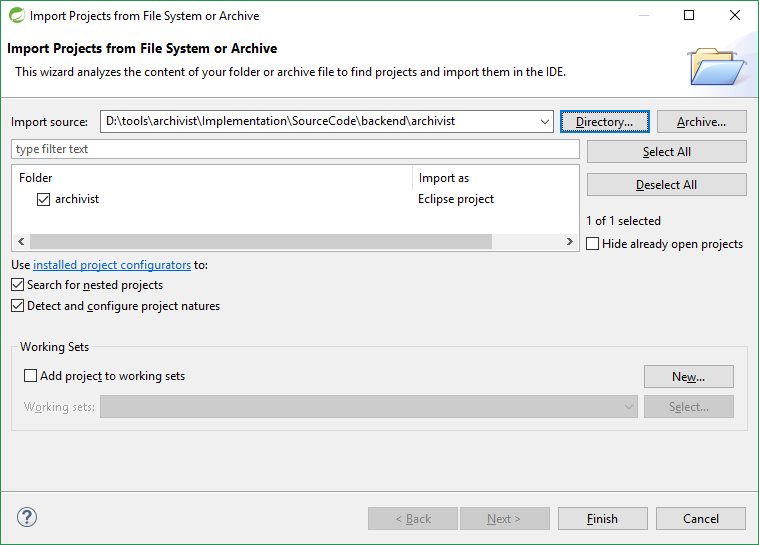
#### WildFly im Server Explorer

Im Server Explorer wird ein neuer lokaler WildFly Server mit dem entsprechenden Home-Verzeichnis angelegt:



#### Projekt Archivist importieren

Zuletzt wird über File>Open Projects From File System… das Projekt importiert.



Zu Beginn können die Schritte

* Project>Clean…
* Maven>Update Project…

durchgeführt werden, damit alle Dependencies geladen werden.

# Frontend

Voraussetzungen:

* Node.js (damit erhält man NPM, den Node Package Manager, welcher für das Frontend-Projekt benötigt wird)
* Brackets (jeder andere **beliebige Editor für HTML-Projekte** möglich!)

## Gulp – Automatisierungstool

Globale Installation über Command Line im Verzeichnis /Implementation/SourceCode/frontend

npm install -g gulp

## Bower – Verwaltung und Aktuellhaltung von Frameworks

Globale Installation über Command Line im Verzeichnis /Implementation/SourceCode/frontend

npm install -g bower

## Dependencies installieren

Für NPM und Bower werden nun die festgelegten Dependencies installiert.

### NPM

Die notwendigen Abhängigkeiten sind in der Datei **package.json** festgelegt. Die Installation dieser wird über Command Line (im Verzeichnis **frontend**) durchgeführt:

npm install

Die Dependencies befinden sich nun im Ordner **node\_modules**.

### Bower

Die Abhängigkeiten sind in der Datei **bower.json** aufgelistet. Die Installation dieser wird über Command Line (im Verzeichnis **frontend**) durchgeführt:

bower install

Die Dependencies werden im Ordner **bower\_components** abgelegt.

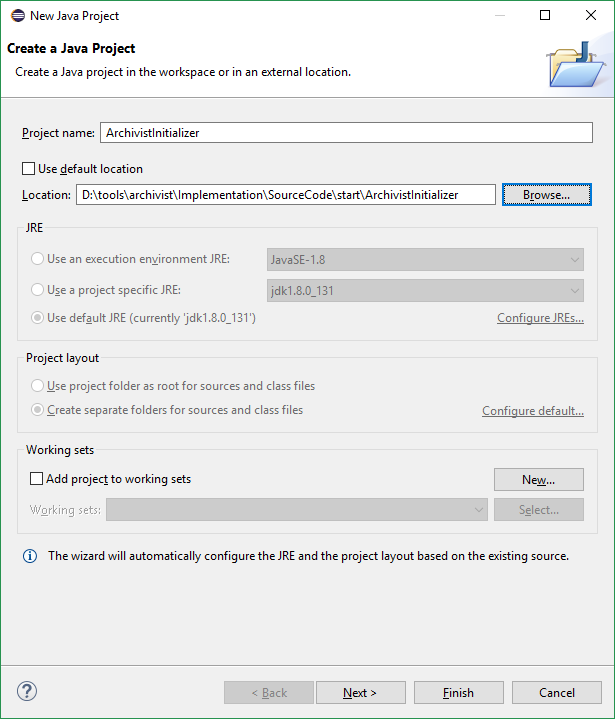
# Start-Programm

Voraussetzungen:

* Eclipse Neon
* /Implementation/SourceCode/start/ArchivistInitializer

Nach der Installation von Eclipse Neon kann das Projekt **ArchivistInitializer** geöffnet werden.

File>New>Java Project…



**Anmerkung**: Sollten während der Entwicklung Assets (zB Bild-Dateien oder Batch-Skripts) im Source-Folder aktualisiert bzw. neue hinzugefügt werden, ist das Ausführen von Project>Clean… für dieses Projekt hilfreich, damit beim Debuggen keine Assets von vorigen Builds verwendet werden.

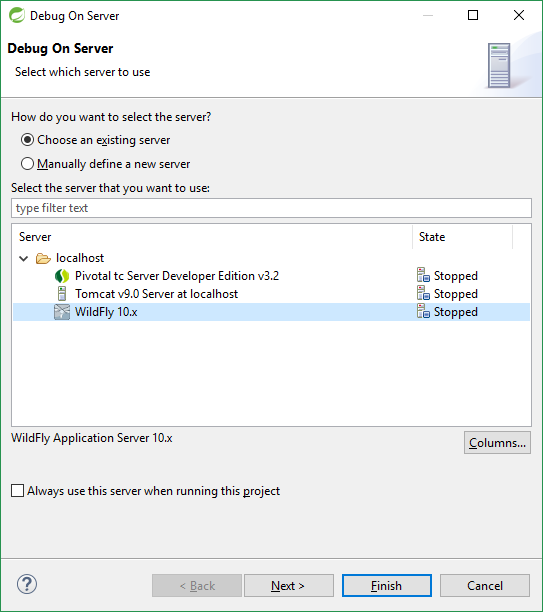
# Debugging

## Backend debuggen

Bei allen API-Controllern ist es wichtig, die CrossOrigin-Annotation vor der Klassendefinition sicherzustellen.



Mit STS wird das Backend-Programm unter Auswahl des eingerichteten WildFly-Servers ausgeführt.

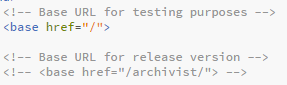


Die API Schnittstelle ist unter <http://localhost:8080/archivist/api> erreichbar.

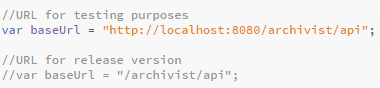
## Frontend debuggen

Folgender Code muss vorhanden sein, um das Projekt zu debuggen.

index.html



main.svc.js



Anschließend kann das Projekt über Command Line im Verzeichnis **frontend** gestartet werden:

gulp run

Das Gulp-Skript wird somit ausgeführt und ein lokaler Webservice auf Port 8000 geöffnet, wo die Applikation aufgerufen werden kann. Bei Code-Änderungen werden dieses sofort übernommen.

Im Browser kann die Applikation unter

http://localhost:8000

aufgerufen werden.

# Release

Um eine auslieferbare Version zu erstellen, müssen folgende Schritte abgewickelt werden.

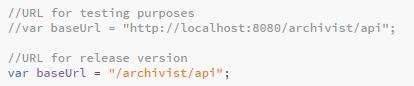
## Frontend

Im Frontend-Projekt sind in index.html und main.svc.js diese Code-Stellen sicherzustellen:

index.html



main.svc.js

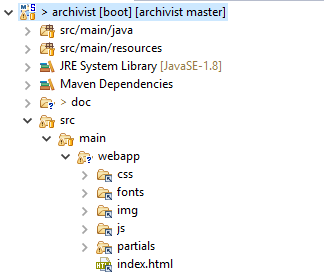


Daraufhin wird im frontend-Verzeichnis von der Command Line der Befehl

gulp build

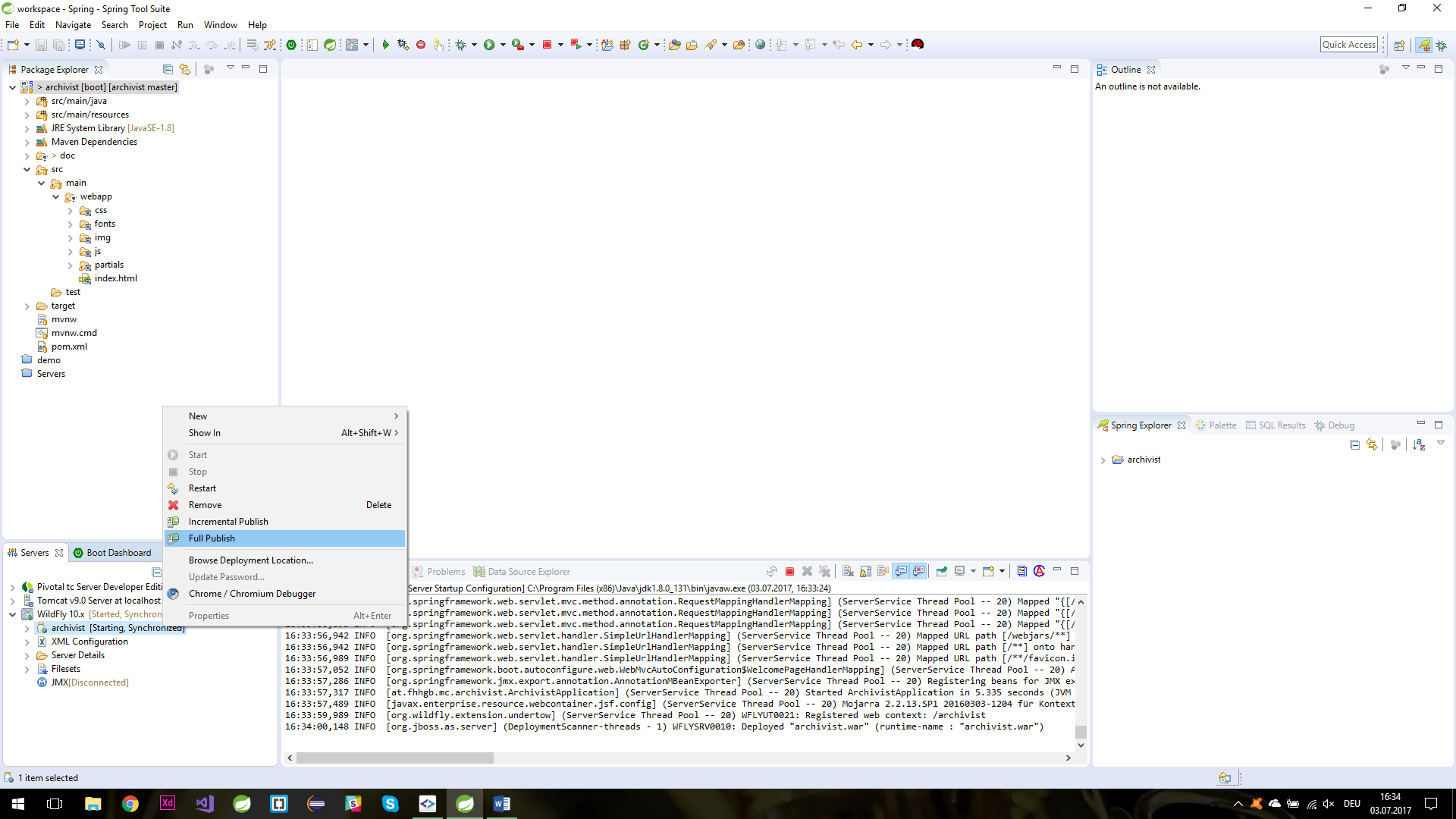
ausgeführt.

Der Inhalt des Ordners **frontend/dist** wird im Backend-Projekt als Verknüpfung eingefügt, sodass folgende Struktur vorliegt:



## Backend

Für das Backend-Projekt wird lediglich ein Republish auf das Archivist-Projekt durchgeführt.



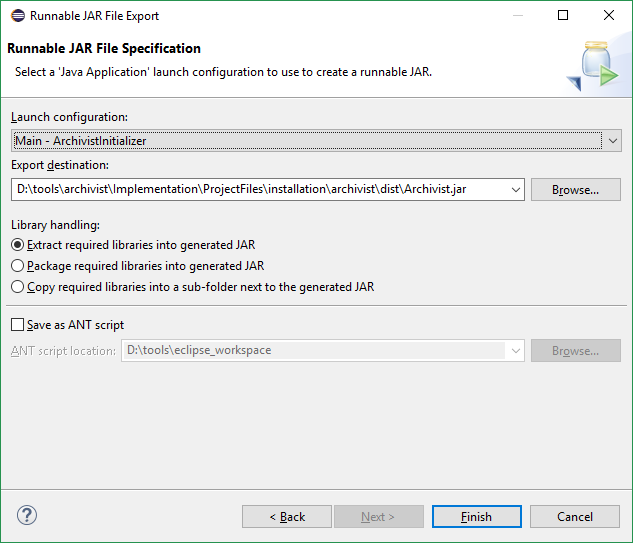
Somit sind auch alle Frontend-Dateien auf dem Server vorhanden.

## Start-Programm erstellen

Im nächsten Schritt wird eine JAR-Datei erzeugt, welche der Endbenutzer verwendet, um den Server zu starten sowie die Weboberfläche per Mausklick zu öffnen.

Dazu wird in Eclipse Neon eine Runnable JAR erzeugt.

Export>Runnable JAR file



## MiniServer MySQL konfigurieren

Vorerst muss das Skript **mysql\_start.bat** des MiniServers gestartet werden (davor muss der MySQL-Dienst vom tatsächlichen MySQL Server gestoppt werden).

Im Verzeichnis <mini\_server>/udrive/bin befinden sich die Binaries von MySQL.

In der Command Line wird das Standardpasswort „root“ auf „admin“ angepasst:

mysqladmin --user=root --password=root password admin

Anschließend werden dieselben Schritte wie in Punkt 2.1.2.1 A) Aufsetzen mit Command Line durchgeführt.

Die Skripts für das Aufsetzen der Datenbank sowie der Testdaten können mit der MySQL Workbench noch jederzeit angepasst werden, falls die Datenbank zuvor geändert wurde:

* TestData mittels Server>Data Export
* init\_db.sql wird aus dem Forward Engineering Fenster kopiert

## Erstellen eines Installationsverzeichnisses

Zum Schluss werden alle Komponenten in einem Ordner zusammengeführt, sodass folgende Struktur entsteht:

/installation

/wildfly-10.1.0.Final/… (gesamtes WildFly-Verzeichnis)

/mini\_server\_11/… (gesamtes MiniServer-Verzeichnis)

/Archivist.jar

Der Ordner **installation** kann nun auf einen Zielrechner kopiert werden und mittels Ausführung von Archivist.jar wird das Programm gestartet.