

Entwicklungsumgebung BiDiB-Wizard

Der BiDiB-Wizard und die Bibliothek jbidibc sind in *Java* geschrieben. Die Projekte basieren auf *Maven* (maven3). Um die beiden Projekte zu builden, muss folgendes installiert sein:

- Java Development Kit (JDK 8),
- Maven 3
- git Client (z.B. [PortableGit](#), TortoiseGit oder SourceTree unter Windows, git Client für Linux ist meistens schon vorinstalliert)
- Entwicklungsumgebung (z.B. [Eclipse Neon - Eclipse IDE for Java Developers](#))

JDK und Maven installieren

JDK 8 von [Oracle Java SE Downloads](#) runterladen und installieren.

Maven 3 als *Binary zip archive* (Windows) oder *Binary tar.gz archive* (Linux) von [Apache Maven](#) downloaden und durch Entpacken *installieren*.



Hinweis: Maven wird während dem ersten Build sehr viele Abhängigkeiten (Dependencies) herunterladen!

Die Konfigurationsdatei von Maven liegt unter <Installationspfad>/conf/settings.xml. Hier kann man Einstellungen anpassen um z.B. den Ablageort des lokalen *Maven-Repository* festzulegen. Standardmässig wird das *Maven-Repository* unter dem Benutzerverzeichnis angelegt (.m2/repository). Wenn man das *Maven-Repository* an einen anderen Ort legen will muss man im settings.xml den Eintrag <localRepository> machen/anpassen:

```
<localRepository>D:/ .m2/repository</localRepository>
```

In diesem Fall liegt das lokale Maven-Repository dann unter D:/ .m2/repository.

Damit Maven funktioniert muss das JDK in den Pfad aufgenommen werden. Dazu das JDK (C:\Program Files\Java\jdk-1.8.x\bin) entweder in den globalen Pfad hinzufügen oder eine Batch-Datei bauen, welche den Pfad setzt.

Ich verwende z.B. folgendes Skript:

[setenv-maven-3.3.9.cmd](#)

```
SET JAVA_HOME=C:\Program Files\Java\jdk1.8.0_73
SET M2_HOME=D:\tools\apache-maven-3.3.9
SET GIT_HOME=D:\tools\PortableGit-2.10.0

SET PATH=%M2_HOME%\bin;%JAVA_HOME%\bin;%GIT_HOME%\bin;%PATH%

SET MAVEN_OPTS=-Xmx1024m
```

```
rem mvn clean install
rem mvn clean install -DskipTests=true

call CMD.EXE
```

Durch Doppelklick im Explorer wird eine Console (Command Prompt) geöffnet und man kann folgendes Maven-Kommando eingeben: `mvn -version`

```
D:\git\jbidibc>mvn --version
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5;
2015-11-10T17:41:47+01:00)
Maven home: D:\tools\apache-maven-3.3.9
Java version: 1.8.0_73, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_73\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

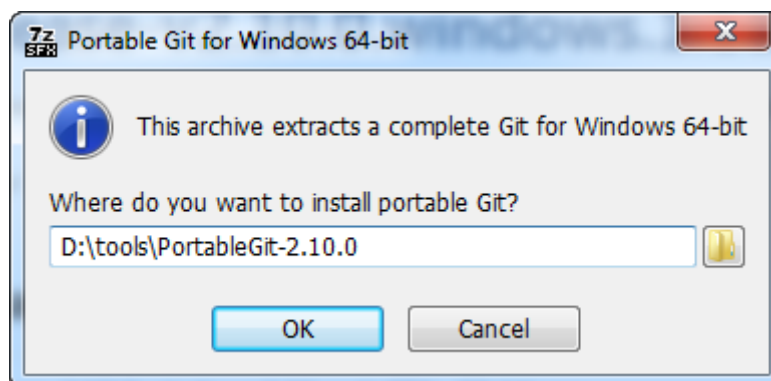
Maven wurde erfolgreich installiert



PortableGit installieren

PortableGit kann von Github heruntergeladen werden: <https://github.com/git-for-windows/git/releases>
Unter dem Punkt *Git for Windows* findet man mehrere Installationsmöglichkeiten, unter anderem auch *PortableGit*. Ich habe in diesem Beispiel <https://github.com/git-for-windows/git/releases/download/v2.10.0.windows.1/PortableGit-2.10.0-64-bit.7z.exe> heruntergeladen.

Anschliessend das exe ausführen und den Ort der Installation wählen:



Anschliessend werden die Git-Programmdateien in dieses Verzeichnis kopiert.

Clonen der Sourcen

Die Sourcen sind in einem git-Repo auf Sourceforge abgelegt. Um die aktuellen Sourcen auf den Rechner zu bekommen müssen die Repositories gecloned werden. Dies erfolgt mit git über das HTTP-Protokoll (geht auch durch Firewalls) oder das git-Protokoll.

jbidibc

Das jbidibc-Git-Repository ist unter <http://git.code.sf.net/p/jbidibc/code/> erreichbar.



Das Git-Repository ist nicht das gleiche wie das Maven-Repository.

Um das clonen zu starten kann im Verzeichnis wo die Sourcen abgelegt werden sollen folgender Befehl ausgeführt werden:

```
git clone http://git.code.sf.net/p/jbidibc/code jbidibc-code
```

Dadurch werden die Sourcen in das neue Unterverzeichnis jbidibc-code geladen.

Das nachfolgende Snippet zeigt den clone-Vorgang:

```
D:\temp>git clone http://git.code.sf.net/p/jbidibc/code jbidibc-code
Cloning into 'jbidibc-code'...
remote: Counting objects: 23829, done.
remote: Compressing objects: 100% (9043/9043), done.
remote: Total 23829 (delta 9654), reused 19682 (delta 8209)
Receiving objects: 100% (23829/23829), 4.02 MiB | 753.00 KiB/s, done.
Resolving deltas: 100% (9654/9654), done.
Checking out files: 100% (653/653), done.

D:\temp>git --version
git version 2.10.0.windows.1
```

Builden auf Kommandozeile

Nach dem Wechsel in das Verzeichnis mit den Sourcen kann man direkt das Maven-Kommando eingeben: `mvn clean install`

Dieses Kommando startet den Maven-Build. Wenn der Build erfolgreich durchgelaufen ist kommt am Ende folgende Ausgabe:

```
[INFO] -----
---
[INFO] Reactor Summary:
[INFO]
[INFO] jBiDiB :: Parent ..... SUCCESS [ 1.096
s]
[INFO] jBiDiB :: jbidibc Core ..... SUCCESS [ 17.867
s]
[INFO] jBiDiB :: jbidibc Serial ..... SUCCESS [ 2.602
s]
[INFO] jBiDiB :: jbidibc RXTX ..... SUCCESS [ 0.489
s]
[INFO] jBiDiB :: jbidibc Net ..... SUCCESS [ 1.901
```

```
s]
[INFO] jBiDiB :: jbidibc Simulation ..... SUCCESS [ 4.214
s]
[INFO] jBiDiB :: jbidibc Tools ..... SUCCESS [ 1.374
s]
[INFO] jBiDiB :: jbidibc Assembly ..... SUCCESS [ 1.426
s]
[INFO] jBiDiB :: jbidibc SCM ..... SUCCESS [ 1.717
s]
[INFO] jBiDiB :: jbidibc Debugger ..... SUCCESS [ 2.027
s]
[INFO] jBiDiB :: jbidibc Decoder ..... SUCCESS [ 6.116
s]
[INFO] jBiDiB :: jbidibc Exchange ..... SUCCESS [ 9.246
s]
[INFO] jBiDiB :: jbidibc Experimental ..... SUCCESS [ 2.750
s]
[INFO] jBiDiB :: jbidibc POM update ..... SUCCESS [ 2.705
s]
[INFO] jBiDiB :: jbidibc UI components ..... SUCCESS [ 2.578
s]
[INFO] -----
---
[INFO] BUILD SUCCESS
[INFO] -----
---
[INFO] Total time: 58.378 s
[INFO] Finished at: 2016-09-15T22:14:10+02:00
[INFO] Final Memory: 51M/467M
[INFO] -----
---
```

Keine Angst, nachher geht das aus Eclipse raus ...



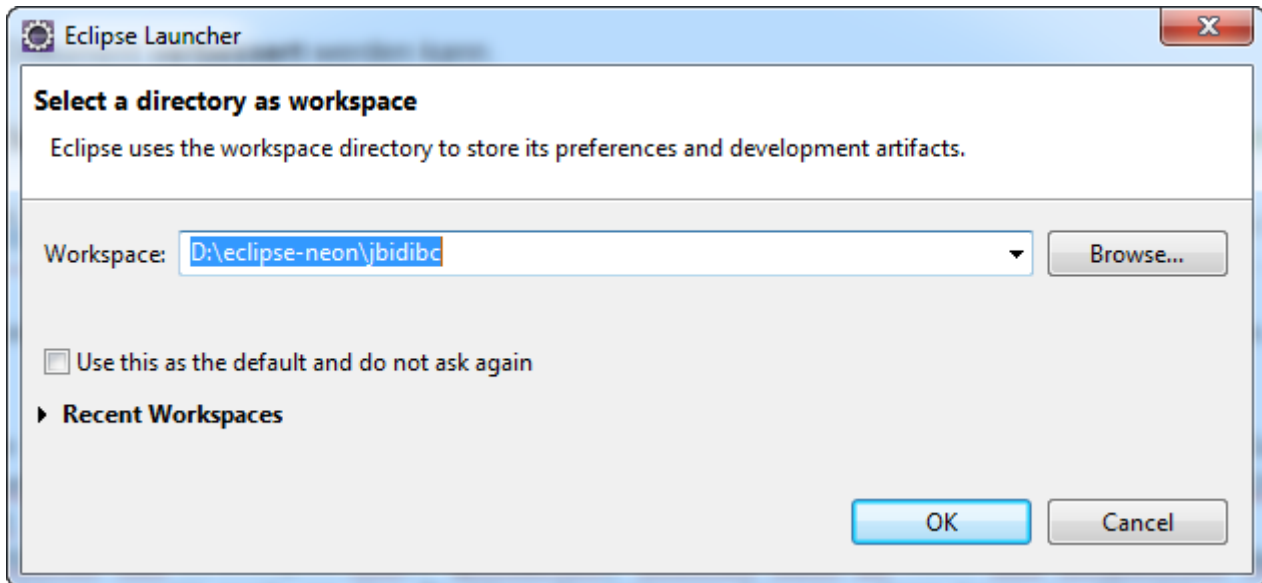
Eclipse IDE installieren

Download von der Eclipse Seite: <https://eclipse.org/downloads/eclipse-packages/>

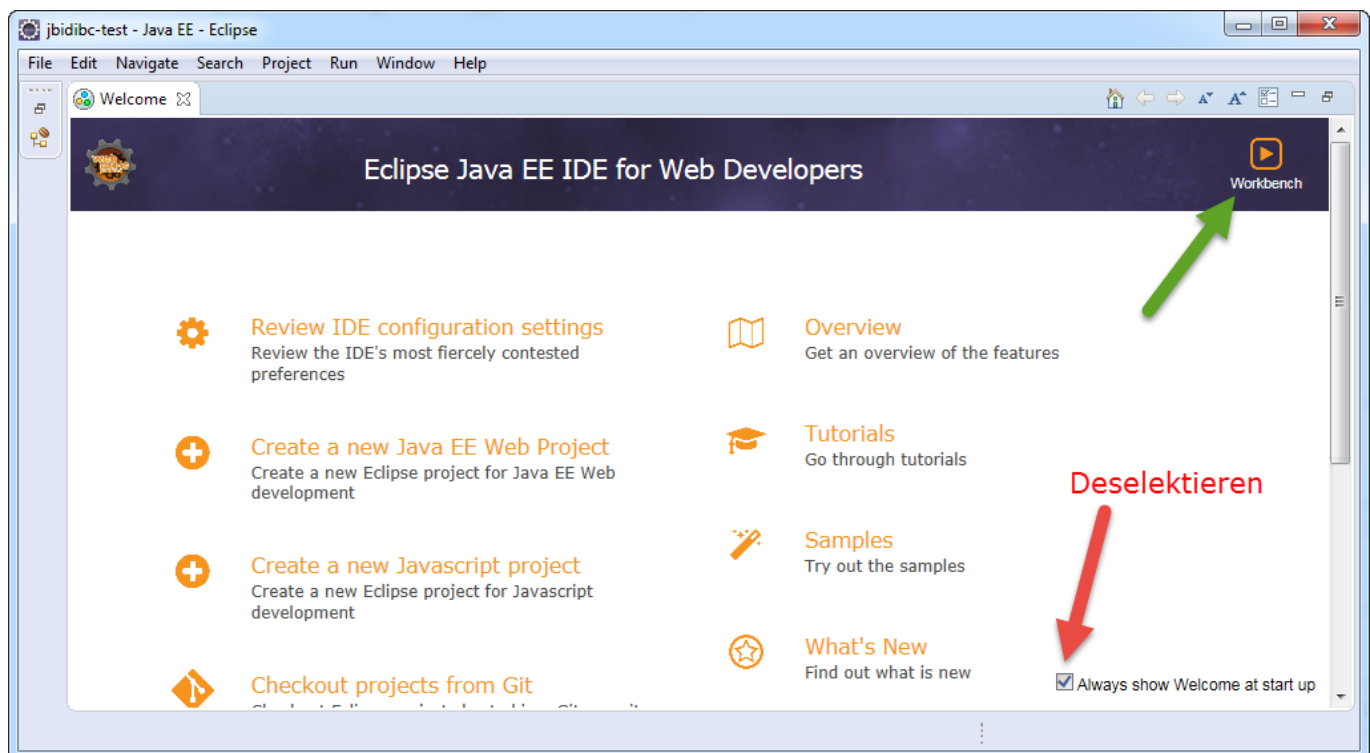
Hier kann die *Eclipse IDE for Java Developers* verwendet werden. In dieser Version sind die Plugins für Maven und Git schon installiert. Nach dem Download des für das OS passenden Pakets (ZIP) kann der Inhalt an einen beliebigen Ort entpackt werden. Eclipse wird durch Ausführen von *eclipse.exe* gestartet, welches sich im Root des entpackten Verzeichnis befindet. Man kann *eclipse.exe* auch an der Taskbar festpinnen lassen, dann kann es direkt aus der Taskbar gestartet werden.

Eclipse arbeitet mit sogenannten *Workspaces* in welchen die Daten abgelegt werden, welche von Eclipse zur Laufzeit benötigt werden. Wenn man mehrere Projekte hat, kann man für jedes Projekt eine andere Workspace verwenden. Man kann Eclipse auch mehrfach starten und dabei mehrere Workspaces parallel offen haben (den gleichen Workspace nicht mehrfach!).

Nach dem Starten von Eclipse kommt der Auswahldialog wo der Workspace erzeugt werden soll.



Anschließend wird eine Begrüssungsseite auf der man am besten die Checkbox zum Anzeigen bei jedem Neustart deselektiert.



Danach kann in die Workbench gewechselt werden. Dieser Schritt entfällt beim nächsten Neustart wenn der gleiche Workspace verwendet wird.



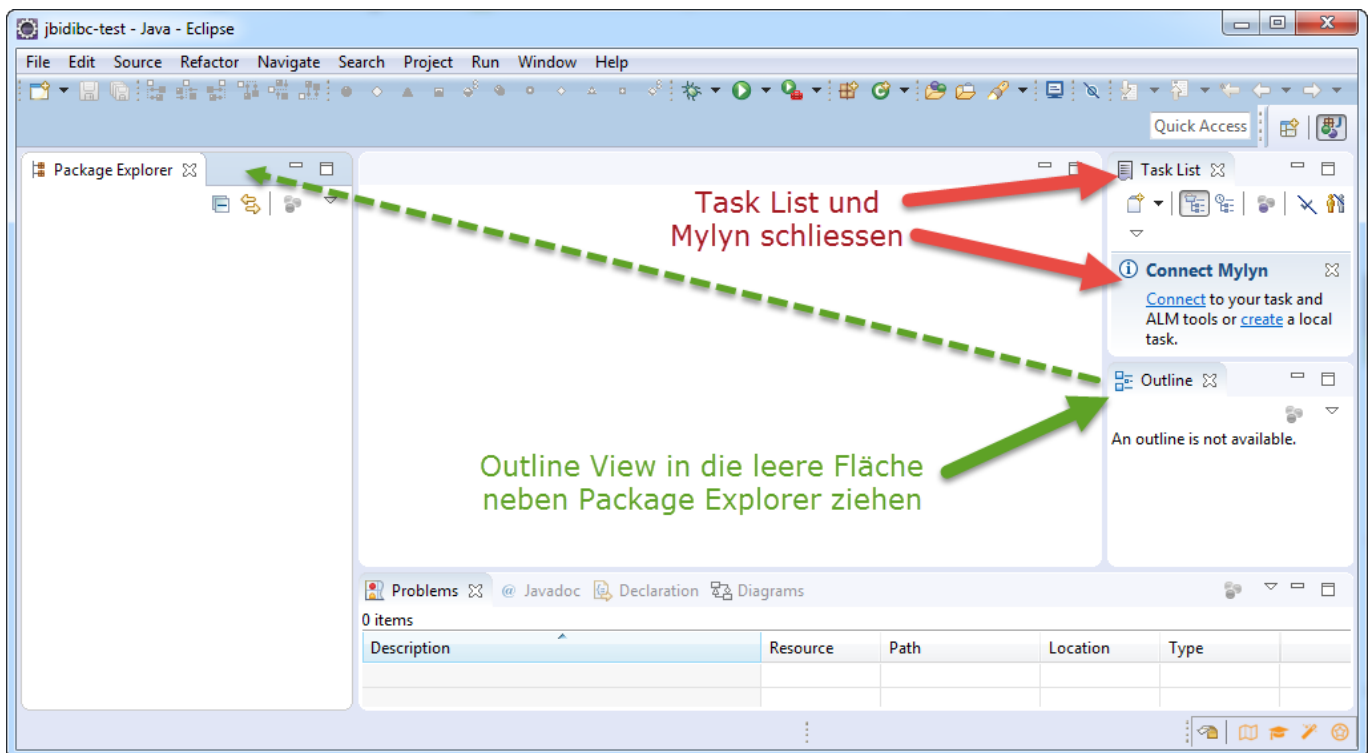
Gleich noch ein Hinweis: In ganz seltenen Fällen (... meistens nach Updates von Eclipse-Plugins ...) kann der Workspace korrupt werden (zumindest war es früher so). Dann hilft nur noch das Löschen und Neuanlegen des Workspace. Und gleich der nächste Hinweis: Das ist mir aber in den

letzten Jahren nicht mehr oft passiert



Jetzt noch den Desktop einrichten (sind nur Vorschläge):

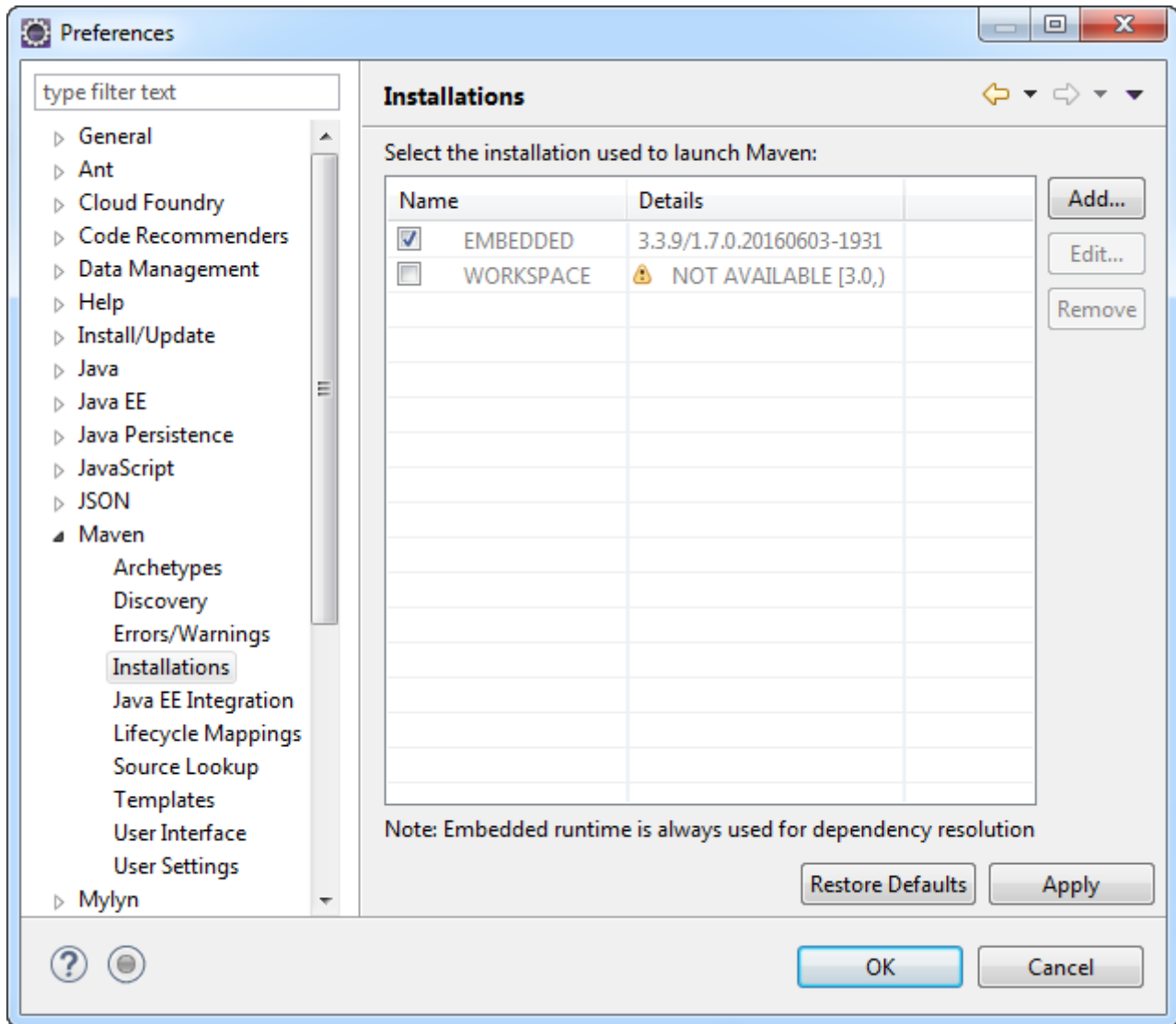
- Mylyn und Task List schliessen
- Outline View neben den Package Explorer verschieben



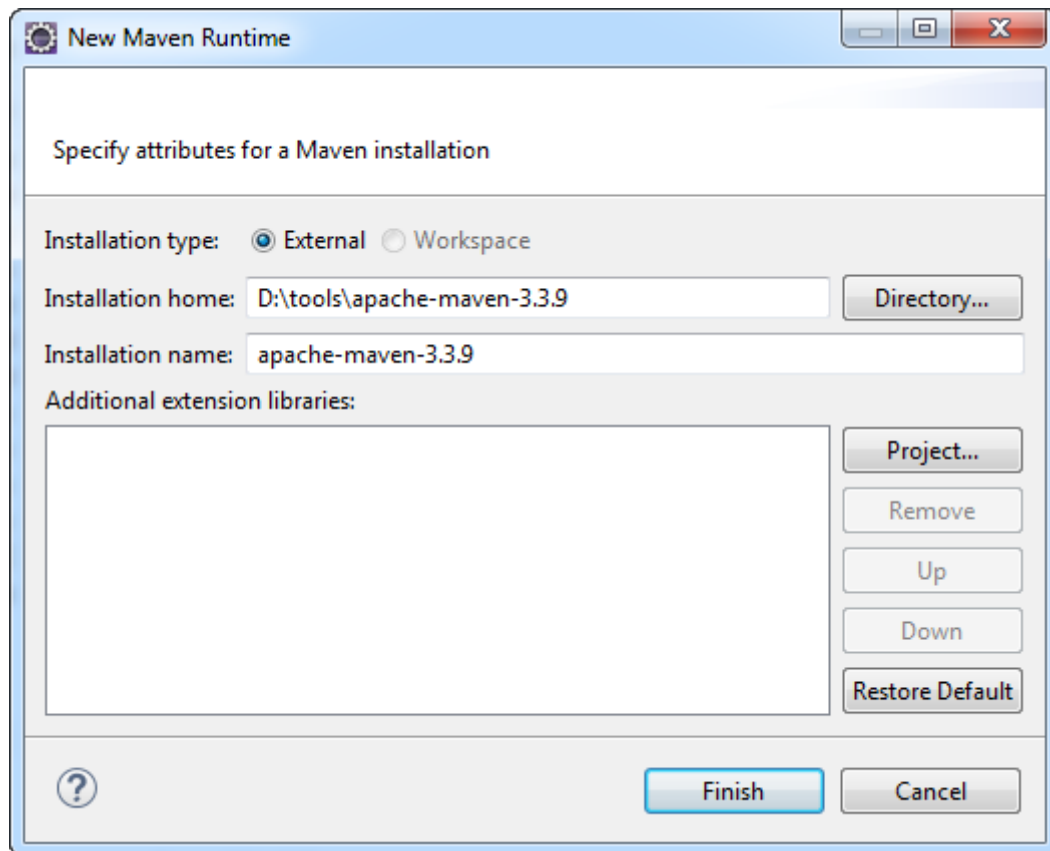
Maven-Integration einrichten

Anschließend muss die Maven-Installation ausgewählt werden, welche verwendet werden soll. Das ist wichtig weil sonst unter Umständen das lokale Maven-Repository an einem anderen Ort (default ...) nochmals angelegt wird.

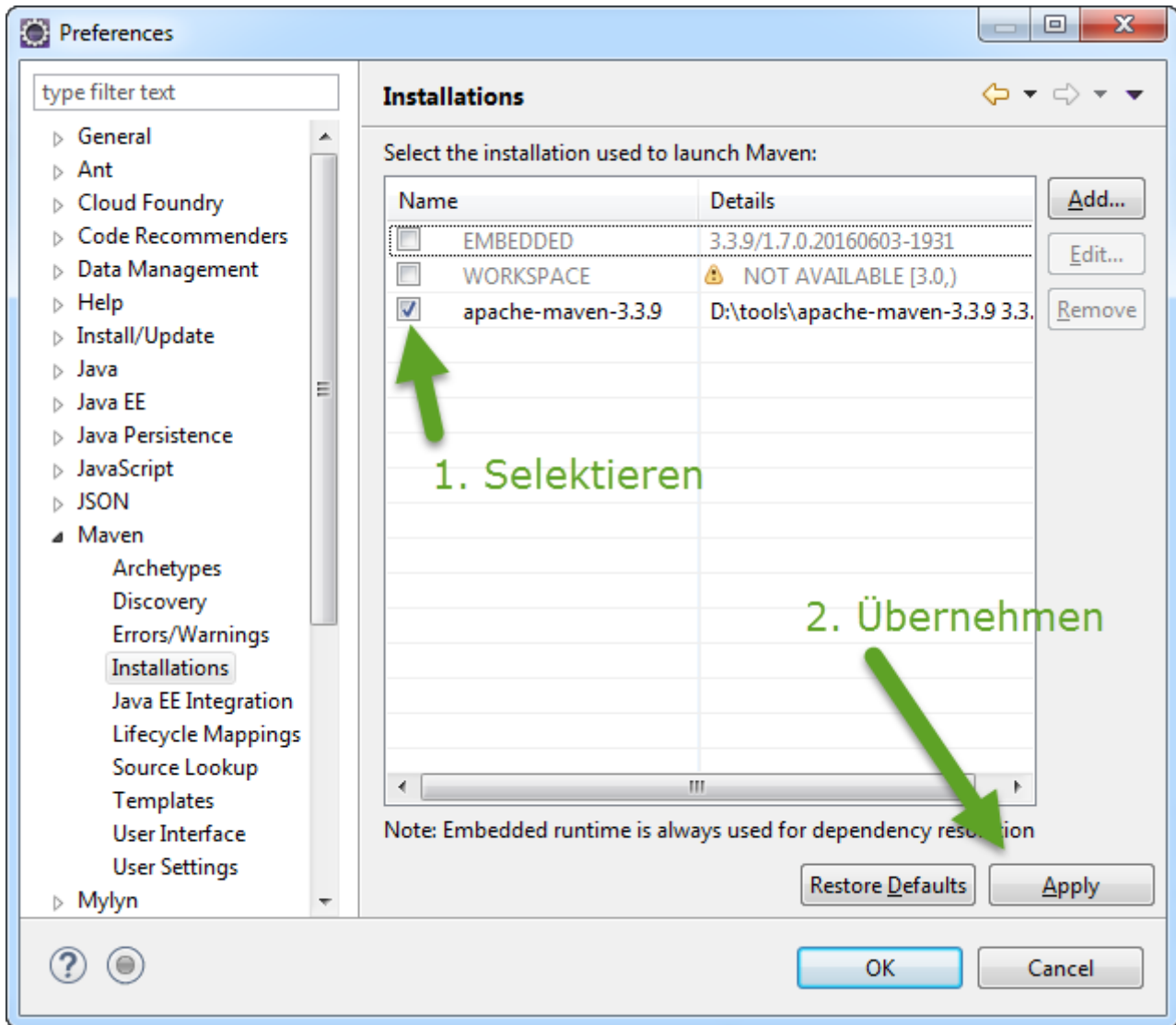
Über das Menü Windows > Preferences wird der Preferences-Dialog geöffnet. Im Preferences-Dialog dann zu Maven > Installations wechseln.



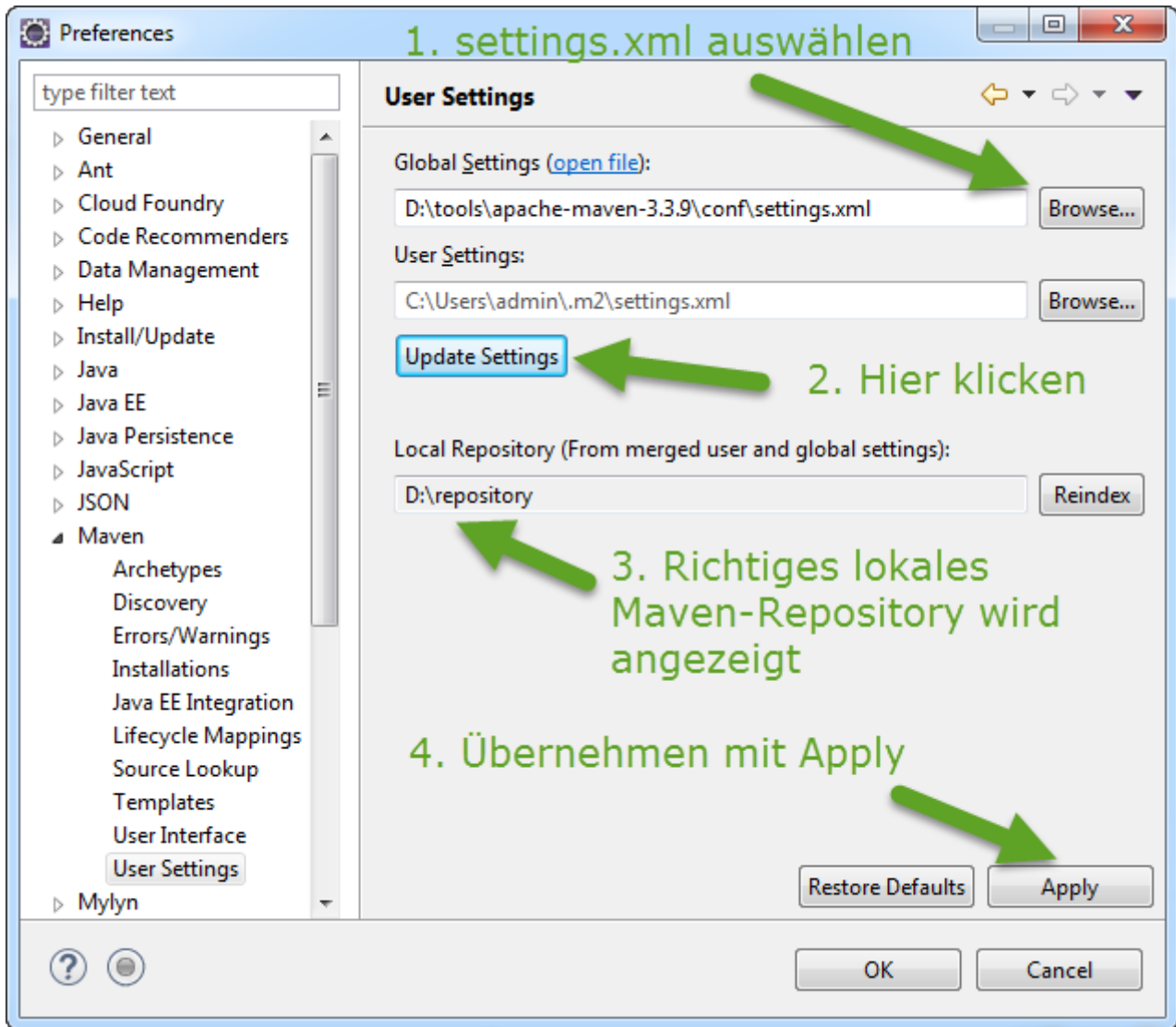
Mit Klick auf Add öffnet sich der folgende Dialog:



Über den *Directory*-Button kann das Verzeichnis der lokalen Maven-Installation ausgewählt werden und nach dem Klick auf *Finish* wird die lokale Installation in der Liste angezeigt. Diese Installation muss jetzt noch selektiert (Checkbox aktivieren) werden und die Änderungen mit *Apply* übernommen werden.



Anschliessend in Maven > User Settings nachschauen ob das *Local Repository* an den gleichen Ort zeigt wie die Daten aus den Schritten oben. Normalerweise ist dies nicht der Fall. Deshalb muss unter *Global Settings* das *settings.xml* der lokale Maven-Installation (unter *conf/settings.xml*) ausgewählt werden. Ein Klick auf *Update Settings* sollte dann unter *Local Repository* das lokale Maven-Repository anzeigen.



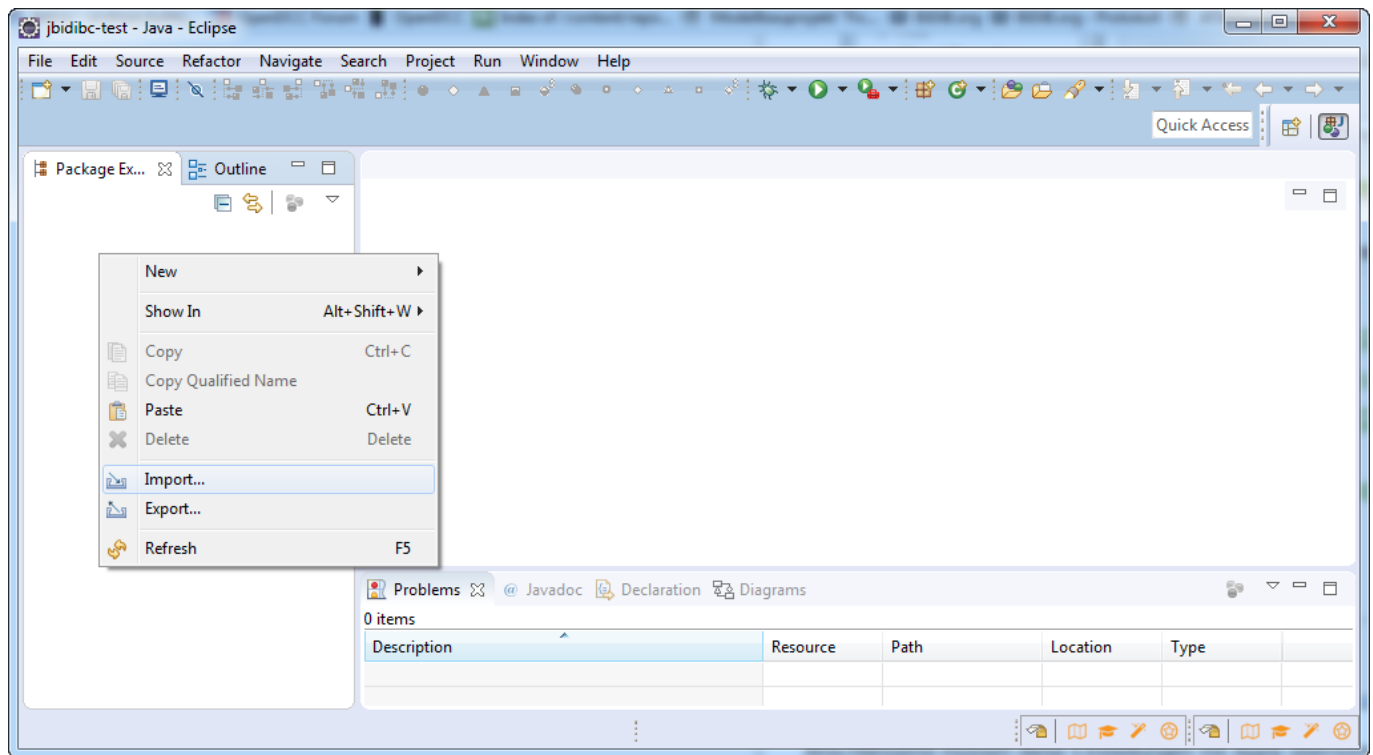
Anschließend müssen diese Einstellungen mit *Apply* übernommen werden. Danach den Settings-Dialog mit *OK* schliessen.



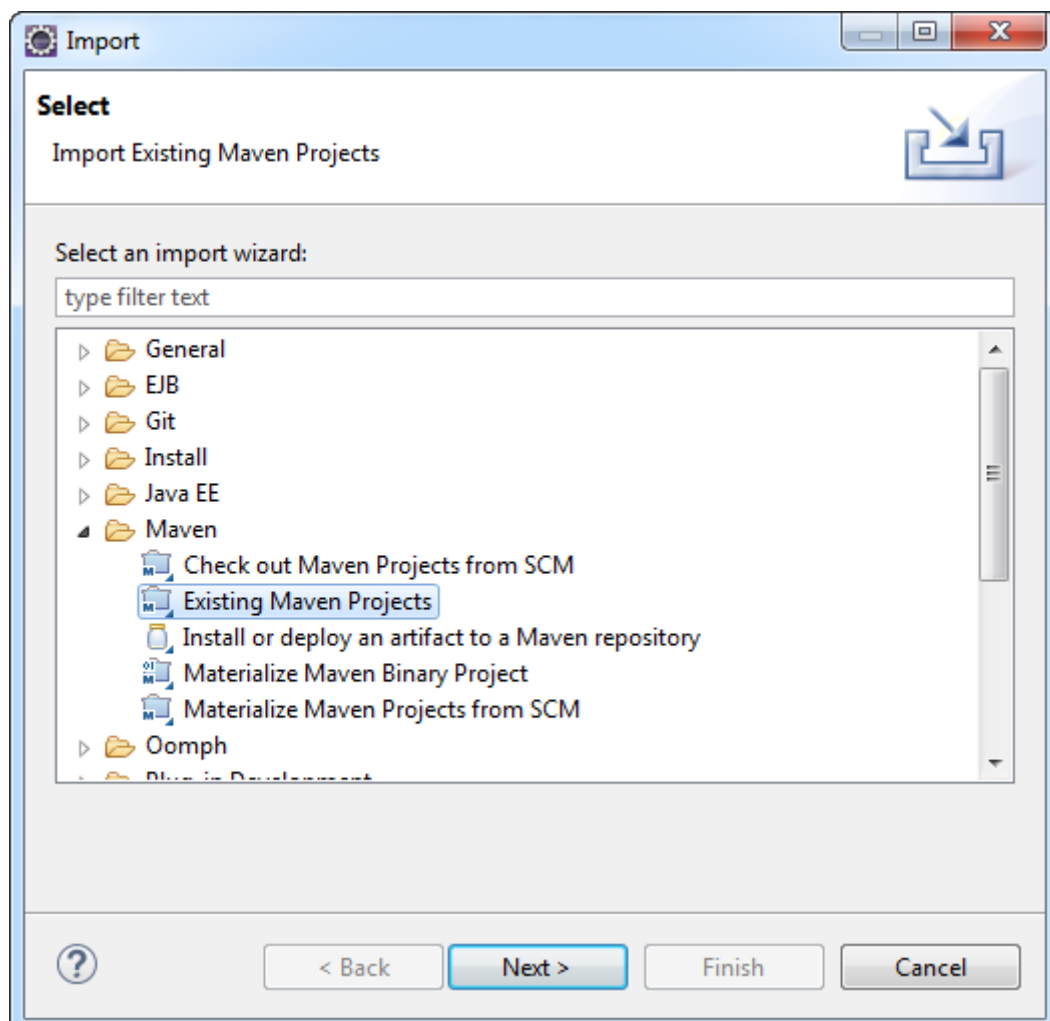
Hinweis: Man kann das *settings.xml* auch im User-Home unter *.m2* ablegen. Wenn dort eine Datei mit dem Namen *settings.xml* existiert, überschreiben die Informationen in dieser Datei die „globalen“ Daten. Der Vorteil von Maven-Settings in *<userhome>/.m2/settings.xml* ist, dass man die Maven-Settings für mehrere Versionen von Maven definieren kann, statt bei jeder Maven-Version immer das *settings.xml* unter */conf* anpassen zu müssen. Falls das jemand machen will: Man kann mit dem Explorer kein Verzeichnis oder Datei mit einem „.“ beginnend erzeugen, also „.m2“ geht nicht. Es funktioniert aber aus der Kommandozeile heraus, also Command Prompt aufmachen, in das Verzeichnis wechseln und dort `md .m2` ausführen.

Maven-Projekte importieren

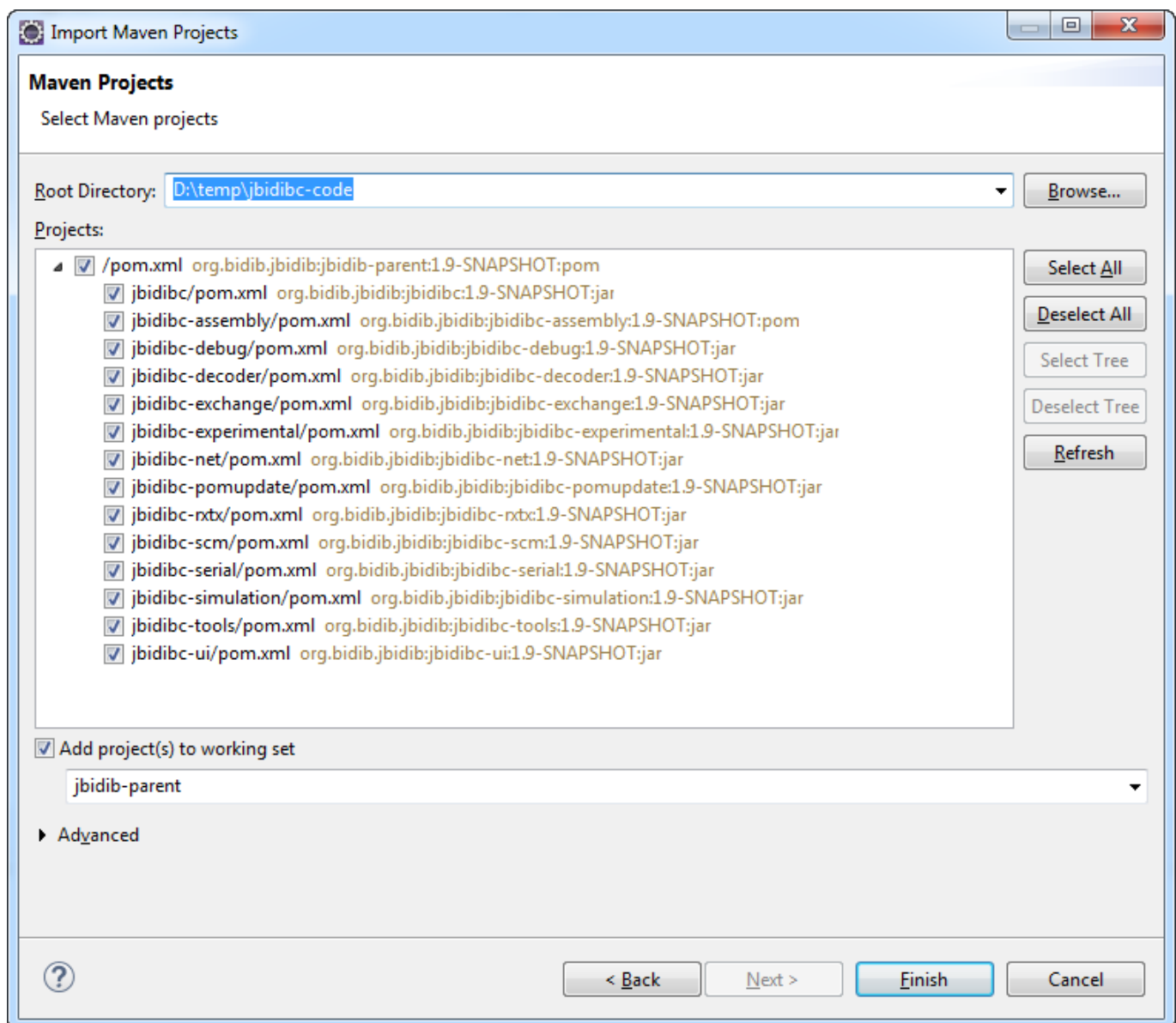
Als nächster Schritt werden die Maven-Projekte in Eclipse importiert. Dazu muss die Package Explorer View aktiviert werden und dort über das Kontext-Menü *Import ...* (oder über das Menü *File > Import ...*) der Import-Dialog aufgerufen werden.



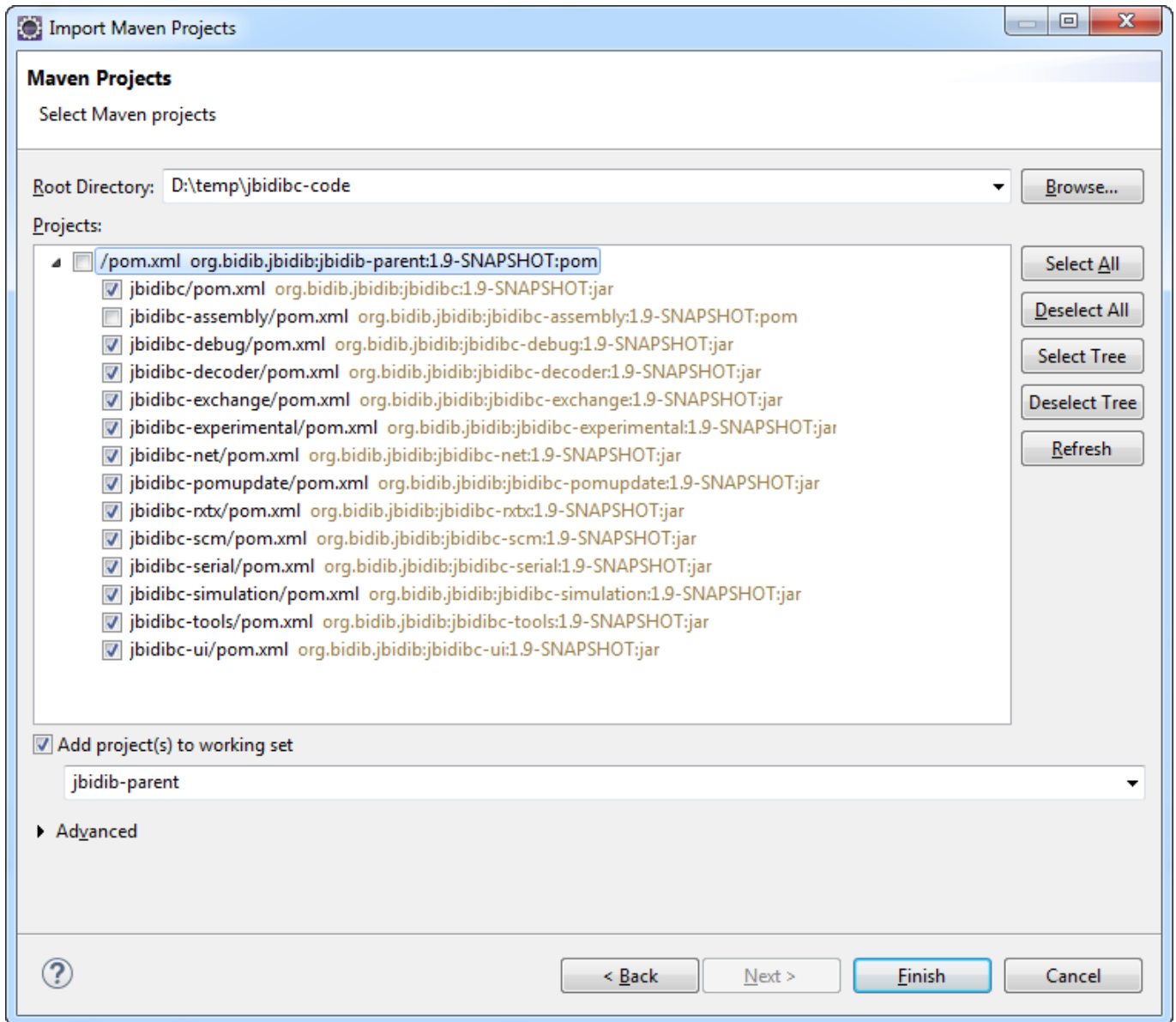
Im Import-Dialog steht unter *Maven > Existing Maven Projects* ein Wizard zum Importieren der Maven-Projekte zur Verfügung.



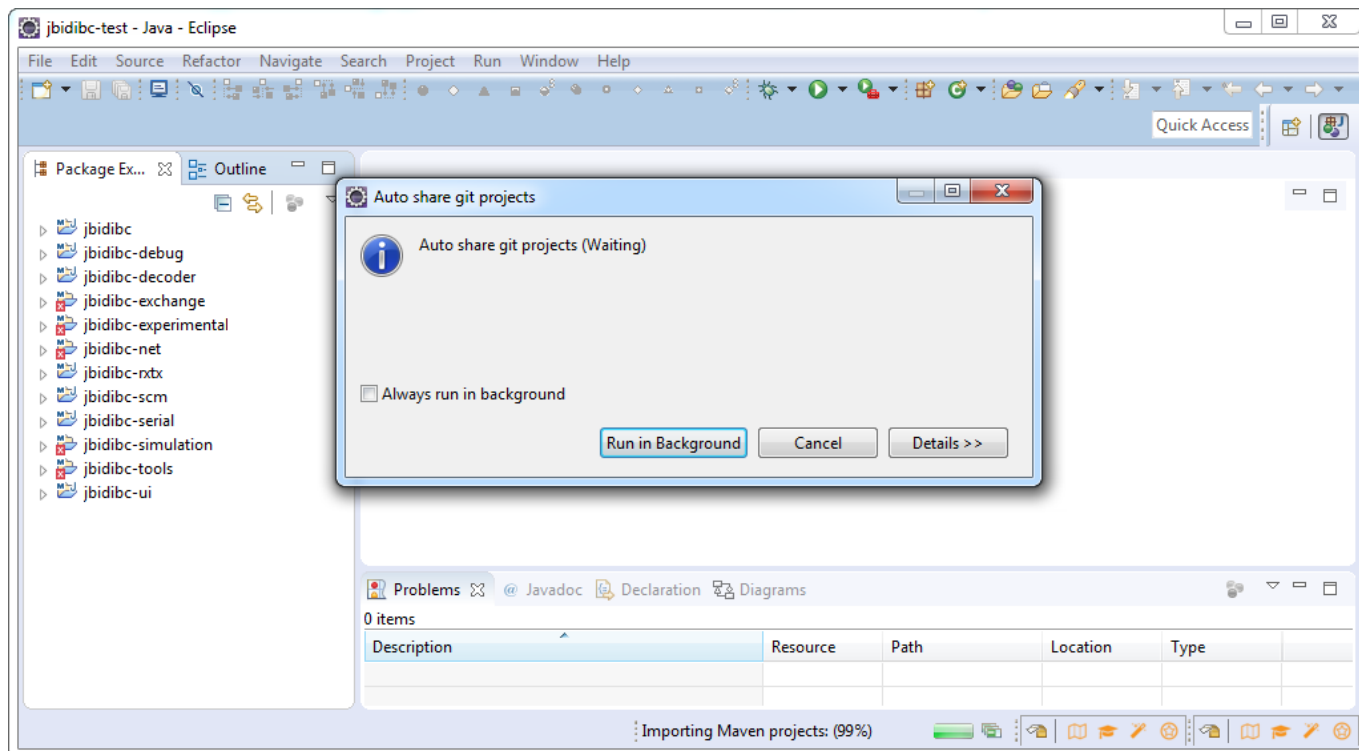
Im nächsten Schritt wird als *Root Directory* das Verzeichnis mit den Quellen ausgewählt. Das ist das Verzeichnis wohin die Quellen gecloned wurden. Anschliessend werden unter *Projects* alle Maven-Module angezeigt die gefunden wurden. Als *Maven-Module* bezeichne ich die Sub-Projekte des Maven-Projekt.



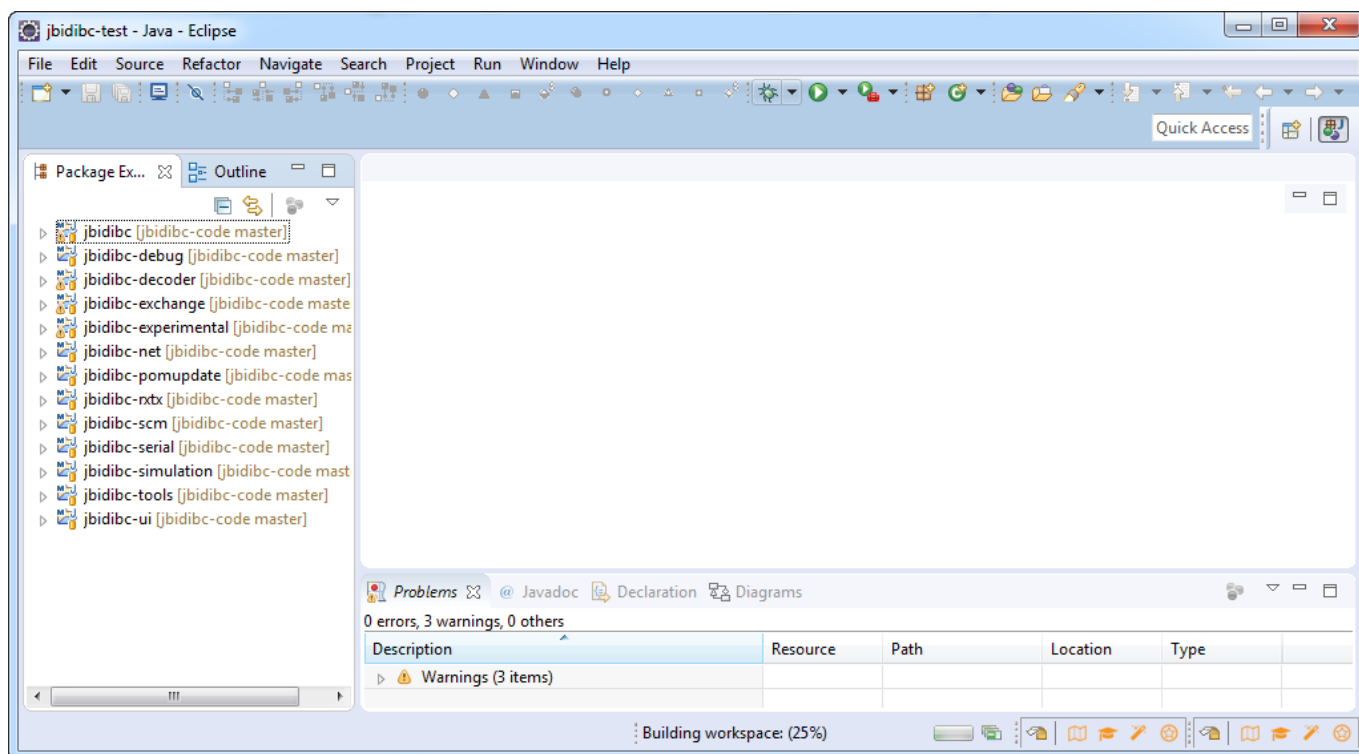
Da nicht alle *Maven-Module* gebraucht werden, kann man jetzt das oberste `pom.xml` und das Modul `jbidibc-assembly/pom.xml` deselektieren.



Nach dem Klick auf *Finish* werden die Maven-Module in den Eclipse Workspace importiert. Dabei erkennt Eclipse dass die Projekte mit Git verwaltet werden und stellt automatisch eine entsprechende Verbindung her.



Wenn der Import abgeschlossen ist, werden die Maven-Module im *Package Explorer* angezeigt und gebildet.



From:
<https://forum.opendcc.de/wiki/> - BiDiB Wiki

Permanent link:
<https://forum.opendcc.de/wiki/doku.php?id=wizard:development&rev=1474120118>

Last update: **2016/09/17 15:48**



