Aufsetzen einer Entwicklungsumgebung

Entwicklungstools

Als IDE wird in dieser Anleitung die auf Eclipse basierende IDE Spring Tool Suite in der Version spring-tool-suite-3.2.0.RELEASE verwendet.

Es können auch Eclipse oder Netbeans verwendet werden. Allerdings müssen dann viele Plugins selber nachinstalliert werden wie z.B. der Maven Support oder die GIT Integration.

Benötigte Software installieren

Für die Nuclos Entwicklung muss Oracle Java verwendet werden, IcedTea o.ä funktioniert nicht zuverlässig. Wer sich unsicher ist welche JVM Standardmäßig verwendet wird, kann das wie folgt prüfen:

Java Installation prüfen

```
$ java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
```

Installation der IDE:

Download der Spring Tool Suite und Installation nach Anleitung von SpringSource.

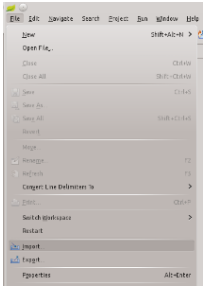
(OPTIONAL) Installation von GIT (Versionsverwaltung).

Installation GIT (Linux)

```
sudo apt-get install git
```

Nuclos (und Nuclos API) Repository auschecken

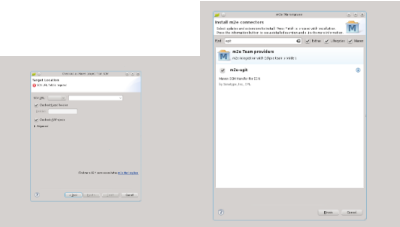
Nuclos steht als Maven Projekt zur Verfügung. SpringSource importiert das Git Projekt und erzeugt auf Grundlage der Mavenkonfiguration die benötigten Projekte im geöffneten Workspace.



Beim ersten Projektimport muss ein Connector für GIT bzw. EGIT installiert werden, da dieser normalerweise nicht vorhanden ist.

Über den Link *m2e Marketplace* wird ein Auswahlfenster geöffnet in dem der *m2e-egit* Connector ausgewählt werden kann.

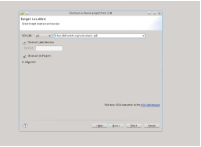
SpringSource wird den Connector herunterladen, installieren und die IDE neustarten.



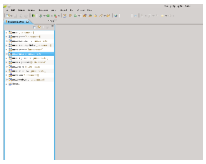
Sobald der Connector installiert wurde, kann er an dieser Stelle ausgewählt werden.

Als *SCM URL* wird das Repository, also https://bitbucket.org/nuclos/nuclos.git für nuclos, https://bitbucket.org/nuclos/nuclos-api.git für nuclos-api eingetragen.

Das Auschecken und Importieren des Projektes dauert je nach Auslastung von System und Netzwerk etwas länger.



Wenn keine Probleme aufgetreten sind, enthält der *Package Explorer* nach dem Vorgang mehrere Projekte. Diese stellen die einzelnen Module von Nuclos dar.



Bei Fehlern bitte den Bereich Troubleshooting beachten.

Nuclos Server einrichten

Apache Tomcat einrichten

Für den Nuclos Server wird der Tomcat Application Server in der Version 7 (Core) benötigt.

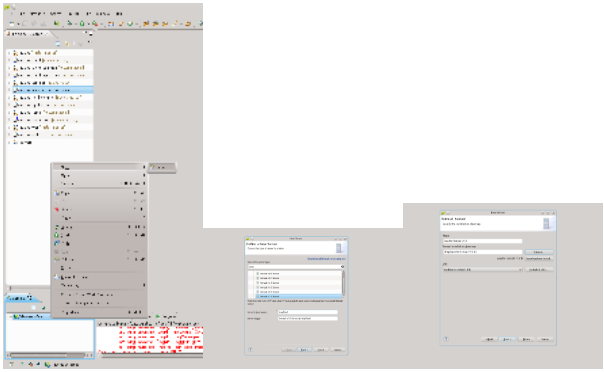
http://tomcat.apache.org/download-70.cgi

Es bedarf keiner Installation. Das heruntergeladene Archiv muss lediglich entpackt werden.

Apache Tomcat entpacken
`$ tar xzf apache-tomcat-7.0.42.tar.gz`

In SpringSource muss über *File->New* oder wie im Screen-Shot über die Serveransicht ein neuer Server angelegt werden.

Als Server Typ wird *Tomcat v7.0 Server* ausgewählt. Als *Tomcat installation directory* wird das Verzeichnis angegeben, in das der Server vorher entpackt wurde.



Zusätzliche Server VM Parameter (zu den Parameter, die von eclipse/sts automatisch erstellt werden)

```
-ea
-ms256M -mx768M
-Dfunctionblock.dev=true
-XX:PermSize=128M -XX:MaxPermSize=256M -XX:+UseThreadPriorities
```

Im Projekt *nuclos-war* muss nun unter src/main/webapp/META-INF eine Datei mit dem Namen *context.xml* angelegt werden.

Als Vorlage kann die Datei *context-netbeans.xml* kopiert werden, sie befindet sich im selben Verzeichnis.

Der Context definiert Umgebungsvariablen für den Nuclos Server.

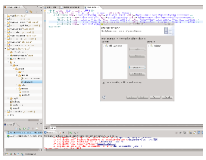
nuclos-war src/main/webapp/META-INF/context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/nuclos-war" reloadable="false">
       <Environment name="nuclos-conf-log4j" type="java.lang.String" value="file://SERVER_HOME/conf/log4j.
properties"/>
       <Environment name="nuclos-conf-jdbc" type="java.lang.String" value="file://SERVER_HOME/conf/jdbc.
properties"/>
       <Environment name="nuclos-conf-quartz" type="java.lang.String" value="file://SERVER_HOME/conf/quartz.
properties"/>
       <Environment name="nuclos-conf-server" type="java.lang.String" value="file://SERVER_HOME/conf/server.
properties"/>
       <Environment name="nuclos-conf-log4j-refresh" type="java.lang.Integer" value="60000"/>
</Context>
```

SERVER_HOME muss durch das Nuclos Umgebungsverzeichnis ersetzt werden Das lässt sich am einfachsten erzeugen, indem ein fertiger Nuclos Installer ausgeführt wird.

In einem letzten Schritt wird das Projekt *nuclos-war* auf dem neu erstellten Server publiziert. Hierzu wird über das Kontextmenü des Servers in der Serveransicht der Punkt *Add And Remove...* ausgewählt.

In dem Auswahlfenster wird die Zuweisung getätigt



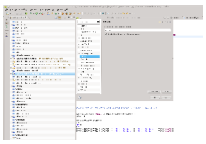
Jetzt kann der Server über Run/Debug gestartet werden.

LTW (Load Time Weaving) Spring Tomcat Instrumentation einrichten (Optional, verbessert Performance der Kompilierung)

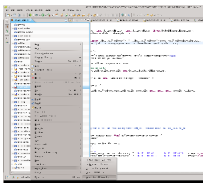
In Eclipse werden die Projekte standardmäßig mit AspectJ Support eingerichtet. Das führt zu einem CTW (Compile Time Weaving) welches das Übersetzen der Quellen spürbar verlangsamt.

Eine Alternative ist LTW (Load Time Weaving) zu verwenden.

Hierzu muss zunächst bei sämtlichen Projekten das Maven Profil *no-ctw* eingetragen werden.



Danach wird bei allen Projekten die *AspectJ Nature* entfernt.



Nun sollte das Projekt neu gebaut werden (siehe Troubleshooting Kapitel)

Damit der Tomcat Server das LTW durchführen kann muss die context.xml wie folgt angepasst werden:

LTW (Load-Time-Weaving)

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/nuclos-war" reloadable="false">
         <Loader loaderClass="org.springframework.instrument.classloading.tomcat.
TomcatInstrumentableClassLoader"/>
                <!--
                        ... siehe Beispiel weiter oben
                -->
</Context>
```

Damit die Klasse *TomcatInstrumentableClassLoader* gefunden werden kann muss die entsprechende Spring Instrumentation Tomcat JAR Datei in den *lib/* Ordner der verwendeten Tomcat Installation hinzugefügt werden.

Die benötigte JAR Datei kann aus dem lokalen Maven Repository kopiert werden z.B. aus

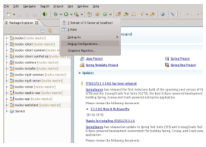
~/.m2/repository/org/springframework/spring-instrument-tomcat/3.2.3.RELEASE.jar

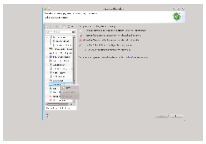
Existieren mehrere Versionen entscheidet man sich einfach für die Neueste.

Nuclos Client einrichten

Um den Nuclos Client zu starten wird eine *Debug Configuration* benötigt.

Über *Run->Debug Configurations...* gelangt man in die Übersicht, in der eine neue *Java Application* Debug Configuration angelegt werden kann.

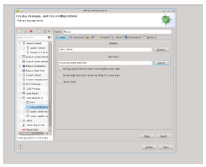




Reiter Main

Als Projekt wird *nuclos-client* ausgewählt.

Als Main class wird *org.nuclos.client.main.M*ain eingetragen (oder über *Search...* gesucht)



Reiter Arguments

In *VM arguments* wird folgendes eingetragen:

VM arguments

```
-ea  -XX:+HeapDumpOnOutOfMemoryError
-Xms256m -Xmx512m
-Dcom.sun.management.jmxremote="true"
-Dserver=http://localhost:8080/nuclos-war
-Dwebclient=http://localhost:4200/#
-Dsun.java2d.xrender=false
-Dfunctionblock.dev=true
```

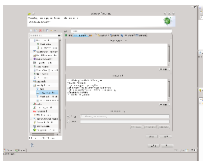
Erläuterung:

- **Xms, Xmx** minimale/maximale Heap Size
- **url.jms, url.remoting** die Adresse des Nuclos Servers. Der Port 8080 bezieht sich auf dem Port des Apache Tomcat servers. nuclos-war ist der Instanzname der auch beim Publizieren zu sehen ist.
- **functionblock.dev** Entwicklungsmodus an (true)/Entwicklungsmodus aus(false). Im Entwicklungsmodus ist es z.B. möglich die Layouts der Systementitäten zu sehen.
- **webclient** Adresse des laufenden Webclients. Wird u.a. vom Menüpunkt "Fenster > Einstellungen verwalten" verwendet.

für Linux noch zusätzlich

> Wird dieses Argument NICHT gesetzt kann es beim Debuggen zu System Freezes kommen sobald ein Breakpoint im Clientcode angesprungen wird.

```
-Dsun.awt.disablegrab=true
```



Troubleshooting

SpringSource stürzt während des Übersetzungsvorgangs häufiger mit OutOfMemory Fehlern ab

Die Heap Size von SpringSource ist standardmäßig relativ klein eingestellt. Das reicht für die Übersetzung der Nuclos Quellen oft nicht aus.

Heap Size und weitere Parameter lassen sich in der Datei STS.ini anpassen. Die Datei befindet sich im Verzeichnis von STS.

Beispiel:

STS.ini

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20120522-1813.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.200.v20120913-144807
-product
org.springsource.sts.ide
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.6
-Xms256m
-Xmx2048m
-XX:MaxPermSize=512m
```

Fehler beim Übersetzen, Abhängigkeit kann nicht gefunden werden, Klasse XYZ implementiert Methode nicht...

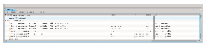
Diese Fehler treten häufig auf, wenn z.B. die aktuell verwendete Branch gewechselt wird.

Auch wenn diese Fehler oft sehr individuell sind, gibt es zwei Dinge die man tun kann um STS dazu zu bringen das Projekt neu einzulesen und evtl. fehlende Abhängikeiten aufzulösen.

Zunächst gibt es die Möglichkeit über Project->Clean die übersetzten Quellen wegzuschmeißen und neu übersetzen zu lassen. Ähnliche Funktionen bietet der Apache Tomcat Server.

Bei fehlerhaften Abhängigkeiten kann über das Kontextmenü eines Projektes unter Maven -> Update Project die Projektkonfiguration neu eingelesen werden.

STS zeigt Abhängikeitsprobleme oft als "Maven Problems" an (siehe Screen-Shot)



Tomcat Server startet, bricht aber nach einer gewissen Zeit ab

Der Server definiert standardmäßig Timeouts für Start und Stop Vorgänge. Wird Nuclos das erste mal gestartet kann es länger dauern, da aufgrund des Auto-Setups Tabellen angelegt werden müssen.

Deshalb sollte man das Timeout gleich zu Anfang auf einen höheren Wert setzen z.b. 300 sec. Bricht der Server den Startvorgang aufgrund des Timouts vorzeitig ab, muss die Datenbank manuell gelöscht werden

und die Nuclosinstallation erneut durchgeführt werden.

Ein Doppelklick auf den Server öffnet das Fenster in dem das Timeout und andere Einstellungen wie z.B. Port und Runtime Settings gemacht werden können.