CommunityMashup - Development - Sources

This set of pages describes how to develop (new) sources for the CommunityMashup.

- Setting up the development environment
- Checking out source code for additional sources
- Creating a new source
    - Create new project in Eclipse
    - Project structure
    - Share project
    - Include needed components
    - Create main class
    - Implement functionality
        - Initialization
    - Comments on using / creating Items
- Implementieren
    - Quellen Initialisierung
    - Activator implementieren
- Testing the sources
- To run the CommunityMashup Demo configuration, the following steps are required
- Packing updated sources for using them in a CommunityMashup installation

Setting up the development environment

We are using Eclipse as (Java) development environment.

You can download a pre-configured version of Eclipse with all components needed for CommunityMashup development 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 Folder)
- 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

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)

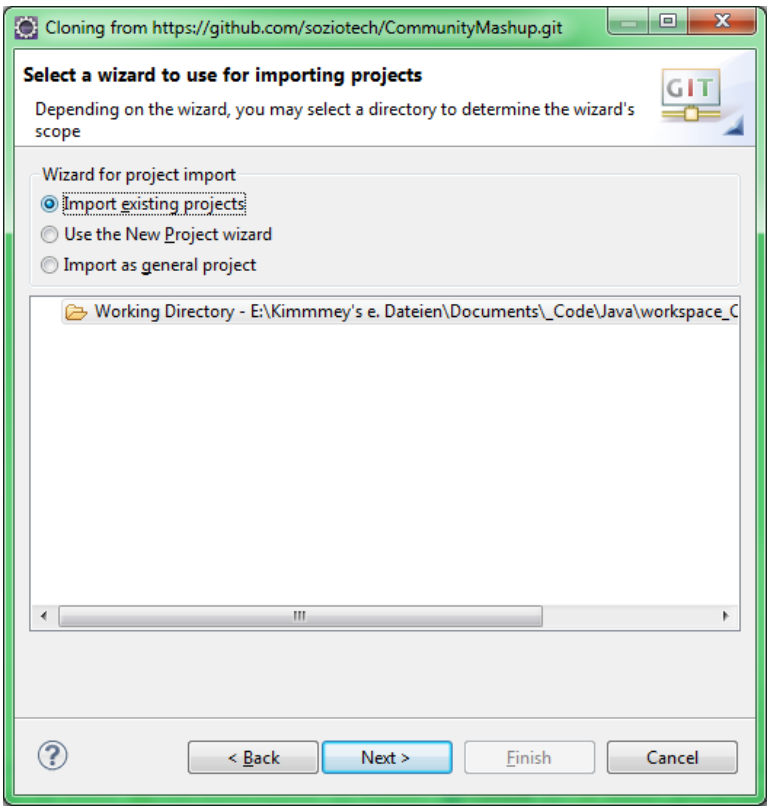
- URI: https://github.com/soziotech/CommunityMashup.git -> Next -> Next (select branch "master")

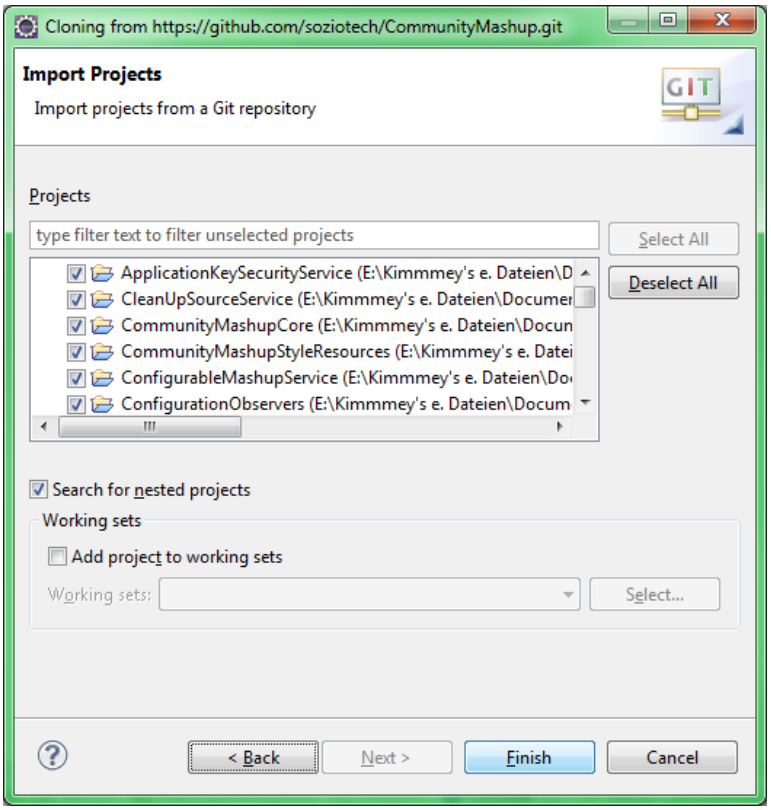
Next screen (Local Destination):

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

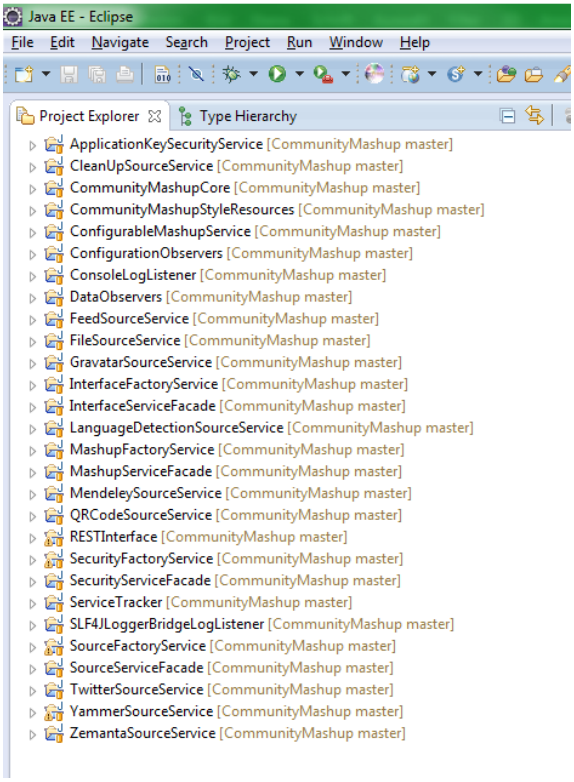
Next screen (Select a wizard)

- 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).

Checking out source code for additional sources

The CommunityMashup core already includes source code of the public sources -

To check out the source code of the other source projects use the Eclipse import function for the corresponding source code repository.

For the development projects at Bundeswehr University Munich for example:

- **File->Import->SVN->Checkout Projects from SVN**
- Browse projects in https://vcs.cscwlab.de/svn/communitymashup/trunk

Creating a new source

Create new project in Eclipse

Usually, you just have to right-click in the package explorer - and select **New -> Project**

In the next screen:

- Select the wizard for creating a "Plug-in Project"

In the next screen:

- Specify a project name - e.g. "IcalFeedSourceService" (the name should end with "SourceService")
- Check **"an OSGi framework" - "standard"** in the "Target Platform" settings

In the next screen:

- Specify the additional properties
- For sources from the core team this would be
    - **ID:** org.sociotech.communitymashup.source.**icalfeed (package name for source)**

    - **Version:** 0.0.1
    - **Name:** IcalFeedSourceService (name of class for source service)
    - **Provider:** org.sociotech.communitymashup
- Select "Generate an activator ..." in the "Options" settings - and add the full package/class name, e.g. "org.sociotech.communitymashup.source.icalfeed.IcalFeedBundleActivator"
- Check that "Enable API analysis" is NOT selected

Now finish the process (do NOT select any template for creating the project).

Project structure

The new project should have the following structure:

Share project

Now you should import the project to a source code management service - in our case SVN:

- right-click in the package explorer (when the new project is selected): Team -> Share Project

Next screen:

- Select "SVN"

Next screen:

- Select the repository location - in our case this usually is https://vcs.cscwlab.de/svn/communitymashup/trunk/

Next screen:

- Specify the project location
- Usually: Use Simple Mode - e.g. https://vcs.cscwlab.de/svn/communitymashup/trunk/FeedSourceService

In some SVN plugins you may now have to commit the initial import - in others this commit already did happen (and you had to specify a commit comment ...)

Include needed components

- Open the file **META-INF/MANIFEST.MF** in the project.
- Change to the tab "Dependencies"
- Add in **Required Plug-ins** the following projects from your workspace (you should have set up your workspace before ...):
    - org.sociotech.communitymashup.core
    - org.sociotech.communitymashup.source.SourceServiceFacade
- Add in **Imported Packages**
    - org.osgi.service.log
- Save the changes!

Create main class

Create a main class for the new source: right-click in your project: **New -> Class**

- **Source folder:** Your project/src
- **Package:** org.sociotech.communitymashup.source.ShortNameForSource (e.g. "icalfeed")
- **Name:** identical to project name (e.g. "IcalFeedSourceService")
- **Modifiers:** public
- **Superclass:** org.sociotech.communitymashup.source.impl.SourceServiceFacadeImpl

Confirm with **Finish.**

Do not forget to add the newly created classes to the source code management system ...

Implement functionality

Initialization

- Open the new class.
- Implement the method **initialize()** to do everything that is needed to initialize your source:

- TBD add source

- **Important:** Call super.initialize() to continue the initialization after you have finished your stuff.

Comments on using / creating Items

In the fill phase of the sources the main task is to create Item objects and add them to the DataSet (and to other Item objects).

For this, please use the following process:

- > DataFactory factory = DataFactory.eINSTANCE
- Now create an empty Item object using the appropriate create method of the factory
  > person = factory.createPerson()
- Now use the add-methods of the new object to add any information that might be needed to specify with which objects to merge the new object - e.g. name, alternativeNames, Identifier objects
  > person.setName(xxx)
  > person.addAlternativeName(xxx)
- Then add the object to the DataSet using the sources add method:
  > person = source.add(person);
  - in an optional second parameter of the add function you can specify a value for the Items ident attribute
  - when the object is merged, the function may return the reference to a different object!
- Now add all other data (e.g. MetaTags)

Implementieren

Quellen Initialisierung

- Öffen Sie nun die zuvor angelegte Klasse
- Sie können nun die Methode **initialize()** wie folgt überschreiben und um zusätzliche Schritte, die speziell für Ihre Quelle notwendig sind, erweitern:
  .
- **Wichtig:** Rufen Sie nachdem ihre Initialisierung erfolgreich war immer super.initialize() auf.

Activator implementieren

Die Activator Klasse kontrolliert den Lebenszyklus jeder Quelle.

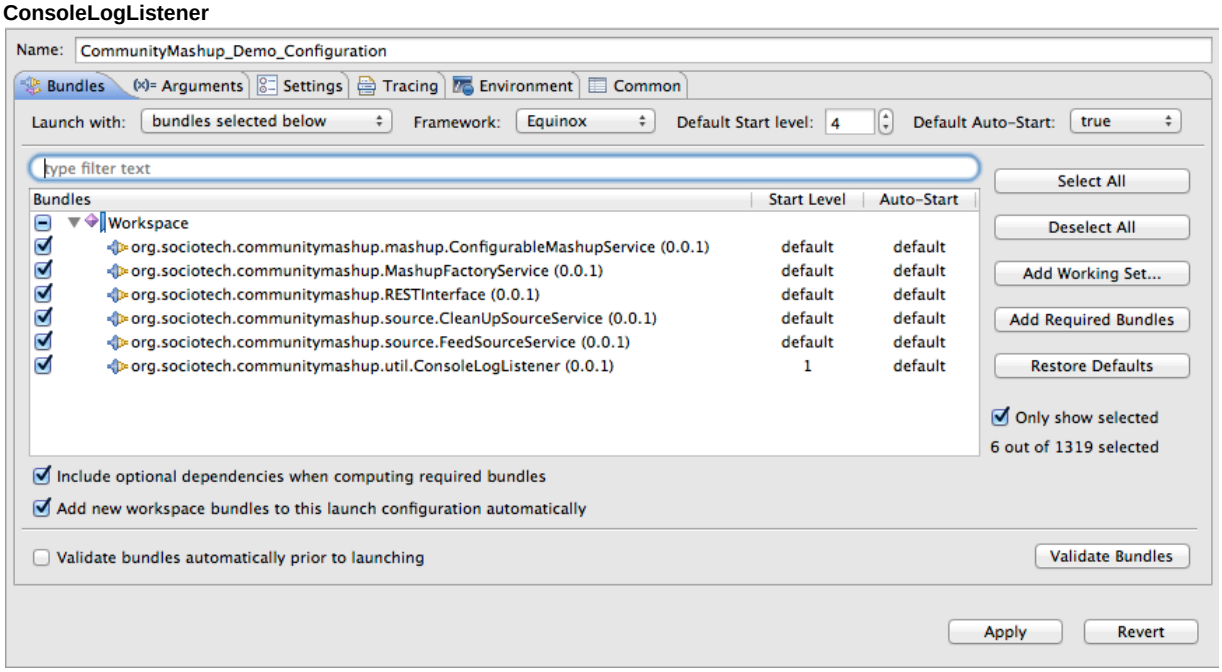
- Erweitern Sie die Activator Klasse in Ihrem Projekt und lassen Sie sie von **SourceServiceActivator** aus dem Paket **de.unibw.cscm.communitymashup.source.impl** erben. Dadurch muss das Interface **BundleActivator** nicht mehr implementiert werden.
- Überschreiben Sie die Methoden **start** und **stop** bei Bedarf um zusätzliche Schritte beim Starten ihrer Quelle durchzuführen.
- **Beispiel:** Sie könnten einen Thread starten, der regelmäßig einen externen Dienst auf Änderungen überprüft.
- **Wichtig:** Rufen Sie immer **super.start** mit dem übergebenen **bundleContext** und einer neuen Instanz ihrer zuvor anglegten **Hauptklass** auf.
- **Beachten Sie:** In diesem Schritt wird die Methode **initialize** ihrer Hauptklasse aufgerufen. Anschließend wird der Service der Registry bekannt gemacht und kann vom Mashup verwendet werden. Überlegen Sie sich also an welcher Stelle innerhalb der Methode **start** Sie diesen Schritt ausführen.
- Räumen Sie innerhalb der Methode **stop** wieder auf.
- **Beispiel:** Stopen Sie Threads die Sie innerhalb von **start** gestartet haben.
  .

Testing the sources

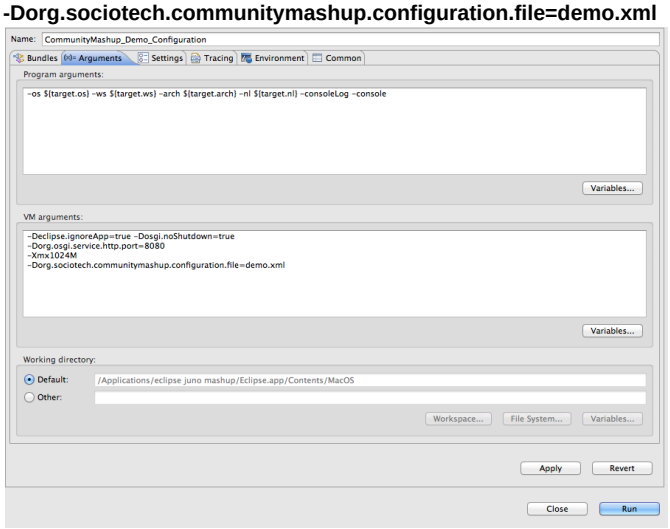
To run the CommunityMashup Demo configuration, the following steps are required

1. **Run** -> **Run Configurations**
2. **New** OSGI Framework Run Configuration
3. Deselect all Bundles (in **Workspace** as well as in **Target Plattform**)

4. Select the following bundles:
    1. **ConfigurableMashupService**
    2. **MashupFactoryService**
    3. **RESTInterface**
    4. **CleanUpSourceService**
    5. **FeedSourceService** - or another SourceService you want to test
    6. **ConsoleLogListener**



5. Set the Start Level of the ConsoleLogListener to **1**
6. Click on **Add Required Bundles**
7. Go to the **Arguments** tab and add the following to **VM Arguments**
    1. **-Dorg.osgi.service.http.port=8080**
    2. **-Xmx1024M**
    3. **-Dorg.sociotech.communitymashup.configuration.file=demo.xml**



8. Click on **Apply** to save the Run Configuration
9. Click on **Run** to execute the Run Configuration
10. Direct your browser to http://localhost:8080/mashup to see the result.

In Step 7 you link to the mashup configuration file. This has to be put in "MashupFactoryService/configuration".

Packing updated sources for using them in a CommunityMashup installation

1. Select the source project - right click: Export
2. Plug-in Development / Deployable plug-ins and fragments -> Next

3. Specify any directory
4. In Options: "Package plug-ins as individual JAR archives" should be selected
5. Finish

Then copy the generated JAR file to the OSGI packages directory ...