

Anleitung zum Starten der Beispielprogramme des Buchs "Der Java Profi: Persistenzlösungen und REST-Services"

Michael Inden, 14.6.2016 — Bei Fragen erreichen Sie mich unter: michael_inden@hotmail.com

Voraussetzungen

Sie benötigen ein installiertes JDK 8 sowie das Build-Tool Gradle ab Version 2, am besten die Version 2.13 oder neuer. Eine Einführung zum Build-Tool Gradle finden Sie in Anhang A.

- <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- <https://gradle.org/gradle-download/>

Mac-User können zur Installation von Gradle alternativ und bequemer auch `brew install gradle` nutzen.

Zur Arbeit mit bzw. zum Import des Projekts in Eclipse benötigen Sie ein aktuelles Eclipse 4.5 oder neuer.

- <https://eclipse.org/downloads/>

Beispielprojekte und Programme „installieren“

Laden Sie die Programmbeispiele als ZIP-Archiv von der Seite www.dpunkt.de/java-persistenz herunter und entpacken Sie dieses in ein beliebiges Verzeichnis.

Start der Programme

Das Ausführen der Programme basiert auf Gradle. Im Hauptverzeichnis liegt eine Gradle-Build-Datei `build.gradle`, mit der sich alle im Buch als ausführbar angegebene Programme starten lassen. Öffnen Sie eine Konsole und starten das jeweilige Programm durch Aufruf von

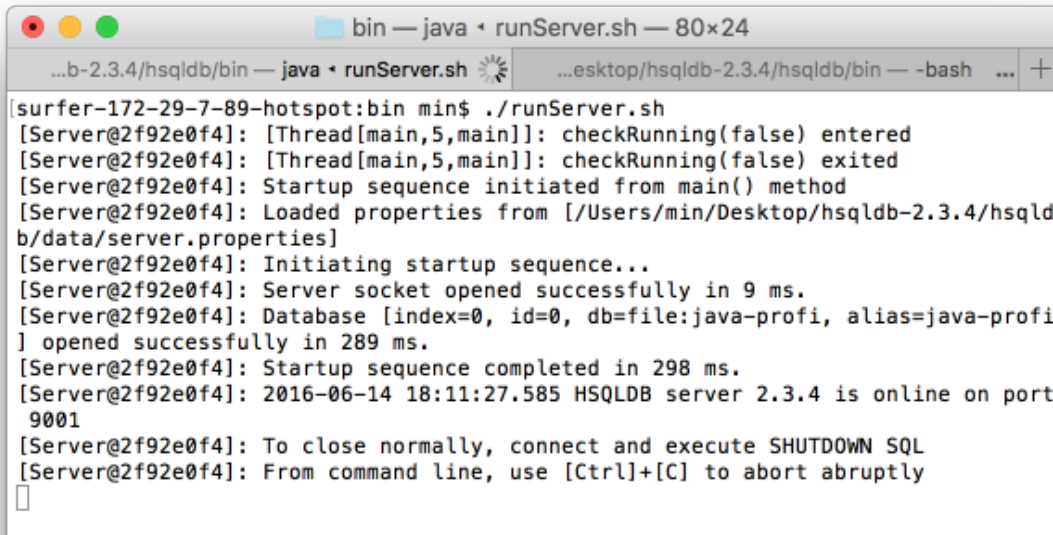
```
gradle <PROGRAMMNAME>
```

Beim ersten Start werden eventuell diverse referenzierte Bibliotheken aus dem Internet heruntergeladen, dann dauert der Build und Programmstart eine Weile. Spätere Starts sollten dann deutlich schneller vonstattengehen.

Voraussetzungen zur Arbeit mit Datenbanken

HSQldb

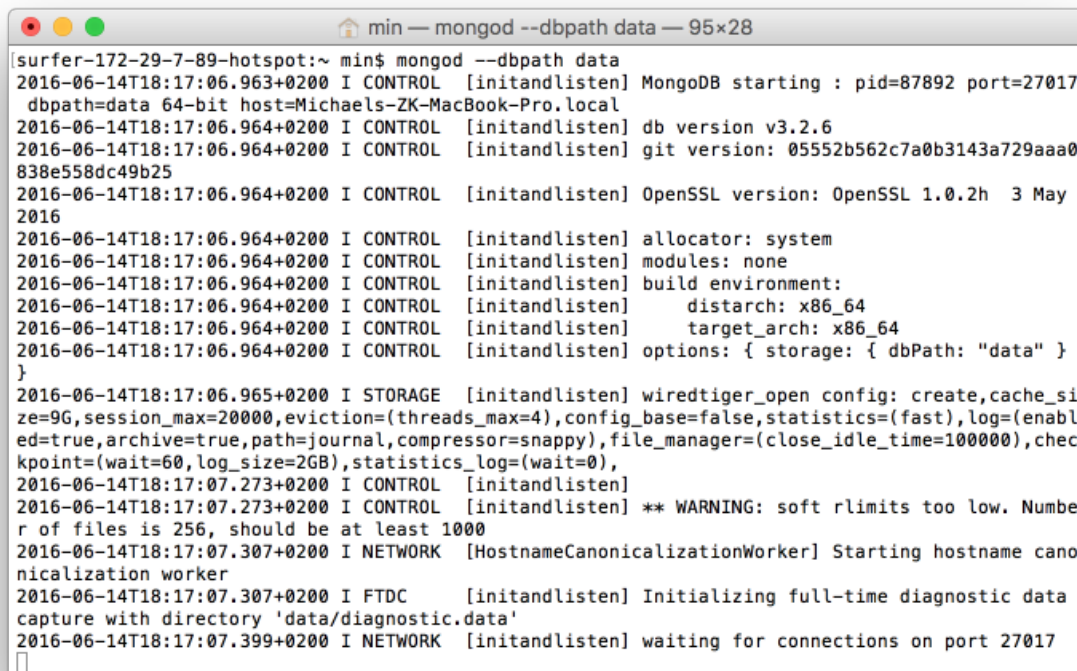
Um die Beispiele zu SQL, JDBC und JPA auszuführen, muss eine Datenbank laufen. Bitte nutzen Sie dafür die Datenbank HSQldb. Installieren Sie HSQldb in der aktuellen Version 2.3.4 wie im Buch beschrieben und starten dann den Server mit dem Skript runServer.



```
surfer-172-29-7-89-hotspot:bin min$ ./runServer.sh
[Server@2f92e0f4]: [Thread[main,5,main]]: checkRunning(false) entered
[Server@2f92e0f4]: [Thread[main,5,main]]: checkRunning(false) exited
[Server@2f92e0f4]: Startup sequence initiated from main() method
[Server@2f92e0f4]: Loaded properties from [/Users/min/Desktop/hsqldb-2.3.4/hsqldb/data/server.properties]
[Server@2f92e0f4]: Initiating startup sequence...
[Server@2f92e0f4]: Server socket opened successfully in 9 ms.
[Server@2f92e0f4]: Database [index=0, id=0, db=file:java-profi, alias=java-profi] opened successfully in 289 ms.
[Server@2f92e0f4]: Startup sequence completed in 298 ms.
[Server@2f92e0f4]: 2016-06-14 18:11:27.585 HSQldb server 2.3.4 is online on port 9001
[Server@2f92e0f4]: To close normally, connect and execute SHUTDOWN SQL
[Server@2f92e0f4]: From command line, use [Ctrl]+[C] to abort abruptly
```

MongoDB

Um die Beispiele zu MongoDB auszuführen, muss MongoDB gestartet sein, etwa mit `mongod --dbpath data`:

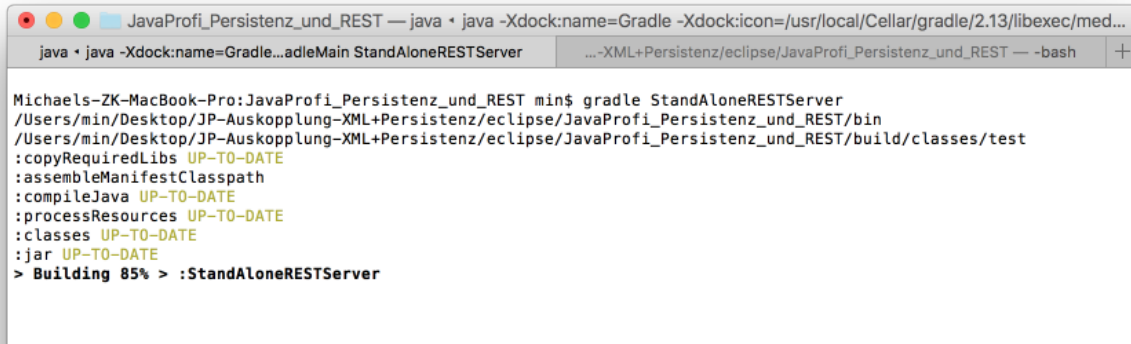


```
surfer-172-29-7-89-hotspot:~ min$ mongod --dbpath data
2016-06-14T18:17:06.963+0200 I CONTROL [initandlisten] MongoDB starting : pid=87892 port=27017
dbpath=data 64-bit host=Michaels-ZK-MacBook-Pro.local
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] db version v3.2.6
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] git version: 05552b562c7a0b3143a729aaa0838e558dc49b25
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2h 3 May 2016
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] allocator: system
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] modules: none
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] build environment:
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] distarch: x86_64
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] target_arch: x86_64
2016-06-14T18:17:06.964+0200 I CONTROL [initandlisten] options: { storage: { dbPath: "data" } }
}
2016-06-14T18:17:06.965+0200 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=9G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2016-06-14T18:17:07.273+0200 I CONTROL [initandlisten]
2016-06-14T18:17:07.273+0200 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
2016-06-14T18:17:07.307+0200 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2016-06-14T18:17:07.307+0200 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'data/diagnostic.data'
2016-06-14T18:17:07.399+0200 I NETWORK [initandlisten] waiting for connections on port 27017
```

Für das Beispiel `HTTPURLCONNECTIONMONGODBEXAMPLE` muss MongoDB mit aktiviertem REST-Interface gestartet werden: `mongod --rest --dbpath data`

Voraussetzungen zur Arbeit mit REST-Server und Clients

Wenn Sie die Beispiele zu den REST-Servern ausprobieren möchte, starten Sie den gewünschten Server in einem eigenen Konsolenfenster, etwa mit gradle StandAloneRESTServer wie folgt:



```
Michael's-MacBook-Pro:JavaProfi_Persistenz_und_REST min$ gradle StandAloneRESTServer
/Users/min/Desktop/JP-Auskopplung-XML+Persistenz/eclipse/JavaProfi_Persistenz_und_REST/bin
/Users/min/Desktop/JP-Auskopplung-XML+Persistenz/eclipse/JavaProfi_Persistenz_und_REST/build/classes/test
:copyRequiredLibs UP-TO-DATE
:assembleManifestClasspath
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:jar UP-TO-DATE
> Building 85% > :StandAloneRESTServer
```

In einem separaten Fenster können Sie dann den gewünschten Client ausführen, beispielsweise per gradle HelloClientExample.

Informationen rund um Eclipse

Im Buch finden Sie weitere Informationen zu den Programmen. Die Quelltexte finden Sie im Unterorder src. Ein Eclipse-Projekt zum Import finden Sie im Hauptverzeichnis. Weil JavaFX in Eclipse als restricted API erkannt wird, müssen wir noch eine Einstellung vornehmen:

