Development with Odysseus

NEW (29.01.2024): We will change some dependency handling! For development with eclipse you must install the **m2e PDE Integration** into the Eclipse IDE.

- *Help - Install New Software…*
- Use the m2e Update Site: https://download.eclipse.org/technology/m2e/releases/latest/
- Select *m2e PDE Integration*
- Finish the installation

After the installation it can be used in the *PDE Target Editor*.

06.02.2024: Due to a bug in Eclipse, you will need to use the latest eclipse version (2024-03): https://www.eclipse.org/downloads/packages/release/2024-03/

Nearly everything in Odysseus is designed to be replaced or extended. Here are the typical ones:

- Language extensions
  - Create a new language (not only for queries, could be a DSL for anything)
  - Create a new Odysseus Script Commands (#....)
  - Create a new Logical/PQL Operator
- Processing function extensions
  - Create new datatypes
  - Create new stream object types
    - Data Handler
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates
  - Create new aggregation functions
  - Create new operators
- Create new schedulers and scheduling strategies
- Create new meta data

- Some OSGi/Eclipse basics
  - OSGi Life Cycle
  - OSGi debugging
- Setting up
  - 1. Prerequisites
  - 2. Checkout Source Code
    - Option 1: Extending Odysseus with New Plugins
      - ODT
    - Option 2: Extending an Existing Odysseus Plugin/Extending Odysseus Core
  - Remark: There could be some updates in odysseus_dev that is not already reflected in each module. So it is always a good idea to update the odysseus_dev submodule to the newest version.
  - 3. Setup Eclipse
  - 4. Target Platform
  - 5. First Run - Available Products
  - 6. Additional Information
      - Features
    - Logging

Some OSGi/Eclipse basics

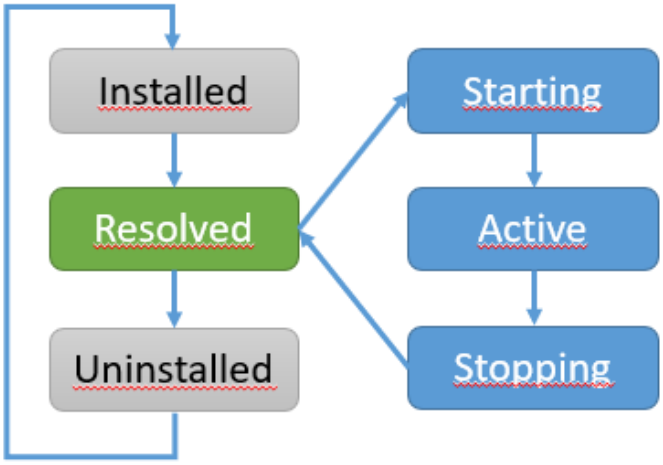
- Bundles
  - Aka plugin: An eclipse project
  - Each bundle has its own class loader
  - MANIFEST.MF: meta data for that bundle (name, version, imports, exports…)
  - In Odysseus: a module that encapsulates functions
- Fragment
  - A special bundle that will not exist allone but together with a host bundle
  - Same class loader as host bundle
  - Used to extend host bundle
  - We do not use this anymore, better approach is declarative services
- Declarative Services
  - OSGi way of dependency injection
  - Defined by so called components
  - Can provide functions by interfaces or use (bind/unbind) implementations by interfaces à examples later
- Feature: a feature is a collection of bundles
  - Define a set of bundles, that belong together and builds some functionality (e.g. each wrapper has its own feature)

    - An update site provides features
    - A bundle can be part of many features

- Update-site:
    - A collection of features that can be installed in Odysseus
    - On the same way, as in Eclipse („Install new software“)
    - Via command on ther server
    - Via Odysseus-Script
- Product:
    - A product is a runnable software (with an application)
    - Can be defined by bundles or features
    - We only use features to define products

OSGi Life Cycle

- Each bundle has a life cycle
- Installed: A bundle (with correct Metadata is installed)
- Resolved: All dependencies (MANIFEST) are found
- Uninstalled: removed from runtime
- Active: a bundle is activated
    - E.g. call of bundle activator
    - Remark: there is no need to start a bundle, if the bundle should only provide classes (as a library)
- Eclipse tries to resolve dependencies lazy, if this cannot be done, the bundle stays installed --> bundle cannot be used



OSGi debugging

TODO: Check commands with apache felix console

- When an application (product) gets started with –console (as in Odysseus always), there is a console available
- ss shows all currently available bundles and their current life cycle state
- Sometimes, there are problems because dependencies are missing (INSTALLED)
- Typical problem: The dependency defined in the MANIFEST.MF was not added to any feature
- diag <id> shows the missing dependencies
- Again: Resolved is no “problem” ;-)

```
0       ACTIVE      org.eclipse.osgi_3.10.102.v20160118-1700
                    Fragments=257
1       ACTIVE      org.eclipse.equinox.simpleconfigurator_1.1.100.v20150423-1455
2       RESOLVED    Commons.Math_1.0.0
3       RESOLVED    TestNG6.8_1.0.0
4       RESOLVED    com.fasterxml.jackson_1.0.0
5       RESOLVED    com.google.guava_21.0.0.v20170206-1425
6       ACTIVE      com.ibm.icu_54.1.1.v201501272100
7       RESOLVED    com.jaxb_2.3.0
8       RESOLVED    com.jayway.jsonpath_1.0.0
9       RESOLVED    com.jcraft.jsch_0.1.53.v201508180515
10      RESOLVED    com.rits.cloning_1.8.5
11      RESOLVED    com.sun.el_2.2.0.v201303151357
12      RESOLVED    de.uniol.inf.is.jfreechart_1.0.0
13      ACTIVE      de.uniol.inf.is.odysseus.aggregation_1.0.1
14      RESOLVED    de.uniol.inf.is.odysseus.bugreport_1.0.0
15      RESOLVED    de.uniol.inf.is.odysseus.client.common_1.0.0
16      ACTIVE      de.uniol.inf.is.odysseus.client.starter_1.0.0
17      STARTING    de.uniol.inf.is.odysseus.compiler_1.0.0
18      ACTIVE      de.uniol.inf.is.odysseus.console.executor_1.0.0
19      ACTIVE      de.uniol.inf.is.odysseus.core_1.0.0
20      ACTIVE      de.uniol.inf.is.odysseus.core.server_1.0.0
21      ACTIVE      de.uniol.inf.is.odysseus.core.server.console_1.0.0
22      ACTIVE      de.uniol.inf.is.odysseus.core.server.defaultdatadictionary_1.0.0
23      ACTIVE      de.uniol.inf.is.odysseus.datatype.interval_1.0.0
24      ACTIVE      de.uniol.inf.is.odysseus.interval_latency_1.0.0
25      ACTIVE      de.uniol.inf.is.odysseus.intervalapproach_1.0.0
26      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.datahandler_1.0.0
27      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.datatype_1.0.0
28      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.mep_1.0.0
29      ACTIVE      de.uniol.inf.is.odysseus.latency_1.0.0
30      RESOLVED    de.uniol.inf.is.odysseus.latency_relational_1.0.0
```

- ls: shows all currently installed (declarative) services
- Unsatisfied: Some dependencies cannot be found
- Use comp <id> to determine missing dependencies

```
All Components:
ID      State                   Component Name                  Located in bundle
1       Active          de.uniol.inf.is.odysseus.aggregation.AVG                        de.uniol.inf.is.odysseus.aggregation(bid=13)
2       Active          de.uniol.inf.is.odysseus.aggregation.Count                      de.uniol.inf.is.odysseus.aggregation(bid=13)
3       Active          de.uniol.inf.is.odysseus.aggregation.DistinctCount                      de.uniol.inf.is.odysseus.aggregation(bid=13)
4       Active          de.uniol.inf.is.odysseus.aggregation.DistinctNest                       de.uniol.inf.is.odysseus.aggregation(bid=13)
5       Active          de.uniol.inf.is.odysseus.aggregation.First                      de.uniol.inf.is.odysseus.aggregation(bid=13)
6       Active          de.uniol.inf.is.odysseus.aggregation.Last                       de.uniol.inf.is.odysseus.aggregation(bid=13)
7       Active          de.uniol.inf.is.odysseus.aggregation.Max                        de.uniol.inf.is.odysseus.aggregation(bid=13)
8       Active          de.uniol.inf.is.odysseus.aggregation.Min                        de.uniol.inf.is.odysseus.aggregation(bid=13)
9       Active          de.uniol.inf.is.odysseus.aggregation.Nest                       de.uniol.inf.is.odysseus.aggregation(bid=13)
10      Active          de.uniol.inf.is.odysseus.aggregation.AggregationFunctionRegistry                        de.uniol.inf.is.odysseus.aggregation(bid=13)
11      Active          de.uniol.inf.is.odysseus.aggregation.Sum                        de.uniol.inf.is.odysseus.aggregation(bid=13)
12      Active          de.uniol.inf.is.odysseus.aggregation.Topk                       de.uniol.inf.is.odysseus.aggregation(bid=13)
13      Active          de.uniol.inf.is.odysseus.aggregation.Trigger                    de.uniol.inf.is.odysseus.aggregation(bid=13)
14      Active          de.uniol.inf.is.odysseus.aggregation.Variance                   de.uniol.inf.is.odysseus.aggregation(bid=13)
15      Active          de.uniol.inf.is.odysseus.console.executor.impl.CommandProviderBinder                    de.uniol.inf.is.odysseus.console.executor(bid=18)
16      Registered              de.uniol.inf.is.odysseus.console.executor.ConsoleCommandExecutor                       de.uniol.inf.is.odysseus.console.executor(bid=18)
17      Active          de.uniol.inf.is.odysseus.core.datahandler.DataHandlerRegistry                   de.uniol.inf.is.odysseus.core(bid=19)
18      Unsatisfied             de.uniol.inf.is.odysseus.core.ObjectHandlerRegistry                     de.uniol.inf.is.odysseus.core(bid=19)
19      Active          de.uniol.inf.is.odysseus.core.ProtocolHandlerRegistry                   de.uniol.inf.is.odysseus.core(bid=19)
```

Setting up

This section describes how to set up Eclipse and Odysseus for development.

New:

- See also our video about this.
- **See also some slides**
- **See a more compact example here**

1. Prerequisites

You will need the following tools:

- Java 17 OpenJDK
- Eclipse *for RCP and RAP Developers* (**Important! Do not use a standard eclipse version!**)
- GIT client (you can use git from command line or with a tool of your choice)

2. Checkout Source Code

Remark: Under windows it is best to use a folder, that is not inside the user home path (because of long file names). A good option would be a base folder git somewhere on the root level.

There are different ways to create new functionality with Odysseus. The first way should be used if you want to add something new to Odysseus (e.g. a new wrapper) but do not want to change the common code base. This should be the preferred way of development with Odysseus:

Option 1: Extending Odysseus with New Plugins

Odysseus has a very large code base. To allow a lightweight start, we provide a template project in our Bitbucket server. You should clone or fork this project and use it to add new plugins to this template. The template can be found here: https://git.swl.informatik.uni-oldenburg.de/projects/ODY/repos/odysseusrepotemplate/browse . When using git you should use the following command to clone Odysseus. The template contains some standard products and the target platform as a submodule. The "--recurse-submodules" flag clones this submodule when cloning the repository. Alternatively, you can clone the submodule individually.

```
git clone --recurse-submodules https://git.swl.informatik.uni-oldenburg.de/scm/ody/odysseusrepotemplate.git
newRepo
cd newRepo
cd odysseus_dev
git checkout master
```

Remark: because you are not allowed to push updates to the template repository you should change your repository URL to your git location and rename your folder to your new repository name.

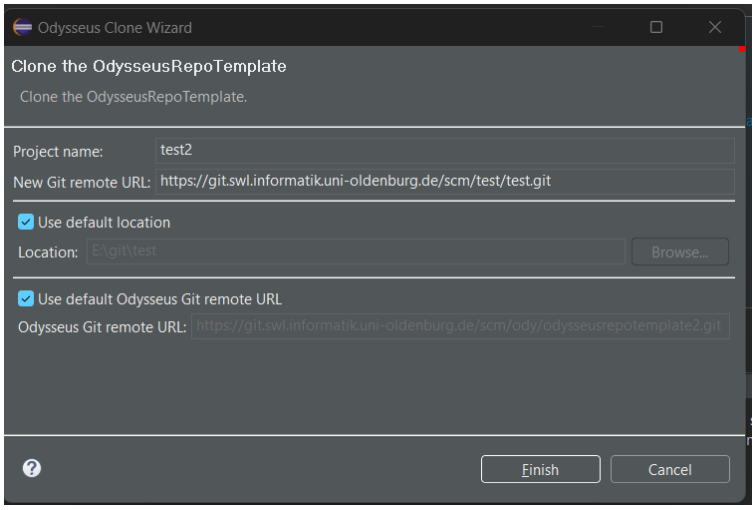
```
// in your newRepo Folder!
git remote set-url origin <new url>
```

ODT

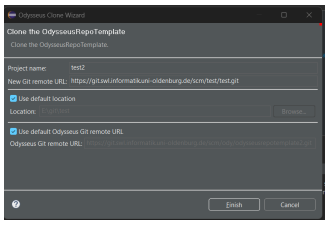
There is also a first approach for an eclipse plugin that could make processing easier. For this, install the Odysseus Development Tools from the following update site

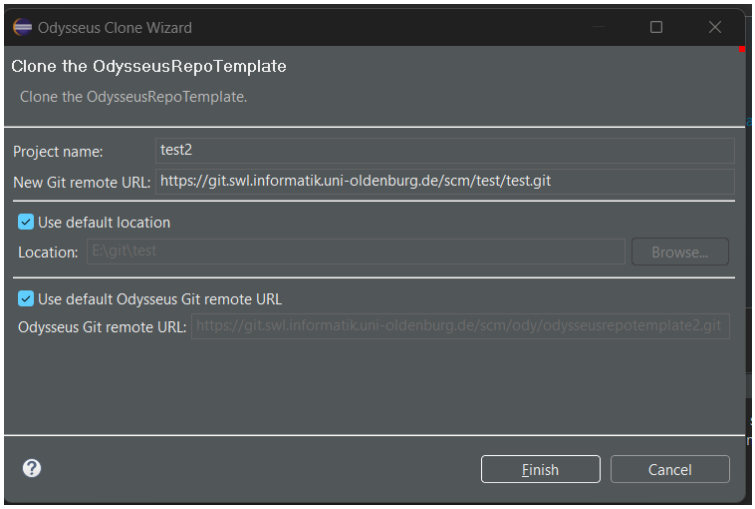
```
https://odysseus.informatik.uni-oldenburg.de/download/odt/
```

and uncheck "Group items by category".



After installation and restart there is a new menu option Odysseus where you can create an initial project:





Option 2: Extending an Existing Odysseus Plugin/Extending Odysseus Core

You want to extend an existing plugin from Odysseus or extend the core system? Then, instead of cloning the empty template from Option 1, clone the repository of the plugin you want to develop on. If the plugin is hosted in our Bitbucket, you can probably find it in one of the projects in this list: https://git.swl.informatik.uni-oldenburg.de/projects . Look for example at the Odysseus, the Odysseus Incubation and Odysseus Wrapper projects. When you have found the repository you want to use, check it out and don't forget to clone the submodule to get the target platform definition and the standard products:

```
git clone --recurse-submodules <your repo URL>
cd <repo>
cd odysseus_dev
git checkout master
```

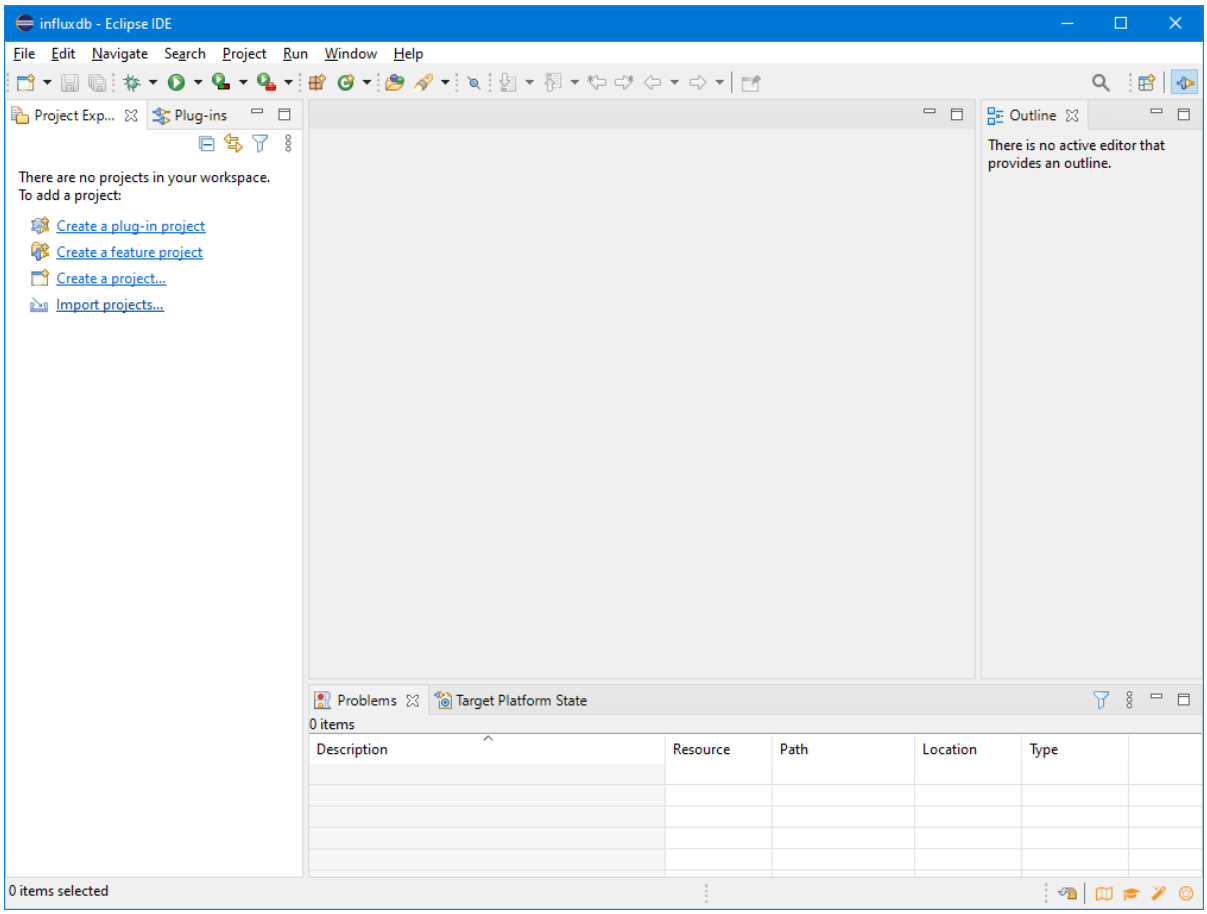
Remark: There could be some updates in odysseus_dev that is not already reflected in each module. So it is **always** a good idea to update the odysseus_dev submodule to the newest version.

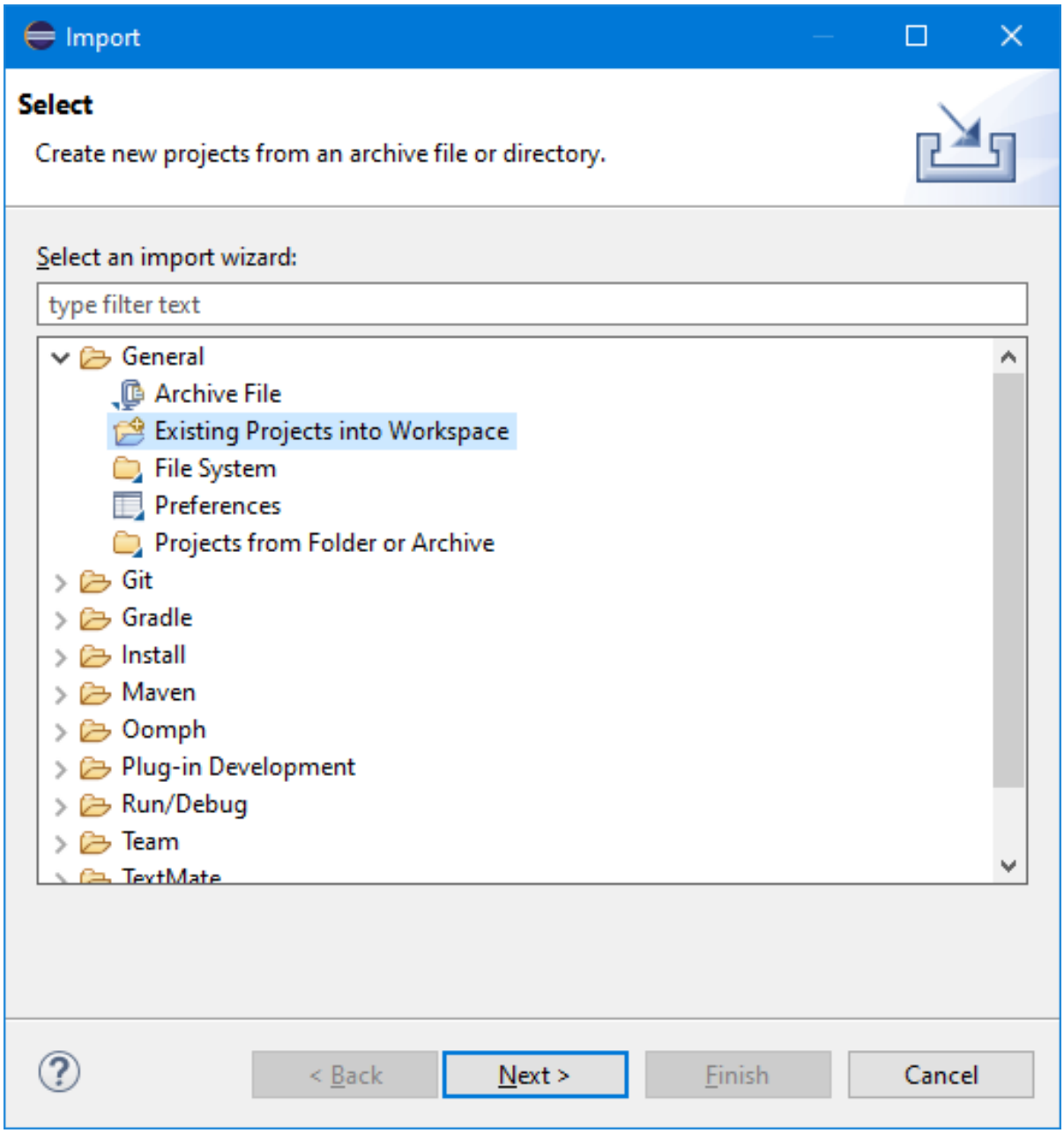
If you missed the --recurse-submodules, you can also use:

```
git clone <your repo URL>
cd <repo>
git submodule init
git submodule update
cd odysseus_dev
git checkout master
cd ..
```

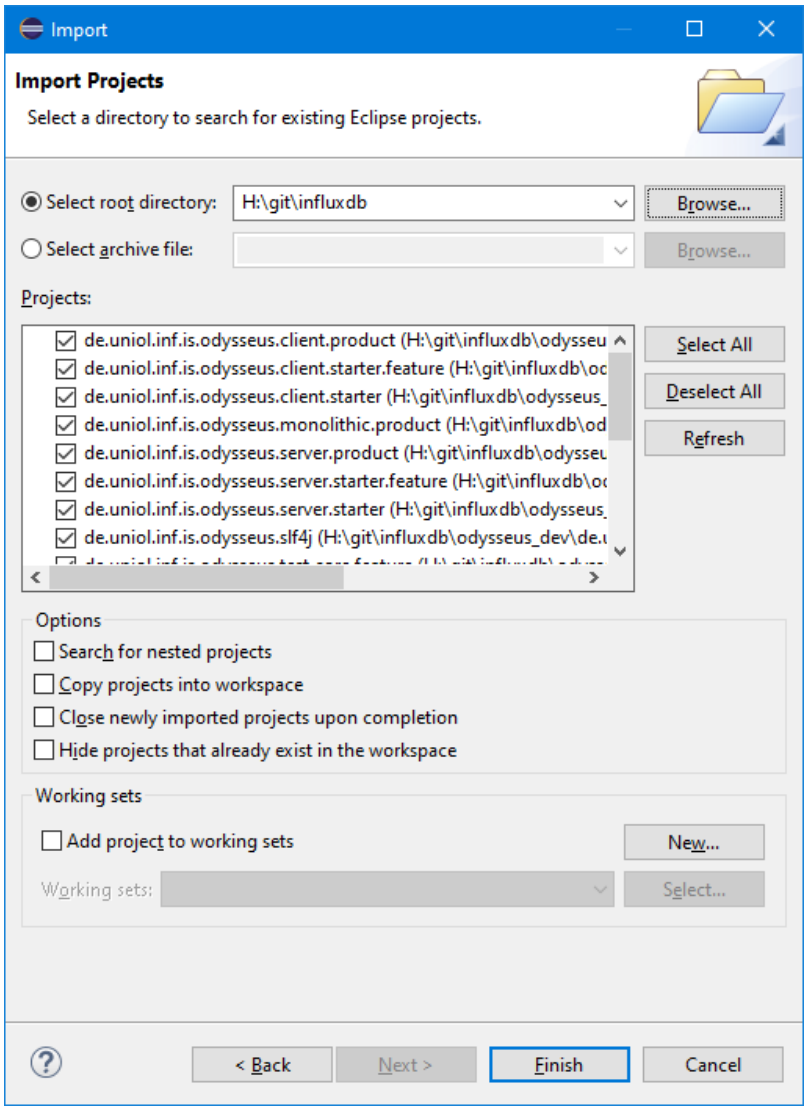
3. Setup Eclipse

After Eclipse started, you have to import all bundles (these are the parts of Odysseus and are equal to an Eclipse project). Use "File -> Import -> Existing Projects into Workspace" to import them into Eclipse as follows:


influxdb - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Exp...
Plug-ins
There are no projects in your workspace.
To add a project:
Create a plug-in project
Create a feature project
Create a project...
Import projects...
Outline
There is no active editor that provides an outline.
Problems
Target Platform State
0 items
Description
Resource
Path
Location
Type
0 items selected

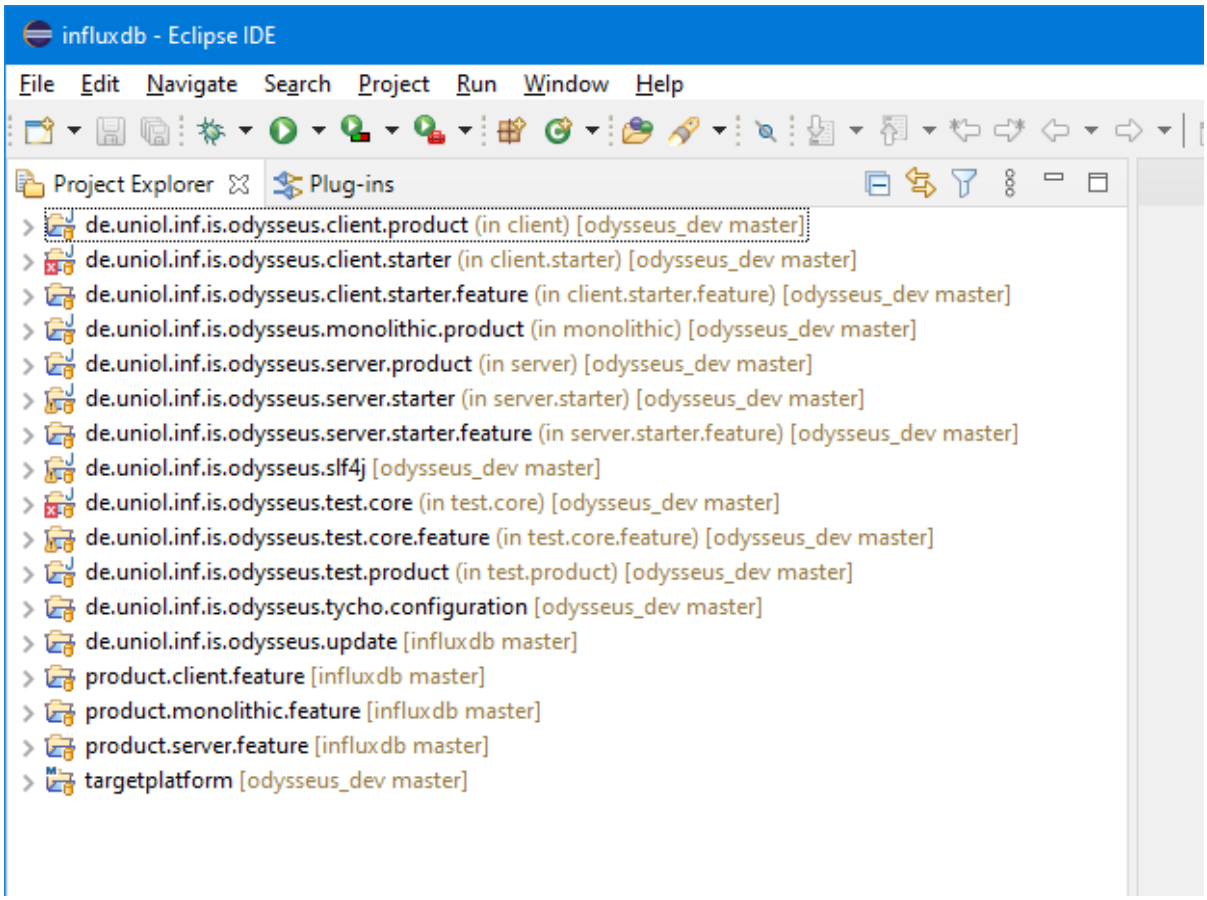


After clicking "Next", select all needed projects.



Now there are many project already opened:

Most of them are from odysseus_dev.



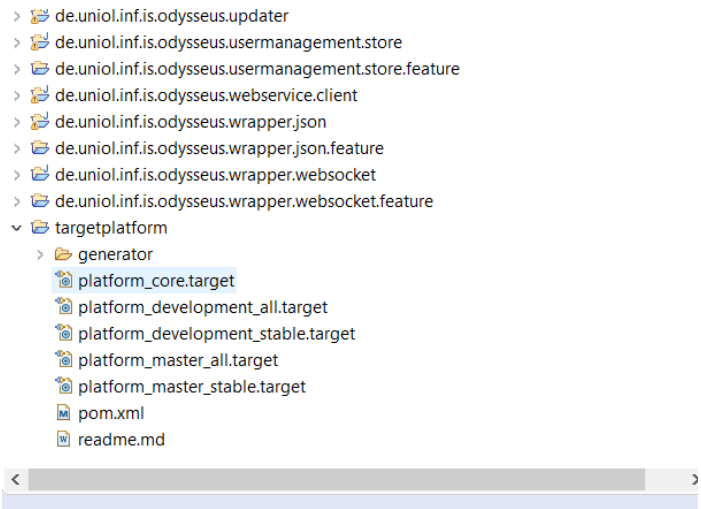
The source code folder has the following subfolders:

- client: contains all bundles that run on the client and don't have any dependencies to the server directly. It mainly offers the GUI.
- common: this holds common stuff and utilities that are needed on both server and client. Common does not have any dependencies to other folders like server, client...
- monolithic: this folder contains bundles that will only work in a monolithic system where server and client are the same product
- resource: possibly some resources for the bundle, but probably empty. The resources are part of the target platform and are downloaded automatically by Eclipse when settings the target platform.
- server: this is the server part of Odysseus and contains Odysseus main functionalities. Except of common, it does not have any dependencies to other folders. It does not have any GUI or client functionalities.
- test: contains stuff for testing and benchmarking

4. Target Platform

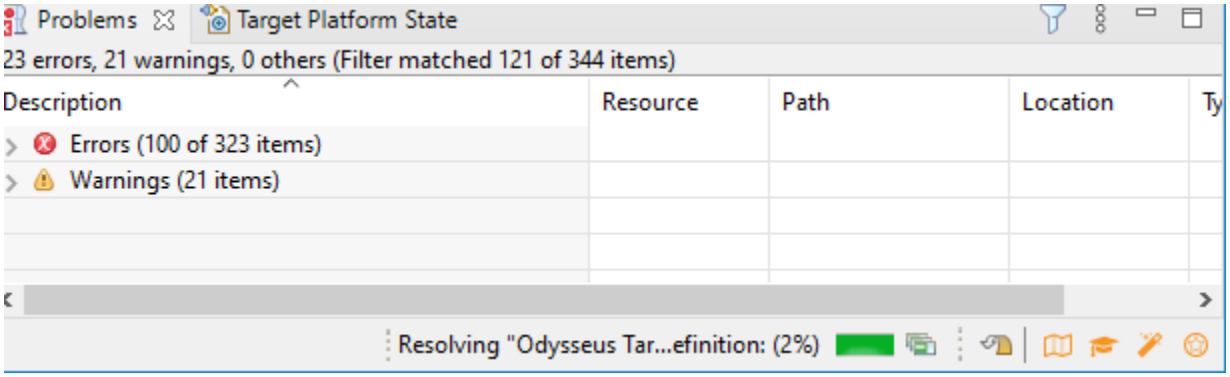
You now see a lot of compiler error. This is because the target platform is not set.

The target platform can be found in the project **targetplatform**.

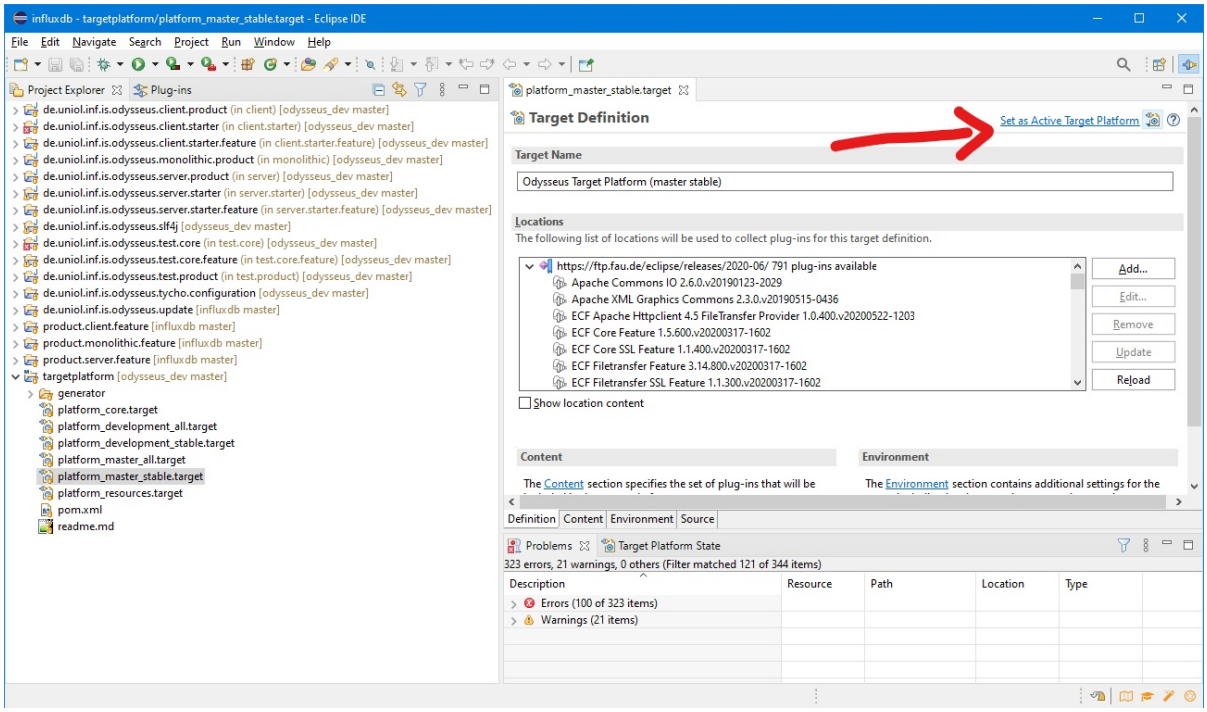


Choose the correct Targetplatform for you work.

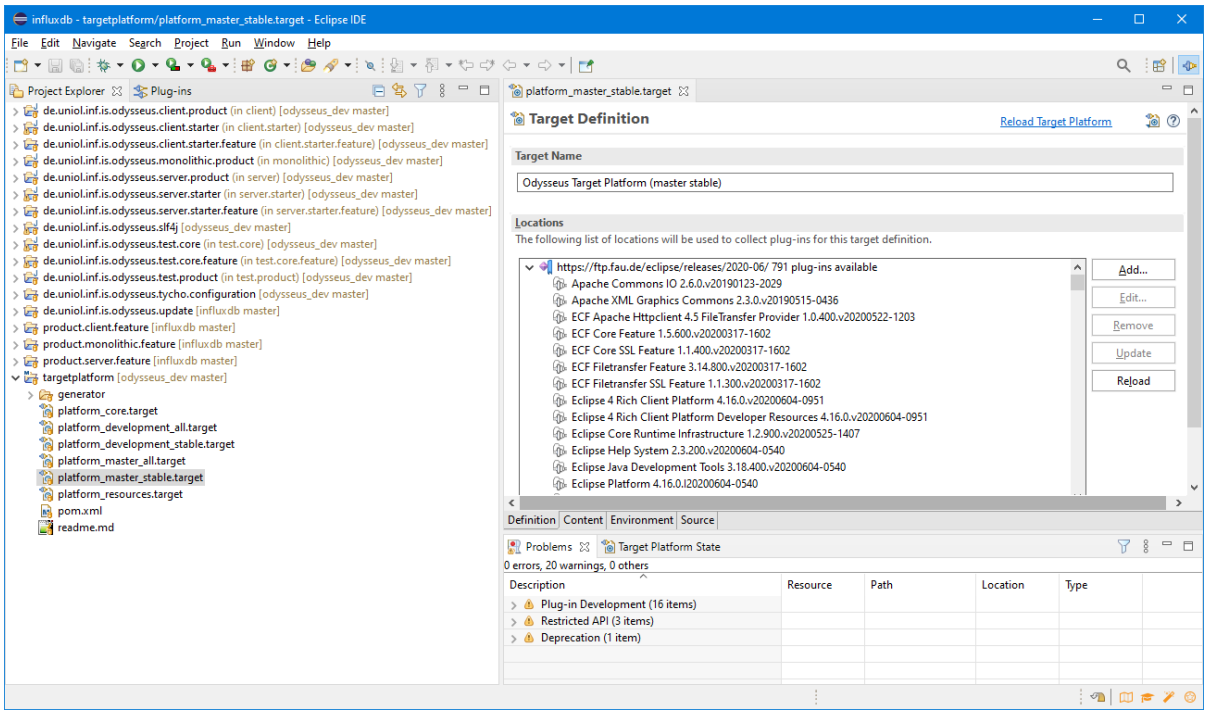
Remark: It works best, when you first open the file (double click) and **wait some time (until the platform is resolved)**.



Click "Set as Active Target Platform" to use this as your target platform. **Important: Do not click, if "Resolving..." is still active!**



After that, the project will be compiled and there should not be any errors anymore.



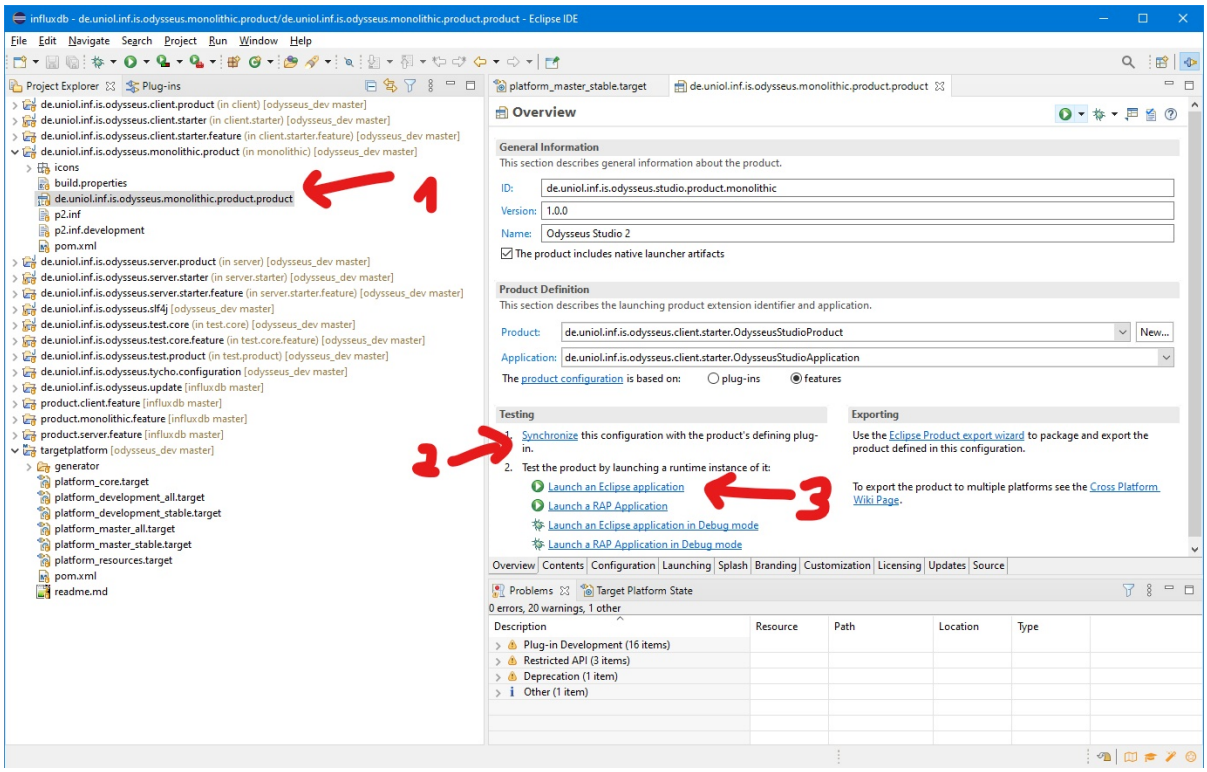
If there are still errors, you should try to update the odysseus_dev folder (sometimes there are errors with the sites, that dliver target platform contents)

5. First Run - Available Products

Now you should be able to run Odysseus. There are some predefined product-definitions to run Odysseus. Depending on the bundles/folders you checked out before they can differ, but the standard submodule should contain three products:

- Odysseus Server - (located in de.uniol.inf.is.odysseus.server.product) - This is just a server-based instance of Odysseus (without any GUI) which can be used e.g. via a REST.
- Odysseus Studio 2 (Client) - (located in de.uniol.inf.is.odysseus.client.product) - This is only the client part (GUI), which tries to connect to an Odysseus Server instance via webservice.
- Odysseus Studio 2 - (located in de.uniol.inf.is.odysseus.monolithic.product) - This combines server and client into a single product and adds some additional bundles that only work in such a monolithic combination.

For your first run, the easiest way is to start the "Odysseus Studio 2" (de.uniol.inf.is.odysseus.studio.product.monolithic). Open this file and go to the tab "Overview". **Click "Synchronize" under "Testing" and afterwards click "Launch an Eclipse application" to run Odysseus** (you can alternatively run it in debug mode if you want). When Odysseus Studio (the GUI) comes up, you have to insert some credentials. A default user is "System" and the password is "manager", the tenant can be left empty. Now Odysseus should be up and running.



6. Additional Information

Features

Since Odysseus is an OSGi based application, it is divided into several bundles (the bunch of projects you checked out and imported before). To keep the overview of all bundles, they are combined to features. Each feature reflects a special functionality of Odysseus. Thus, we have a core.feature that encapsulates all minimal needed bundles or the studio.feature that contains all bundles for the RCP (i.e., GUI) of Odysseus (what we call Odysseus Studio). The combination of several features is called a "product" - which should be runnable. The section above, for example, lists three products that combine different sets of features.

The core feature is necessary because it contains all fundamental functions for Odysseus. Here's a list of Odysseus Features: Features. See Adding features to products how to add a new Feature to an existing product.

Logging

Odysseus comes with the Simple Logging Facade for Java (SLF4J). To use logging, add the `de.uniol.inf.is.odysseus.slf4j` bundle. In this bundle you can find the log4j2.xml to configure the logging behavior (or use one of the given).