CommunityMashup - Development - Core

This set of pages describes how to develop for the CommunityMashup core.

Attention: Currently, generating classes from the models only works in a Windows environment - When trying to do so in a Mac environment, errors in the models are reported and the generation fails (creates wrong code).

Setting up the development environment

We are using Eclipse as (Java) development environment.

You can download a pre-configured version of Eclipse from https://sociotech.atlassian.net/wiki/display/MASHUP/Development+Environment.

Alternatively, just install a clean version of the newest release of Eclipse and add the following components (in the given order) - tested with Eclipse Neon.1a Release (4.6.1):

- Modeling / *EMF* (All EMF components from Eclipse Modeling Project)
- EMF Ecore and MDT XSD / *ecore* (Ecore Tools SDK)
- OCL Classic SDK (Eclipse Modeling Project)
- Equinox Target Components
- e(fx)clipse
- javax.transaction: copy the jar files into the dropins directory of eclipse: http://www.java2s.com/Code/JarDownload/javax/javax.transaction_1.1.1.v201002111330.jar.zip - or newer
- org.jdom: copy jar files into the dropping directory of eclipse: version 1.0.0 - or newer

- ??? Mortbay Jetty Plugins: load jetty and jetty-util from http://mvnrepository.com/artifact/org.mortbay.jetty/ and put into dropins directory of eclipse
- ??? javax.mail: copy the jar files into the dropins directory of eclipse: http://repository.springsource.com/ivy/bundles/external/javax.mail/com.springsource.javax.mail/1.4.5/com.springsource.javax.mail-1.4.5.jar

After having installed Eclipse, you have to import the CommunityMashup core from the GitHub repository:

- **File -> Import -> Git -> Projects from Git -> URI**

Next screen(s)

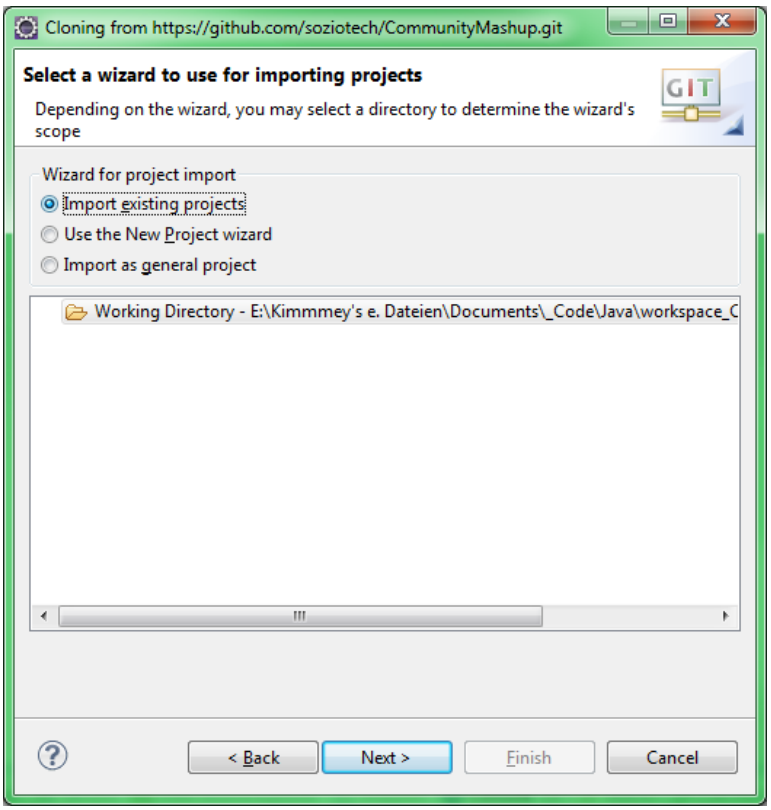
- URI: https://github.com/soziotech/CommunityMashup.git -> Next -> Next (select branch "master")

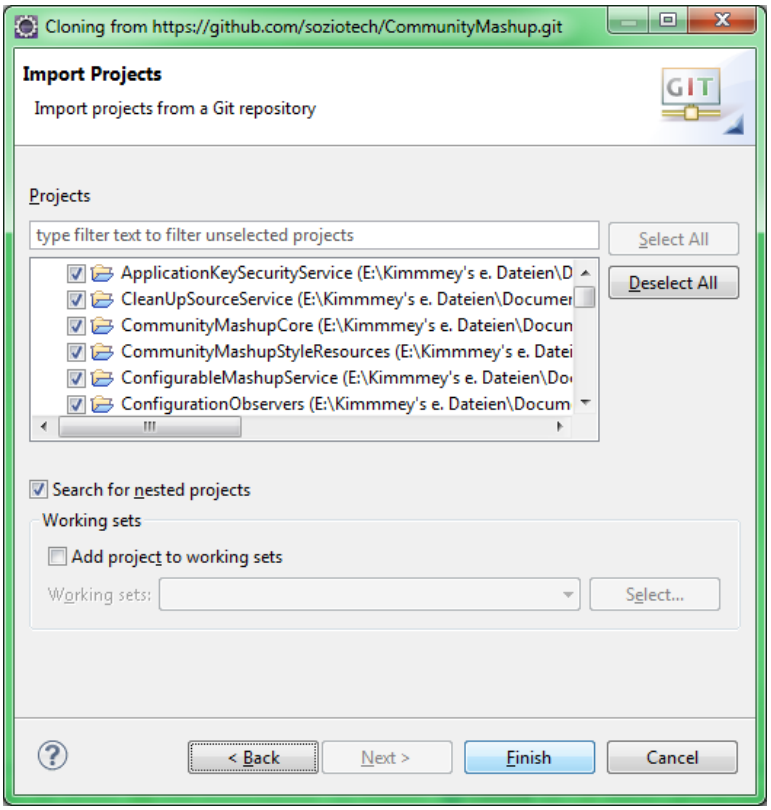
Next screen (Local Destination):

- Directory: Choose destination for local Git repository (this can be a sub directory of your Eclipse workspace - but does not have to be)

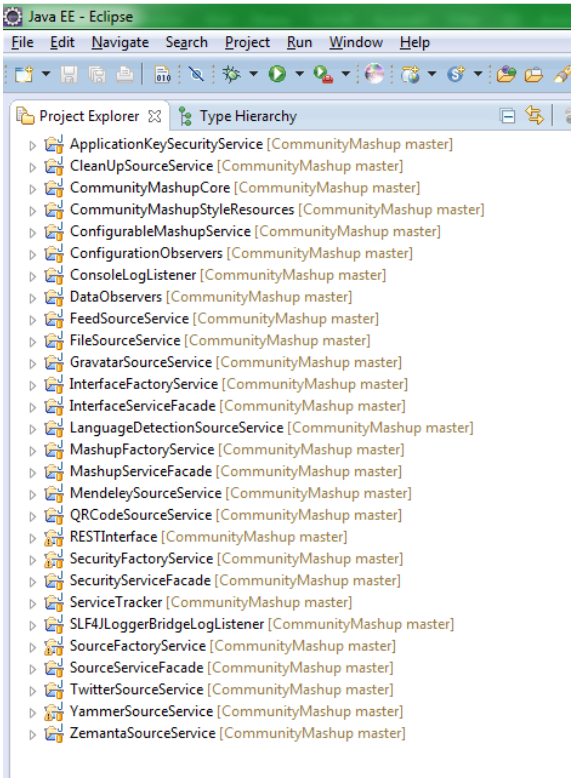
Next screen (Select a wizard)

- Import existing projects -> Next -> Finish

- 

- 

The result should look like this:



If there are problems compiling the sources, please check in the Eclipse Workspace Settings that the (Java) Compiler compliance level is set to "1.6" (or higher).

Kerntechnologien

Die technische Lösung des CommunityMashup baut im wesentlichen auf OSGi und EMF auf.

OSGi

Das CommunityMashup wird in einfache unabhängige Komponenten zerlegt, die sich einzeln konfigurieren, deployen, starten und stopen lassen. Diese Komponenten werden als OSGi Services realisiert.
Für einen Einstieg in die Entwicklung von OSGi Services mit Eclipse und Java bietet sich das OSGI Tutorial von Lars Vogel an.

EMF - Eclipse Modelling Framework

EMF stellt die Basis des modellbasierten Entwicklungsansatzes des CommunityMashup dar. Mit ihm werden das interne Datenmodell sowie Datenmodelle der einzelnen Quellen modelliert. Des weiteren dient es auch zur Meta-Modellierung von CommunityMashup Konfigurationen.
Als Einstiegspunkt kann das EMF Tutorial von Lars Vogel verwendet werden.

Meta-Modell, Konfigurationen und Deployment-Szenarios

Über das CommuntiyMashup Meta-Modell lassen sich unterschiedlichste einfache und komplexe Mashup-Konfigurationen beschreiben die sich wiederum in unterschiedlichen Deployment-Szenarien wiederfinden können.

Update model

For updating the data model, please follow this process:

1) The data model is stored in CommunityMashupCore/model/data.ecore

You may add new classes or add new attributes or functions to the existing classes as needed.

Then you have to save the changes.

2) Then change to CommunityMashupCore/model/mashup.genmodel (select the object)

In the editor you now should select (click on) Data

The do a right click and select "Generate Model Code"

This should generate the interfaces and implementations of the data classes in CommunityMashupCore/src

Now you can check in the edited model and the newly generated data classes.

Export OSGi Bundles

Nach einer Änderung des Datenmodells oder allgemein der Core-Klassen müssen folgende beiden OSGi-Bundles neu erzeugt und auf den Server hochgeladen werden:

- org.sociotech.communitymashup.core_0.0.1.jar
- org.sociotech.communitymashup.data.DataObservers_0.0.1.jar

Dazu in Eclipse: Export -> Plugin Development -> Deployable Plugins and Fragments

1. Rechte Mausklick auf CommunityMashupCore Project -> Export...
2. Plug-in Development -> Deployable plug-ins and fragments -> Next >
3. Available Plug-ins and Fragments -> Select All
4. Destination: Export-Verzeichnis über "Directory" angeben
5. Options: folgende Optionen auswählen
    1. Package plug-ins as individual chars
    2. Qualifier replacement
    3. Allow for binary cycles in target platform
    4. Use class files compiled in the workspace
6. Finish

Components

Content from Atlassian Wiki:

- Security
    - ApplicationKey
- Interfaces
    - REST
- Core
    - CommunityMashupCore
    - ConfigurableMashupService
    - InterfaceServiceFacade
    - MashupServiceFacade
    - SecurityServiceFacade
    - SourceServiceFacade
- Construction
    - Interface Factory
    - Mashup Factory
    - Security Factory
    - Source Factory
- Utils
    - Configuration Observers
    - Console Log Listener
    - Data Observers
    - Sentry Log Listener
    - Service Tracker
    - SLF4J Logger Bridge
    - Style Resources