Entwicklungsumgebung

Definition

Falls Sie mit den umfangreichen Konfigurationsmöglichkeiten von Nuclos nicht alle Anforderungen abbilden können, haben Sie die Möglichkeit, Nuclos um eigene Java-Implementierungen zu erweitern. Dieser Artikel erklärt kurz, welche Schritte für die Einrichtung der Entwicklungsumgebung notwendig sind. Bitte wenden Sie sich bei Fragen oder auftretenden Schwierigkeiten an das Nuclos Forum unter http://www.nuclos.de/forum.

Voraussetzungen

Sie können den Client und Server um zusätzliche Implementierungen in Form von *.jar-Dateien erweitern. Diese *.jar-Dateien müssen bestimmte Voraussetzungen erfüllen, die später noch genauer erläutert werden. Prinzipiell können Sie den Source Code und die benötigten Konfigurationsdateien mit einem einfachen Texteditor erzeugen und mit Werkzeugen aus dem Java Development Kit weiter verarbeiten, um schließlich die gewünschten *.jar-Dateien zu erhalten. Wir empfehlen Ihnen allerdings die Verwendung eines Build-Management-Tools sowie eine integrierten Entwicklungsumgebung (IDE).

Da Nuclos seit Version 3.1. Apache Maven nutzt, empfehlen wir Ihnen ebenfalls die Verwendung dieses Build-Management-Werkzeugs. Besorgen Sie sich eine aktuelle Version von http://maven.apache.org/download.html und folgen Sie der dort skizzierten Installationsanleitung.

Eine integrierte Entwicklungsumgebung können Sie frei wählen. Als Voraussetzung sollte Ihre IDE allerdings Unterstützung für Maven-Projekte bieten.

Erstellen der Extension-Projekte

Erzeugen Sie mit folgendem Befehl die benötigten Extension-Projekte:

```
mvn archetype:generate -DarchetypeCatalog=http://maven.novabit.de/content/groups/publicsnapshots/
```

1. Früher befanden sich die Archetypen unter http://maven.novabit.de/content/groups/public/. Wir benutzen aber seit einiger Zeit nur noch SNAPSHOT-Archetypen.
2. Die Erstellung eines neuen Maven Projektes aus einem Archetype gelingt auch mit Eclipse. Die URL von oben muss mit 'Add Remote Catalog...' zu den Maven Archetypen hinzugefügt werden. Bitte auch Checkbox 'Include snapshot archetypes' anwählen.

Wählen Sie dabei das Archetype "nuclos-extension". Ab Nuclos Version 3.3.x benötigen Sie die Version 2.0.0-SNAPSHOT (oder höher).

Geben Sie anschließend die groupId (z.B.: com.mycompany), artifactId (z.B. für eine CRM-Anwendung: crm), die Versionsnummer und den Basis-Packagenamen ein. Die Verzeichnisstruktur sollte anschließend (abhängig von der artifactId) wie folgt aussehen:

```
+ workspace
|- + ${artifactId}
   |- + ${artifactId}-common
      |- + src
         |- + main
            |- java
            |- resources
         |- test
      |- pom.xml
   |- + ${artifactId}-client
      |- + src
         |- + main
            |- java
            |- + resources
               |- + META-INF
                  |- nuclos
               |- nuclos-app.properties
         |- test
      |- pom.xml
   |- + ${artifactId}-server
      |- + src
         |- + main
            |- java
            |- + resources
               |- + META-INF
                  |- nuclos
         |- test
      |- pom.xml

   |- + ${artifactId}-war
      |- + src
         |- + main
            |- + webapp
               |- + META-INF
                  |- MANIFEST.MF
               |- + WEB-INF
                  |- web.xml
      |- pom.xml
 |- pom.xml
```

Im Projekt "${artifactId}-server" bzw. "${artifactId}-client" befinden sich Erweiterungen für den Server bzw. Client. Das Projekt "${artifactId}-common" enthält Erweiterungen für Client und Server, also beispielsweise Funktionen, die sowohl im Server als auch im Client benötigt werden, oder zum Beispiel Schnittstellen und Klassen, über die eine Client-Server-Kommunikation stattfindet.

Als letzten Schritt tragen Sie bitte im Project Object Model (pom.xml) der übergeordneten Projekts die benötigte Nuclos-Versionssnummer ein (Eigenschaft "nuclos.version").

Schlüssel zur Codesignatur erstellen und konfigurieren

> Dieser Abschnitt ist veraltet. Es wird empfohlen, die (bash) Skripte aus der `test-extension` zu verwenden (s.u.). Diese benötigen jedoch eine Linux Umgebung.

Die POMs der einzelnen Module erben alle Einstellungen des Parent-POMs. Im Parent-POM ist ein zusätzliches Plugin (maven-jarsigner-plugin, http://maven.apache.org/plugins/maven-jarsigner-plugin/) konfiguriert, das benötigt wird, damit Ihre Erweiterungen auch per Webstart gestartet werden können. Die Konfiguration des Plugins benötigt vier weitere Einstellungen:

- keystore.file: Pfad zur Keystore-Datei
- keystore.alias: Alias des Schlüssels im Keystore
- keystore.storepass: Keystore-Passwort
- keystore.keypass: Schlüssel-Passwort

Um einen Keystore zu erzeugen, können Sie das Programm keytool verwenden, das im JDK enthalten ist (http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html).

Beipiel-Befehl zur Erzeugung eines Keystores:

```
keytool -genkeypair -dname "cn=Firstname Lastname, ou=Unit, o=Novabit, c=DE"
      -alias extension -keypass qwertz -keystore C:\workspace\mykeystore
      -storepass asdfg -validity 180
```

Um die Variablen zu füllen, können Sie die entsprechenden Properties im POM definieren. Sie können die Varibalen aber auch durch die konkreten Werte ersetzen oder auch durch andere Variablen-Namen ersetzen, z.B. wenn Sie die Einstellungen in Ihren lokalen Maven-Einstellungen definieren möchten.

Migration Extension Projekte

Auf Bitbucket finden Sie eine `test-extension`, die verdeutlichen soll, wie eine Extension Projekt am besten aufgebaut sein sollte. Mit `bin/prepare_deploy.sh` und `bin/prepare_deploy_with_nuclos_sources.sh` finden sich hier auch 2 (bash) Skripte, die vereinfachen, die gebaute Extension (und die von ihr benutzten JAR Abhängigkeiten) zu signieren (so wie das für Java Web Start notwendig ist).

Wenn Sie ein Extension Projekt migrieren wollen, können Sie das `test-extension` Projekt als Vorlage verwenden.

Tipps zum Maven Build von Extension Projekten

Sollten Sie nicht mit unserem Maven Archetypen starten wollen und auch die `test-extension` nicht als Vorlage verwenden wollen, hier einige Tipps:

- Ihr Extension-Projekt sollte `org.nuclos:nuclos:<Version>` als *parent pom* verwenden. Dadurch ist sichergestellt, dass ihre Extension Jar Abhängigkeiten in der Version verwendet, die von Nuclos vorgegeben wird. Ferner ist so das AspectJ Weaving (s.u.) automatisch konfiguriert.
- Web-App Initialisierung
  - Bis Nuclos 4.12:
    Das WAR Module Ihrer Extension sollte als Overlay `org.nuclos:nuclos-war:<Version>` verwenden. Dadurch müssen Sie nicht aufpassen, dass das von Ihnen verwendete `web.xml` alle Einstellungen der `web.xml` aus nuclos-war der entsprechenden Version enthält. Nachteil ist allerdings, dass in diesem Fall `org.nuclos:nuclos:<Version>` selbst bauen müssen, da wir dieses Artefakt nicht auf unserem Repository Server zur Verfügung stellen.
  - Ab Nuclos 4.13:
    Durch Upgrade auf die Servlet API v3 kann die Initialisierung jetzt programmatisch durch Implementierung eines org.springframework.web.WebApplicationInitializer erfolgen. Die Verwendung einer web.xml ist jedoch weiterhin möglich.
- Verwenden Sie die beiden Skripte aus der test-extension um Ihre JARs (und Ihre JAR Abhängigkeiten) zu signieren. Allerdings sind diese Skripte für Linux gedacht.
- Benutzen Sie das maven-jar-plugin, um ein Manifest mit den Manifest Attributen zu erzeugen, die ab Java 7u25 für Java Web Start nötig sind. Details finden sich unter Neuere Java 7 Versionen Abschnitt 'Was ist bei Nuclos Extensions zu beachten?'.

SNAPSHOTS

Möchte man seine Extension basiert auf einer SNAPSHOT Version entwicklen, so muss in der pom.xml, in der zuvor auch die Nuclos Versions Nummer eingetragen wurde, dies noch freigeschaltet werden:

```
<snapshots>
        <enabled>true</enabled>
</snapshots>
```

AspectJ Weaving

Nuclos ab Version 3.3.x verwendet Spring AspectJ Weaving. Wenn Sie z.B. Nuclos Controller überschreiben, müssen Sie Spring AspectJ Weaving auch in ihren Erweiterungsprojekten verwenden. Erstellen Sie neue Erweiterungen wird empfohlen, diese mit Maven und dem oben beschriebenen Maven Archetype zu erstellen. Führen Sie jedoch einen *Upgrade* alter Erweiterungsprojekte (Nuclos 3.2.x oder kleiner) auf Nuclos 3.3.x (oder höher) durch, müssen Sie in Ihre bisherigen Build Dateien (pom.xml) das Spring AspectJ Weaving *manuell* hinzufügen. Sie können sich hierbei an dem Maven Archetypen orientieren.

Integration von Erweiterungen

- nuclos-app.properties: Factory-Klassen für Collect-Controller
- <extension-server>/src/main/resources/META-INF/nuclos/*-beans.xml: Spring-ApplicationContext Server
- <extension-server>/src/main/resources/META-INF/nuclos/*-remoting.xml: Konfiguration für Remote-Export über HTTP
- <extension-client>/src/main/resources/META-INF/nuclos/*-beans.xml: Spring-ApplicationContext Client / Konfiguration für Remote-Import über HTTP)

Build und Deployment

Um Ihre Erweiterungen zu kompilieren und für eine Installation in Ihrer Nuclos-Instanz vorzubereiten rufen Sie die folgenden Maven-Goals im Verzeichnis des Parent-POMs auf:

```
mvn clean package
```

Erstellen Sie einen temporären Ordner mit dem Namen "extensions" und kopieren Sie die erzeugten Artefakte aus den target-Ordnern der einzelnen Projekte in die folgende Verzeichnisstruktur:

```
+ extensions
|- + common
   |- ${artifactId}-common-1.0.0.jar
   |- + native
      |- native_lib_wrapped_in_jar.jar
|- + client
   |- ${artifactId}-client-1.0.0.jar
   |- + themes
      |- ${artifactId}-theme-1.0.0.jar
|- + server
   |- ${artifactId}-server-1.0.0.jar
```

Der native Ordner wird nur benötigt, wenn ihre Extension JNI benutzt. Zur Zeit fehlt im JNI Fall die Möglichkeit, mehrere Betriebssysteme zu unterstützen.

Kopieren Sie anschließend den extensions-Ordner in das Verzeichnis Ihrer Nuclos-Installation und führen Sie den Installer erneut auf der Installation aus.