Extension Regeln

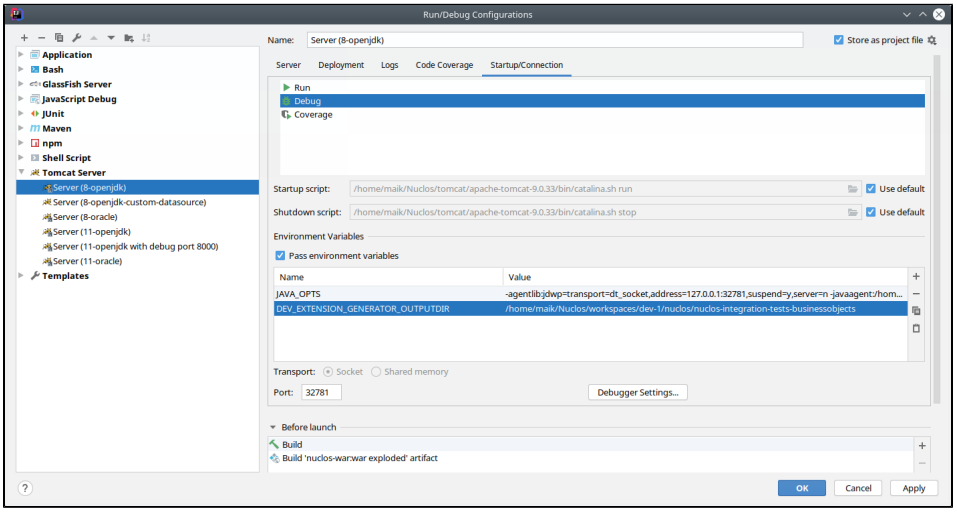
- Generierung der Businessobjekt Klassen & Co. in eine Extension
- Ablage der Regeln
- Verwendung der Regeln
- Regel Kompilierung
- Auslieferung einer Regel Extension
- Codegenerator deaktivieren
- Weitere Anmerkungen zum ClassLoading

Ab 4.41 unterstützt Nuclos die Entwicklung und Ausführung von Regeln aus Extensions (JAR Bibliothek).

Generierung der Businessobjekt Klassen & Co. in eine Extension

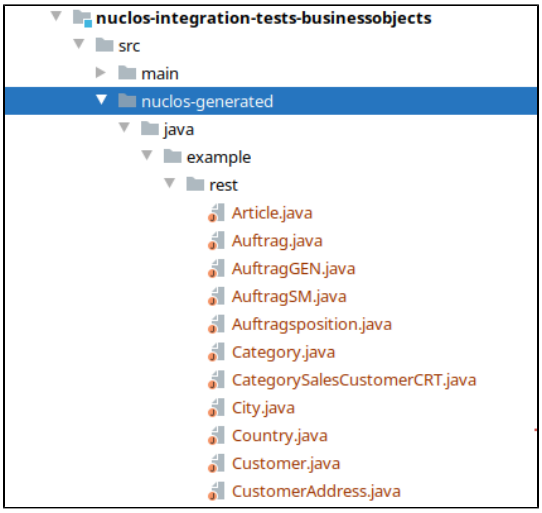
Einstiegspunkt in dieses Thema ist die Generierung der für die Regeln essentiell benötigten Klassen. Hierfür muss ein Environment Parameter **DEV_EXTENSION_GENERATOR_OUTPUTDIR** beim Starten des Servers mitgegeben werden. Inhalt muss der Pfad zum Maven Modul sein, in dem die von Nuclos automatisch erzeugen Klassen abgelegt werden sollen.

Ein Beispiel:



Ist in diesem Verzeichnis noch keine Maven **pom.xml** vorhanden, wird eine Standardmäßige erzeugt. Diese kann nach Belieben angepasst werden, da Nuclos eine vorhandene Datei priorisiert und nicht mehr ändert.

Die generierten Sourcen werden unter **src/nuclos-generated/java** abgelegt, wie in diesem Beispiel zu sehen:



Ablage der Regeln

Die Regeln sollten im Normalfall unter **src/main/java** abgelegt werden. Auch dieses Verzeichnis wird von Nuclos ignoriert und nicht angefasst.

Maven Modul Name

Nuclos scannt beim Hochfahren des Servers nach Regeln. Jedoch nur in Bibliotheken deren Name nachfolgender Konvention entspricht (aus Performance-Gründen):

```
*-rule-*jar
*-rules-*jar
oder
*-server-*.jar
```

Das Maven Modul der Regeln muss entsprechend benannt sein!

Sie können auch Regel und Businessobjekt Klassen in unterschiedlichen Modulen ablegen, um so z.B. die Businessobjekte nur zum kompilieren hinzuzuziehen, und in einem produktiven System wieder auf den Codegenerator zu setzen. Das kann jedoch unter Umständen zu Einschränkungen führen, wenn z.B. in den Regel Klassen eigene Services angeboten werden, die außerhalb der Nuclos Rule-Engine angesprochen werden. ClassNotFound-Exceptions wären dann zu erwarten.

Vorhandene Regeln überführen

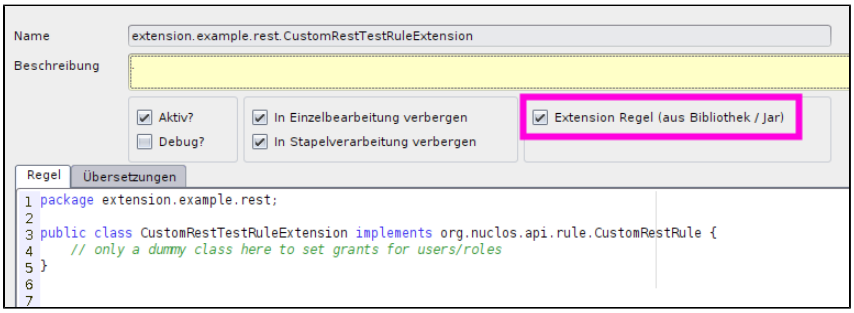
Sollen vorhandene Regeln überführt werden? Am schnellsten geht es, in dem die Regeln aus dem alten NUCLOS_HOME/data /codegenerator/src-rule Verzeichnis gesichert werden.

Es ist denkbar, dass neben der Nuclos API weitere Drittbibliotheken in den Regeln Verwendung finden, oder eine bereits vorhandene Server-Extension. Hierfür müssen in der pom.xml entsprechende Dependencies nachgetragen werden.
Beispiel siehe angehängte pom.xml

Verwendung der Regeln

Regeln aus einer Extension stehen zur Konfiguration ganz normal zur Verfügung. Können also einem BO, Job etc. zugewiesen werden.

Eine Besonderheit bildet die neue Checkbox **Extension Regel** im Regeleditor.



Regeln in der Datebank bzw. im Nuclet können neben dem eigentlichen Code auch einige weitere Eigenschaften aufweisen und zusätzlich berechtigt werden (CustomRestRule). Um einer Extension Regel diese Eigenschaften zuzuweisen oder sie zu berechtigen wird eine Dummy Regel ohne "Inhalt" aber mit dieser neuen Eigenschaft benötigt. Damit der Name der Klasse aus dem Code ermittelt werden kann ist lediglich ein valider Rumpf von Nöten.

Dummy Regel Rumpf Beispiel

```
package extension.example.rest;
public class CustomRestTestRuleExtension {}
```

Regel Kompilierung

Hier kann die volle Bandbreite des Maven Frameworks eingesetzt werden.

Auf der Kommandozeile

```
cd [DEV_EXTENSION_GENERATOR_OUTPUTDIR]
mvn clean package
```

Die Jar Bibliothek wird im **target** Verzeichnis abgelegt (Sofern nicht vom Standard abgewichen wird).

Bei Bedarf kann auf viele vom System bereitgestellte Funktion wie AspectJ zurückgegriffen werden, wie im Beispiel der Nuclos Intergrationstest eingestellt: https://bitbucket.org/nuclos/nuclos/src/master/nuclos-integration-tests-rules/pom.xml

Auslieferung einer Regel Extension

Allgemein gibt es 2 Möglichkeiten eine Regel Extension in einem System zu installieren. Grundlegend, also wenn man sich im Standard Java Umfeld und der Nuclos API bewegt, sind beide Verfahren miteinander gleichzusetzen. Jedoch wird eine über einen Nuclos Installer (1. Möglichkeit) installierte Extension vom Spring Framework gleich gefunden, gescannt und geladen. Bei einer Nuclet Extension wird dies erst nachträglich beim Starten des Server durchgeführt, damit ergeben sich Unterschiede beim ClassLoading, welche Nuclos versucht auszugleichen (siehe auch "Weitere Anmerkungen zum ClassLoading" weiter unten).
Wir empfehlen bei einem Verfahren zu bleiben, bzw. das Gleiche beim testen wie auch produktiv einzusetzen!

1. Mit der Installation von Nuclos

Hierzu muss unter **NUCLOS_HOME/extensions/server** die Jar Bibliothek abgelegt und ein Installer erneut ausgeführt werden. Siehe auch Extensions

2. Mit dem Import eines Nuclets

Die Jar Bibliothek kann auch als Extension im Nuclet hinterlegt werden. Unter **Nuclet Management** das gewünschte Nuclet öffnen und im Reiter **Extensions** die Datei als **Server Extension** (Checkbox setzen nicht vergessen!) hochladen. Ein Neustart des Server wird gefordert. Im Anschluss ist diese Extension im exportierten Nuclet enthalten und wird auf einem anderen System mit dem Nuclet Import automatisch installiert (ebenfalls nach einem Neustart).

Codegenerator deaktivieren

Der Standard Codegenerator kann mit dem Parameter **CODEGENERATOR_ENABLED** (Wert: false) komplett deaktiviert werden. Damit zwingt man ein System nur noch Regeln aus Extensions zu verwenden. Ein eingestellter Environment Parameter DEV_EXTENSION_GENERATOR_OUTPUTDIR (siehe oben) ist hiervon nicht betroffen und bleibt für die Entwicklung aktiv.

Weitere Anmerkungen zum ClassLoading

Wie unter dem Punkt der Auslieferung bereits angedeutet sind bereits ein paar Unterschiede bekannt die hier beschrieben werden. Diese Liste darf gerne um weitere Erfahrungen erweitert werden 😉

Setup	Problem
Extension im Nuclet; In einer Regel Klasse soll mittels AspectJ (**@Configurable**) ein Autowiring **@Autowired** durchgeführt werden. Ein Beispiel wäre diese Klasse im Integrationstest Nuclet Java8RuleExtension.java, mit einer @Configurable Annotation; Integrationstest RulesTest würde fehlschlagen.	Die **@Component** Klasse wird bei Ausführen der Regel nicht gefunden, da AspectJ mit dem Standard Application ClassLoader arbeitet . Es wird nicht der RuleClassLoader von Nuclos verwendet, bzw. dessen Context der das Objekt kennen würde. Mögliche Lösungen: 1. Extension mittels Installer installieren, womit auch die @Component Klasse im Standard Application ClassLoader landet. 2. Wie im Beispiel zu sehen, auf AspectJ an dieser Stelle verzichten und somit auf "Standard Spring" setzen.
...	