Bootware Benutzerhandbuch

Inhaltsverzeichnis

1 Installation....3
1.1 Benötigte Dateien....3
1.2 Installation des modifizierten BPEL UI Eclipse Plugins....3
1.3 Installation des Bootware Eclipse Plugins....3
1.4 Installation des Bootware Repository....3
2 Konfiguration....4
2.1 Konfiguration des Bootware Repository....4
2.2 Konfiguration der Local und Remote Bootware....4
2.3 Konfiguration des Bootware Eclipse Plugins....4
3 Benutzung....6
3.1 Vorbereitung....6
3.2 Starten....6
3.3 Stoppen....7

1 Installation

Hinweis: Wurde die Installation einmal wie beschrieben ausgeführt kann Eclipse inklusive der installierten Bootware weitergegeben werden. In diesem Fall ist keine Installation mehr nötig.

1.1 Benötigte Dateien

Für die Installation der Bootware werden die drei in Abbildung 1 gezeigten Teile benötigt:

Abbildung 1: Zur Installation benötigte Dateien.

1. Der Ordner *bootware*: Dieser Ordner enthält das Bootware Eclipse Plugin sowie die Local und Remote Bootware.
2. Der Ordner *repository*: Dieser Ordner enthält das Bootware Repository sowie Mappings und Plugins.
3. Das jar-Archiv *org.eclipse.bpel.ui-0.8.0-SNAPSHOT.jar*: Dieses Archiv enthält das modifizierte BPEL UI Eclipse Plugin.

1.2 Installation des modifizierten BPEL UI Eclipse Plugins

Das modifizierte BPEL UI Eclipse Plugin wird installiert, indem das Plugin Archiv *org.eclipse.bpel.ui-0.8.0-SNAPSHOT.jar* in den Unterordner *plugins* der Eclipse Installation kopiert wird und das bereits vorhandene Plugin ersetzt wird.

1.3 Installation des Bootware Eclipse Plugins

Das Bootware Eclipse Plugin wird installiert, indem der Ordner *bootware* in den Unterordner *plugins* der Eclipse Installation kopiert wird.

1.4 Installation des Bootware Repository

Der Ordner *repository* sollte auf einen Rechner kopiert werden von wo aus das Repository später für die Bootware erreichbar sein soll. Zu bedenken ist, dass dieser Rechner über das Internet erreichbar sein muss (ohne VPN etc.), da die remote Bootware, die im Prinzip in jeder beliebigen Cloud laufen kann, auch auf das Repository zugreifen muss.

2 Konfiguration

Hinweis: Wie bei der Installation kann auch die Konfiguration einmal ausgeführt werden und dann mit Eclipse weitergeben werden. In diesem Fall ist keine Konfiguration mehr nötig. (Das geht natürlich nur, wenn die konfigurierten Werte auch bei dem Nutzer, an den sie weitergegeben werden, Sinn machen)

2.1 Konfiguration des Bootware Repository

Das Bootware Repository bietet keine Konfigurationsmöglichkeiten. Es muss lediglich über den Port 8888 von außen erreichbar sein (auch für die Remote Bootware, d.h. das Repository kann nicht auf dem lokalen Rechner laufen). Dann muss es lediglich gestartet werden. Dazu muss das *jar* Archiv ausgeführt werden mit dem Befehl: *java -jar repository-1.0.0.jar*

2.2 Konfiguration der Local und Remote Bootware

Die Local und die Remote Bootware können beide jeweils durch eine *properties* Datei konfiguriert werden, die sich im jeweiligen Ordner der Bootware befindet (*…/bootware/bin* für die Local Bootware, *…/bootware/bin/local/remote* für die Remote Bootware). Beide *properties* Dateien konfigurieren im Moment zwei Werte, *eventPlugins* und *repositoryURL.*

EventPlugins konfiguriert die Event Plugins die die jeweilige Bootware laden soll. Hier werden die Namen der Event Plugins aufgelistet. Als Trennzeichen dient ein Semikolon. Beispiel:

eventPlugins = consoleLogger-1.0.0.jar;fileLogger-1.0.0.jar;zeromq-subscriber-1.0.0.jar

RepositoryURL konfiguriert die URL des Bootware Repository. Beispiel:

repositoryURL = http://54.77.201.71:8888/repository

2.3 Konfiguration des Bootware Eclipse Plugins

Im Ordner des Bootware Eclipse Plugins (*…/bootware*) befinden sich zwei XML Dateien, die vom Bootware Eclipse Plugin beim Aufruf der Bootware an diese weiter gegeben werden.

Die Datei *defaultConfiguration.xml* kann Standardkonfigurationen für diverse Bootware Plugins enthalten. Sie wird beim starten der Local Bootware von dieser geladen und später auch an die Remote Bootware weiter gegeben. Hier sollten Konfigurationsparameter eingetragen werden, die sich nicht von Aufruf zu Aufruf unterscheiden, wie z.B. Logindaten von Cloud Providern. Im Beispiel in Abbildung 2 werden die secret und access keys, sowie die Region für AWS gesetzt, da diese für alle Aufrufe der Bootware gleich sind. Zu bedenken ist, dass Werte, die in der Standardkonfiguration gesetzt wurden, durch Werte in deploy Aufrufen und in Mappings überschieben werden. Welche Konfigurationswerte genau nötig sind ist abhängig von den Plugins. Das AWS EC2 Resource Plugin erwartet z.B. einen Eintrag *aws-ec2* mit den Untereinträgen *secretKey, accessKey, region,* etc.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configurationListWrapper>
  <configurationList>
    <entry>
      <key>aws-ec2</key>
      <value>
        <configuration>
          <entry>
            <key>secretKey</key>
            <value>aU7cD26+grgtluyqk0/YeMaRag5tmOZIUcvN+2Us</value>
          </entry>
          <entry>
            <key>accessKey</key>
            <value>AKIAJYIAG2HYG23MDSZQ</value>
          </entry>
          <entry>
            <key>region</key>
            <value>ec2.eu-west-1.amazonaws.com</value>
          </entry>
        </configuration>
      </value>
    </entry>
  </configurationList>
</configurationListWrapper>
```

Abbildung 2: Beispiel einer Standardkonfiguration in defaulConfiguration.xml.

Die Datei *context.xml* enthält den Context, der beim initialen Aufruf der lokal Bootware zur Provisionierung der Middleware an diese übergeben wird. Sie definiert mindestens drei Werte, wie in Abbildung 3 zu sehen ist. *Resource* beschreibt die Ressource auf der die Provisioning Engine zur Provisionierung der Middleware aufgesetzt werden soll. *Application* beschreibt die Provisioning Engine, die zur Provisionierung benutzt werden soll. *ServicePackageReference* beschreibt, wo das Service Package ,das provissioniert werden soll, gefunden werden kann. Im Beispiel möchten wir das Service Package *simtech.csar* mit der Provisioning Engine *opentosca* in der Cloud Umgebung *aws-ec2* provisionieren. Hier kann außerdem auch ein *configurationList* Element (wie in Abbildung 2 gezeigt) mitgeschickt werden, dass dann Konfigurationsparameter aus der Standardkonfiguration (und dem Request Context aus dem Bootware Repository) für diesen Aufruf überschreibt.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<context>
  <resource>aws-ec2</resource>
  <application>opentosca</application>
  <servicePackageReference>http://.../simtech.csar</servicePackageReference>
</context>
```

Abbildung 3: Beispiel eines User Context in context.xml.

Anmerkung: Je nach Cloud Anbieter kann es möglich sein, dass noch Geschäftsbedingungen angenommen werden müssen, bevor eine Aktion wie z.B. das starten einer VM möglich ist. In diesem Beispiel wird OpenTOSCA in einer Ubuntu VM installiert, für die separate AGBs akzeptiert werden müssen. Ist dies noch nicht geschehen wird der Bootstrappingprozess fehlschlagen und darauf hinweisen (zumindest bei AWS).

3 Benutzung

3.1 Vorbereitung

Nach dem Start von Eclipse sollte zuerst sichergestellt werden, das die Konsole sichtbar ist, da diese die Ereignisse der Bootware ausgibt. Sollte dies nicht der Fall sein, so kann die Konsole wie in Abbildung 4 gezeigt über *Window* → *Show View* → *Other...* → *General* → *Console* aktiviert werden.

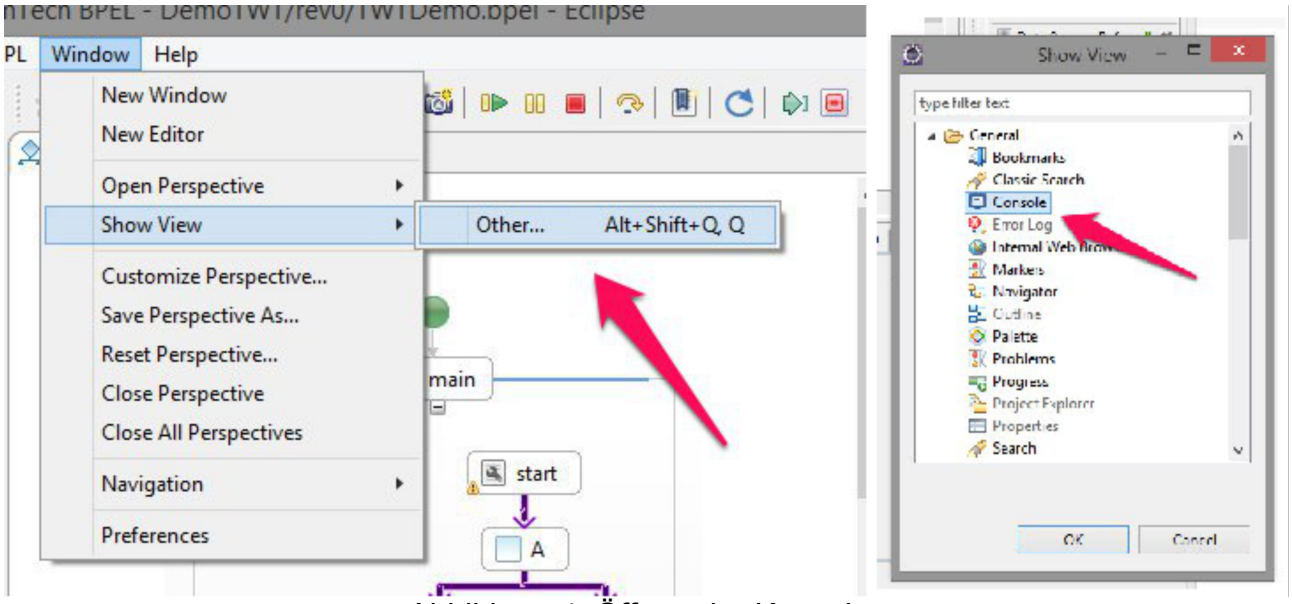

Abbildung 4: Öffnen der Konsole.

3.2 Starten

Die Bootware wird automatisch ausgeführt wenn eine neues Prozessmodel durch drücken auf den Start Knopf, wie in Abbildung 5 angedeutet, deployed wird.

Abbildung 5: Starten der Bootware.

Jetzt sollte in der Bootware Konsole der Fortschritt des Bootstrapping-Prozesses sichtbar sein (Abbildung 6). Der gesamte Bootstrapping-Prozesses wird in der Regel mehrere Minuten in Anspruch nehmen. Sobald der Bootstrapping-Prozesses abgeschlossen ist und die Middleware deployed wurde wird eine Instanz des deployeten Prozessmodelles wie gewohnt automatisch gestartet. Jetzt können ebenfalls weitere Prozessmodelle und -instanzen durch drücken des Start Knopfes auf der Middleware ausgeführt werden.

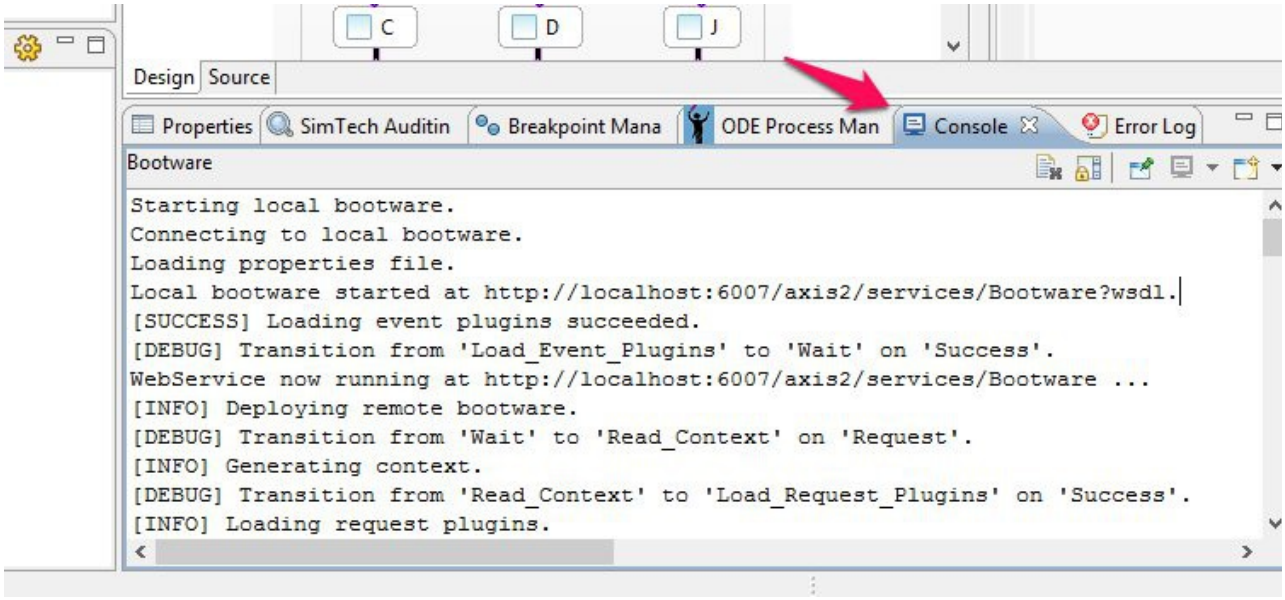

Abbildung 6: Ereignisse der Bootware in der Konsole

3.3 Stoppen

Das Bootware Eclipse Plugin kommuniziert mit der Workflow Engine und erfährt so, wie viele Prozessmodelle gerade aktiv sind. Sobald keine Prozessmodelle mehr aktive sind (d.h. der Benutzer hat alle Prozessmodeller undeployed, z.B. über den ODE Process Manager), öffnet sich ein Dialogfenster, wie in Abbildung 7 gezeigt, das fragt, ob der Shutdown-Prozess der Bootware eingeleitete werden soll. Bricht der Benutzer hier ab bleiben alle Komponenten aktiv und weitere Prozessmodelle können deployed werden. Bestätigt der Benutzer positiv, wird der Shutdown-Prozess gestartet und nach mehren Minuten sollten alle Resourcen die durch die Bootware deployed wurden (inklusive der remote Bootware) wieder undeployed worden sein. Der Ablauf des Shutdown-Prozesses ist ebenfalls in der Bootware Konsole sichtbar.

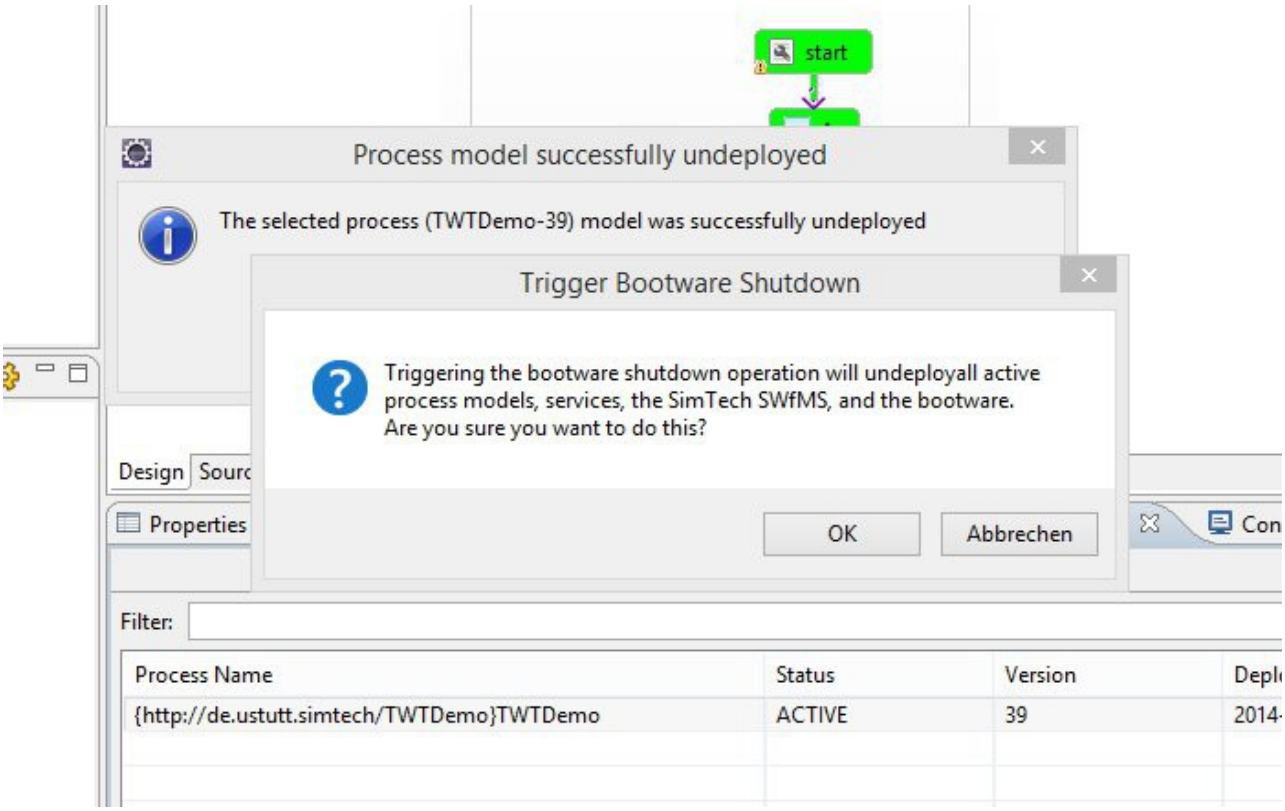

Abbildung 7: Automatisch durch Shutdown Trigger ausgelöster Dialog.

Der Shutdown-Prozess kann auch manuell ausgelöst werden, wie in Abbildung 8 gezeigt, durch den Menüeintrag *SimTech → Trigger Bootware Shutdown*. Der Shutdown-Prozess der hier bei Bestätigung des Dialogs ausgeführt wird ist identisch zum automatisch ausgelösten Prozess. In diesem Fall könnten allerdings noch Prozessmodelle auf der Workflow Middleware aktiv sein.

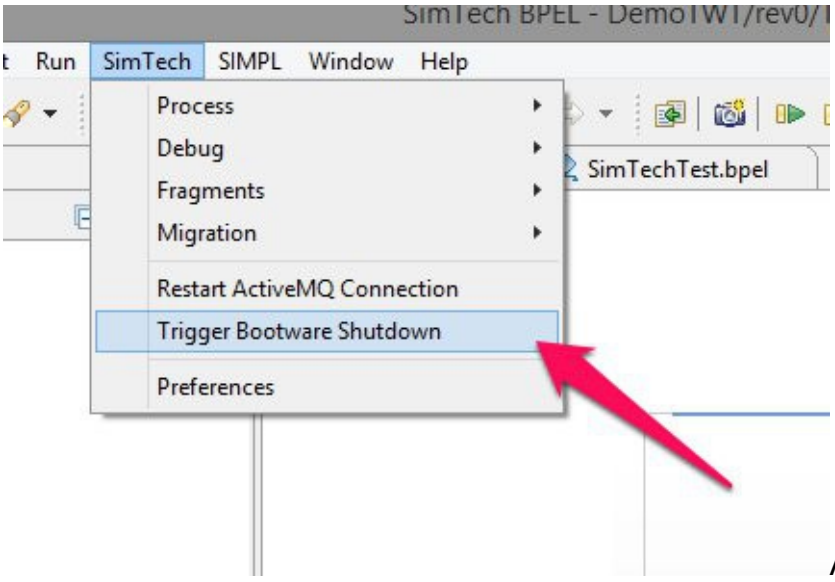

Abbildung 8: Menüeintrag für manuelles Herunterfahren der Bootware.