Transformation AML-Model zu Maven/Java/RMI-Code

Erster Entwurf für Besprechung am 23.06.2017.

Inhalt

[Maven pom.xml 2](#_Toc486269554)

[Package-Struktur 3](#_Toc486269555)

[Klassen, Interfaces und Methoden 4](#_Toc486269556)

[AML-Elemente ohne Auswirkung auf den generierten Code 12](#_Toc486269557)

[Festlegungen und bekannte Einschränkungen 13](#_Toc486269558)

[betreffend das AML-Model 13](#_Toc486269559)

[Festlegungen 13](#_Toc486269560)

[Einschränkungen 13](#_Toc486269561)

# Maven pom.xml

|  |  |
| --- | --- |
| * Für Attribute Service.artifactId und Service.groupId des IE mit Role ArchitectureRoleClassLib/Service: groupId und artifactId setzen * Für jedes IE mit Role ArchitectureRoleClassLib/Component: neues Profil anlegen mit Name des IE | |
|  | *<?***xml version="1.0"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>  <**groupId**>com.jwa.pushlistener</**groupId**>  <**artifactId**>pushlistener</**artifactId**>  <**profiles**>  <**profile**>  <**id**>Node1</**id**>  <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.codehaus.mojo</**groupId**>  <**artifactId**>exec-maven-plugin</**artifactId**>  <**version**>${exec-maven-plugin.version}</**version**>  <**executions**>  <**execution**>  <**goals**>  <**goal**>java</**goal**>  </**goals**>  </**execution**>  </**executions**>  <**configuration**>  <**mainClass**>com.jwa.pushlistener.node2.Main</**mainClass**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**>  </**profile**>  </**profiles**> </**project**> |

# Package-Struktur

|  |  |
| --- | --- |
| * Main-Package laut groupId in pom.xml anlegen und darunter:   + für jedes IE mit Role ArchitectureRoleClassLib/Component: Package mit dessen Namen anlegen   + wenn mindestens ein IE mit Role ArchitectureRoleClassLib/Component ein IE-Kind mit Role CommunicationRoleClassLib/Ports besitzt:     - communication anlegen und darunter:       * messagemodel       * wenn mindestens ein IE mit Role ArchitectureRoleClassLib/Component ein IE-Kind mit Role CommunicationRoleClassLib/Ports besitzt, welches die Role CommunicationRoleClassLib/CommunicationStyle/Synchronous/RMI besitzt:         + rmi und darunter:   components (Code darin ist unabhängig der AML-Modellierung)  interfaces   * + - * + für jedes IE mit Role ArchitectureRoleClassLib/Component in dessen Package: communication.rmi.impl anlegen | |
|  |  |

# Klassen, Interfaces und Methoden

|  |  |
| --- | --- |
| * Für jedes IE mit Role ArchitectureRoleClassLib/Component jeweils mit Kind-IE mit Role CommunicationRoleClassLib/Ports jeweils für alle IE mit Role CommunicationRoleClassLib/Port: wenn Role DomainModelRoleClassLib/MessageModel/X vorhanden aber noch nicht als Klasse in communication.messagemodel angelegt ist: X.java anlegen mit Default-Konstruktor: extends MessageModel | |
|  |  |

|  |  |
| --- | --- |
| * wenn mindestens ein IE mit Role ArchitectureRoleClassLib/Component ein IE-Kind mit Role CommunicationRoleClassLib/Ports besitzt, welches die Role CommunicationRoleClassLib/CommunicationStyle/Synchronous/RMI besitzt:   + RMIClient.java anlegen mit bereits komplett vorgefertigtem Code   + RMIRemoteInterface.java anlegen mit bereits komplett vorgefertigtem Code   + RMIServer.java anlegen mit bereits komplett vorgefertigtem Code | |
|  |  |

|  |  |
| --- | --- |
| * für alle IE mit Role ArchitectureRoleClassLib/Component, die ein IE-Kind mit Role CommunicationRoleClassLib/Ports besitzen, welches die Role CommunicationRoleClassLib/CommunicationStyle/Synchronous/RMI besitzt:   + für alle Kind-IEs mit Role CommunicationRoleClassLib/Port und EI mit Namen ConnectionPoint und RefBaseClassPath CommunicationInterfaceClassLib/Port/Receiver:     - wenn noch nicht vorhanden: RMIRemoteInterfaceNode<Name des IE>.java anlagen     - execute-Methodensignatur anlegen mit Parameter gleich Message-Model der Gegenstelle (ConnectionPoint hat IL dahin) und Return-Type gleich eigenem zugewiesenen Message-Model | |
|  |  |

|  |  |
| --- | --- |
| * für alle IE mit Role ArchitectureRoleClassLib/Component in dessen Package:   + Main.java anlegen     - package Zeile anpassen     - portRegistry setzen wie im Attribute RMI.portRegistry im Kind-IE mit Role CommunicationRoleClassLib/Ports angegeben     - eventuell: alle Methodenaufrufe (= alle Ports im AML-Model mit ConnectionPoint gleich Sender) eintragen | |
|  |  |

|  |  |
| --- | --- |
| * für alle IE mit Role ArchitectureRoleClassLib/Component, die ein IE-Kind mit Role CommunicationRoleClassLib/Ports besitzen, welches die Role CommunicationRoleClassLib/CommunicationStyle/Synchronous/RMI besitzt, in dessen Sub-Package communication.rmi.impl:   + RMIRemoteInterfaceImpl.java anlegen     - package Zeile anpassen     - implements Zeile anpassen     - Methoden des referenzierten Interface anlegen und //TODO-Flag (eventuell auch println) setzen   + RMIClientImpl.java anlegen     - package Zeile anpassen     - für alle unique Gegenstellen (pro Port mit Rolle Sender den IL zum ConnectionPoint der Gegenstelle):       * private Variablen anlegen       * close()-Methode anpassen       * getClient<Gegenstelle>()-Methode anlegen         + hostname und portRegistry setzen wie in den Attributen der CommunicationPorts der Gegenstelle festgelegt     - für alle Ports mit Rolle Sender:       * exexcute(...)-Methoden anlegen         + Signatur   Parameter: eigenes Message-Model  Return-Type: Message-Model der Gegenstelle   * + - * + Body   [return] getClient<Name der Gegenstelle>().execute(message) | |
|  |  |

# AML-Elemente ohne Auswirkung auf den generierten Code

Folgende RoleClassLibs haben derzeit keine Auswirkung auf den generierten Code:

* EnvironmentRoleClassLib
* BusinessRoleClassLib
* SourcecodeRepositoryRoleClassLib

# Festlegungen und bekannte Einschränkungen

## betreffend das AML-Model

### Festlegungen

* alle IEs mit Role ArchitectureRoleClassLib/Component
  + müssen unique Names haben (da diese im Code verwendet werden)
  + dürfen nur ein Kind-IE mit Role CommunicationRoleClassLib/Ports haben
    - dessen Kind-IEs mit Role CommunicationRoleClassLib/Port
      * einen unique Name besitzen (nicht zwingend nötig, macht aber Sinn, um Verwirrungen vorzubeugen)
      * Message-Models sind optional
      * nur ein EI mit Namen ConnectionPoint haben dürfen
        + muss IL zu einem IE mit Role ArchitectureRoleClassLib/Component haben und nicht zu sich selbst
        + der RefBaseClassPath auf CommunicationInterfaceClassLib/Port/Sender oder CommunicationInterfaceClassLib/Port/Receiver gesetzt hat

Gegenstelle muss immer entsprechender Gegenpart sein (Sender – Receiver oder Receiver – Sender aber nicht Sender – Sender oder Receiver – Receiver)

### Einschränkungen

* Code kann derzeit nur generiert werden, wenn alle IEs mit Role ArchitectureRoleClassLib/Component eine Role ProgrammingLanguageRoleClassLib/Java zugewiesen haben
* Kommunikationsart (RMI etc.) wird derzeit für alle Ports festgelegt, individuell pro Port ist nicht möglich