
P
C
P
G
P
C
P
G
P
G
S
G
S

Table of Contents

Current state

ESDL is still being heavily developed, with new versions being released periodically. We are using ESDL in many different projects now to get experience in using it and collect feedback. Some parts are still subject to discussion. Feel free to start using it, but some things will definitely change in the coming months.

ESDL Documentation

This documentation contains the following chapters:

- Introduction: General introduction into what ESDL is and for what purposes it can be used.
- Energy Data Modelling: Explanation of the general concepts of Energy Data Modelling
- ESDL concepts: Explanation of the what and why of different parts of the ESDL language
- How to use ESDL: Explanation of how to use ESDL, practical tips
- Tooling for ESDL: Explanation of tooling to contribute to the developments of ESDL, to use ESDL to model an energy system and to integrate ESDL in your own tooling
- Examples: Several examples of how ESDL can be applied

Alternatively, you can download an eBook version as pdf, epub, mobi.

Introduction

The energy transition from fossil to renewable requires a major change in the construction and the operation of the energy system. The energy system is a complex system with numerous relations and dependencies. Therefore, the impact of possible changes is hard to determine. For instance, the transition from heating with natural gas to electrical heating has impact on: gas transport network, electricity supply, required insolation level of build environment and many other facets of the energy system. In order to constructively reason on these changes of the energy system, an objective and complete information basis is necessary. This leads to an integral understanding of energy system. This document provides a description of the Energy System Description Language (ESDL), a first step to an integral understanding of the energy system.

What is ESDL?

The Energy System Description Language (ESDL) is a modelling language created for modelling the components in an energy system and their relations towards each other. Furthermore ESDL is capable of expressing the dynamic behaviour of components in the energy system. For instance the power consumption of an neighbourhood. ESDL describes components by their basic functionality (so called *Energy Capabilities*), these are modelled in 5 abstract categories: Production, Consumption, Storage, Transport and Conversion. ESDL enables energy modelers to model a complex energy system in a generic way. The language is machine readable so makers of energy transition calculation tools and GIS applications can support ESDL in order to enforce the interoperability of their products.

Application Areas of ESDL

ESDL can be used:

- by energy transition calculation tools: common language for energy transition calculation tools. To describe inputs and outputs of those tools. This allows for integration of multiple tools.

- in an Energy Information System: ESDL can be used as a basis for a central energy information system where the energy system of a certain region is registered.
- as a language for (local) governments to model and share their (local) energy system.
- to monitor the evolution of an energy system: multiple ESDL snapshots of a certain area over time provide insight in the evolution of an energy system.
- as a format to share data relevant to energy systems or the energy transition. Examples:
  - CO2 emissions per energy carrier
  - Technology factsheets for specific components, brands, types (e.g. a heat pump factsheet that describes its typical parameters)
  - Cost information of assets, or expected cost developments in future
  - Standard configurations or templates of typical parts of the energy system (e.g. a house with a heat pump, solar panels and an EV charging station)

# #

ESDL concepts

This chapter describes the ESDL concept. The following topics will be covered:

- Design principles: the general design principles behind the language: capabilities and aggregation
- Data types: the most important data types in the language: EnergySystem, Instance, Potential, Area, Asset, EnergyAsset, Port, Profile

Design principles

Energy Capabilities

There are vast amount of different assets in an energy system all having their specific characteristics (e.g cables, heat networks, power plants, solar PV and many more). Modelling an energy system in much detail therefore requires lots of work describing all different assets. However, if we look into the different assets and compare theirs behaviour we see that primarily all assets provide one (or more) core functionality and can all be categorized in one of the following five capabilities.

Some examples:

- Production: SolarPV panel, Wind Turbine, etc.
- Consumption: Households electricity consumption, heat consumption of a city, etc.
- Storage: Home battery, heat buffer, etc.
- Transport: District heating, electricity grid, etc.
- Conversion: Heat Pump, Transformer, Gas burner, etc.

Describing the assets in terms of capabilities achieves two things that are very welcome. Firstly we can make an abstraction of an assets by only modelling characteristics that are relevant per capability. This results in an simplification for the user of ESDL since he now can approach different asset types in a similar way. For instance a PV panel (Production) can be approached in a similar way as a wind turbine (Production) with slightly different characteristics. This makes life easier for those who want to reason on energy systems (evolution) with calculation tools.

Secondly, every capability comes with a natural way of aggregation. For instance, if the consumption capability for every house in a street is defined (for instance as electric power consumption/year), then the aggregated consumption capability of the whole street can be determined by calculating the sum of all the consumption capabilities. This is achieved by modelling all energy consumers as consumer capability. The following section describes the use of this aggregation in ESDL.

Levels of aggregation

Modelling an energy system can be done in multiple different levels of detail (geographical scales). The level of detail needed depends on the application. For making a energy plan on country level it is much likely not necessary to model the energy consumption of every single building in that country separately with a high time resolution. However, for designing a district heating installation in a specific area it might be possible that that detailed information is necessary. ESDL does not prescribe a certain detail level of modelling of an energy system, it supports describing energy systems on multiple levels. Depending on the application, or the availability of data, a modeler can choose an appropriate modelling level

ESDL supports the modelling of energy systems on detail levels such as: house, building, neighbourhood, city etc . Because in ESDL everything is modelled in terms of capabilities (previous section) the different levels are also modelled in terms of capabilities. This means that for instance a house can have a certain level of production, consumption and storage, the ESDL model on neighbourhood scale can be expressed in the same capabilities by

taking the aggregated values of the capabilities of all houses in that neighbourhood. The same can be done for city and country scales. This results in a generic way of modelling energy systems independent of the scale (detail level). This simplifies reasoning on different energy systems for applications.

Example of modelling with energy capabilities

An model of a possible future energy system of the island of Ameland, a small island in the north of the Netherlands, is depicted in Figure 1.

Data types

This section describes the data types in ESDL:

Energy System

The EnergySystem class will be the place to start if you want to model an energy system for a certain region.

An EnergySystem contains the following information:

- Instances
- Potentials
- Measures
- Parties
- EnergySystemInformation

Instances

An EnergySystem can contain zero or more Instances. Instances are used to represent different representations of the **same** EnergySystem. Most of the times only one Instance will be used. The primary use case for having more than one Instance is when you have different aggregations of the same EnergySystem in the same model (e.g. the same region on house level and aggregated on neighbourhood level).

All instances contain an Area. The Area is the class to represent a geographical or logical area. An Area contains all assets in that area or can be subdivided into more Areas

Potentials

Potentials can be used to describe all different kinds of energy potentials. Examples of potentials are:

- GeothermalPotential: to describe the potential for geothermal energy
- WindPotential: to describe the potential for wind turbines
- SolarPotential: to describe the potential for solar PV fields
- LegalArea: to describe for example areas where no aquifers are allowed because the underground water is used for drinking water (a negative potential).

Measures

Measures can be used to describe possible assets that can be installed in the EnergySystem (e.g. added by a model calculating minimum investments required to match future demands).

Parties

Parties can be used if you want to model ownership of assets

EnergySystemInformation

The EnergySystemInformation class is a container for additional information about the EnergySystem in general.

Currently the following information can be added:

- AvailableEnergyCarriers: list of available energy carriers with information about energy content and CO2 emissions. Power plants typically refer to energy carriers to indicate what the source of their energy is.
- EnergyPrices: prices for gas, electricity

Areas

The Area class represents a physical geographic area or a more abstract logical area. In both cases it is the 'asset container', in a sense that all assets within the area are contained by the Area instance. A physical geographic area is a specific area with a location and geometry (and can be pointed at on a map). A logical area can be an non-specific area, like 'a neighbourhood with houses from the period 1970-1990' without specifically stating where this neighbourhood is located.

An area can have a scope (like country, region, city, village, street, building and lots more) and have a type (like land, road, water, sea, ...). An area can be subdivided into smaller areas, like a city that is divided into neighbourhoods and neighbourhoods being divided into streets.

Areas can have the following properties:

- Location and geometry
- EnergyPotential (wind, solar PV, geothermal, aquifer, ...)
- Legal restrictions
- Social properties
- Economic properties

Items, Assets, EnergyAssets and Services

Items

The Item class represents an abstract item. Items can have an id, a name, a short name and a description. The following classes are derived from the item class:

- Assets: an Asset is an Item with a physical representation.
- Services: A Service is a logical Item.

Assets

Assets are all physical items in the EnergySystem. Assets can have a location, a geometry, commissioning and decommissioning dates, cost information (investment, installation and operation and maintenance costs).

Assets are contained by Areas.

EnergyAssets

An EnergyAsset is an Asset with one or more ports allowing to model connections between assets.

The EnergyAsset class is subdivided into 5 categories:

- ProductionAsset: e.g. wind turbine, solar panel, geothermal source
- ConsumptionAsset: e.g. electricity demand, heat demand, gas demand
- StorageAsset: e.g. battery, buffer
- TransportAsset: e.g. electricity network, gas network, cable, pipe, transformer, heat exchange
- ConversionAsset: e.g. gas heater, heat pump, power plant, CHP, fuel cell

Services

Example Services are demand response services and aggregator services.

Overview of EnergyAssets (TODO)

Production Assets

Consumption Assets

Storage Assets

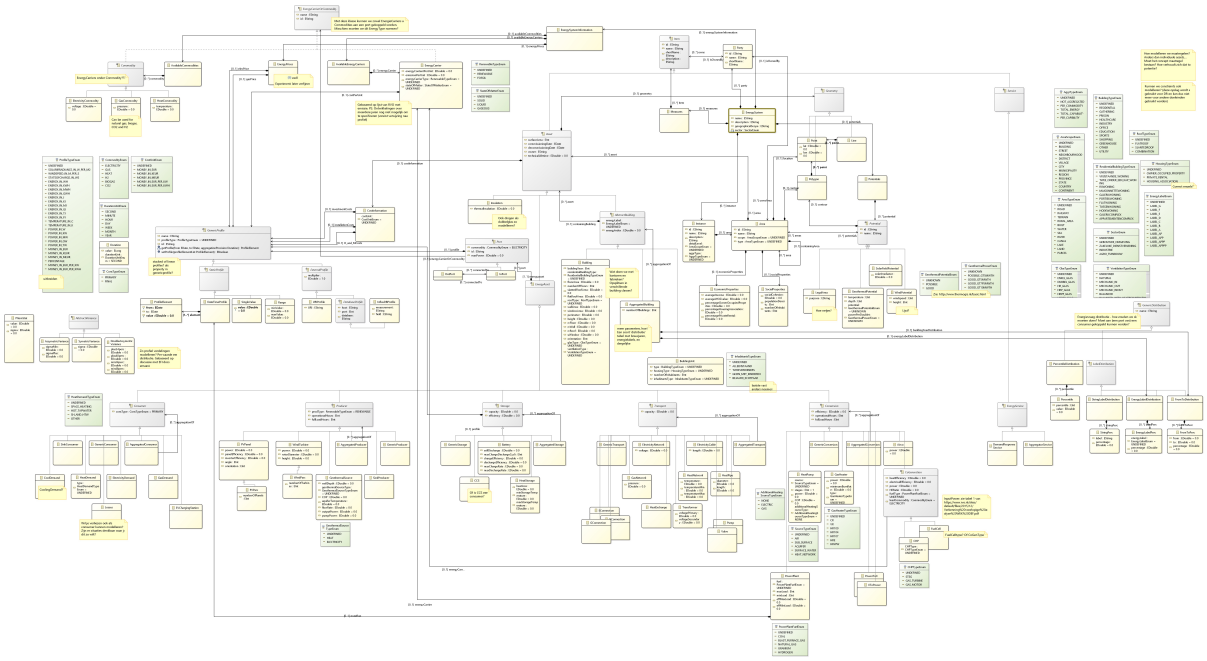
Transport Assets

Conversion Assets

Profiles (TODO)

ESDL model

A graphical view on the ESDL model can be found here:



Source: https://raw.githubusercontent.com/EnergyTransition/ESDL/master/esdl/model/esdl.png

How to use ESDL

Models and descriptions

Levels, house -> country

Geographic or non-geographic based models

Start with:

- ES
- Energy Carriers
- Profiles
- *house
- HP template

Tooling for ESDL

Modelling

The concepts and the relations among the concepts in ESDL are defined in a structured model: the ESDL model. A modelling language is subsequently used to model the ESDL model. This modelling language is called the Unified Modelling Language (UML). But to make ESDL more accessible to both modelers, developers and end-users, ESDL is based on a dialect of UML called ECore. Therefore ESDL is modelled in ECore.

ECore is part of the Eclipse Modelling Framework (EMF). Eclipse is a well-known open-source Integrated Development Environment (IDE) that supports developers with all kinds of tools to develop software programs.

The EMF gives us several tools for free compared to using UML:

- Code generation. It can automatically generate Java-classes, XSD schema's and UML models from the ECore model.
- Provide semantics for modelling concepts. These semantics are geared towards generating tools to read, visualize and edit your ESDL files. EMF provides two ways for these semantics: (1) a static version (using the code generation facility, creating actual files on disk), and (2) a dynamic version that is in real-time created from the concepts you are modelling. This allows for direct feedback on the changes you make to the model as editors are instantly updated with these changes.
- A graphical editor to manipulate the model.

These properties make it easier to adopt ESDL, as this project not only provides an XML Schema for usage in energy transitions tools, letting the developers write XML-based ESDL-files themselves, but also provide rich editors to manipulate ESDL-files.

The tools can be spilt up in two categories:

1. Tools to edit the ESDL model itself and contribute the ESDL specification
2. Tools to edit the ESDL model instances (the ESDL-files)

#

#

Contributing to the ESDL model

To contribute to the ESDL language you have to install Eclipse and the Eclipse Modelling Tools which contain the Eclipse Modelling Framework.

These can be downloaded from the eclipse.org website. Easiest is downloading the modelling package directly with a new Eclipse installation, but alternatively you could also install these modelling tools as plugins in your existing Eclipse installation. Since EMF is still under active development, and ESDL is using new features from these latest releases, it is required to use the latest version of Eclipse.

Currently Oxygen 3a is the latest stable release of Eclipse. The complete Zip-package can be downloaded here. Or you can download the Eclipse installer from the general download page and select the Eclipse Modelling Tools from the list of options after running the executable.

Setting up the Eclipse Modelling Tools

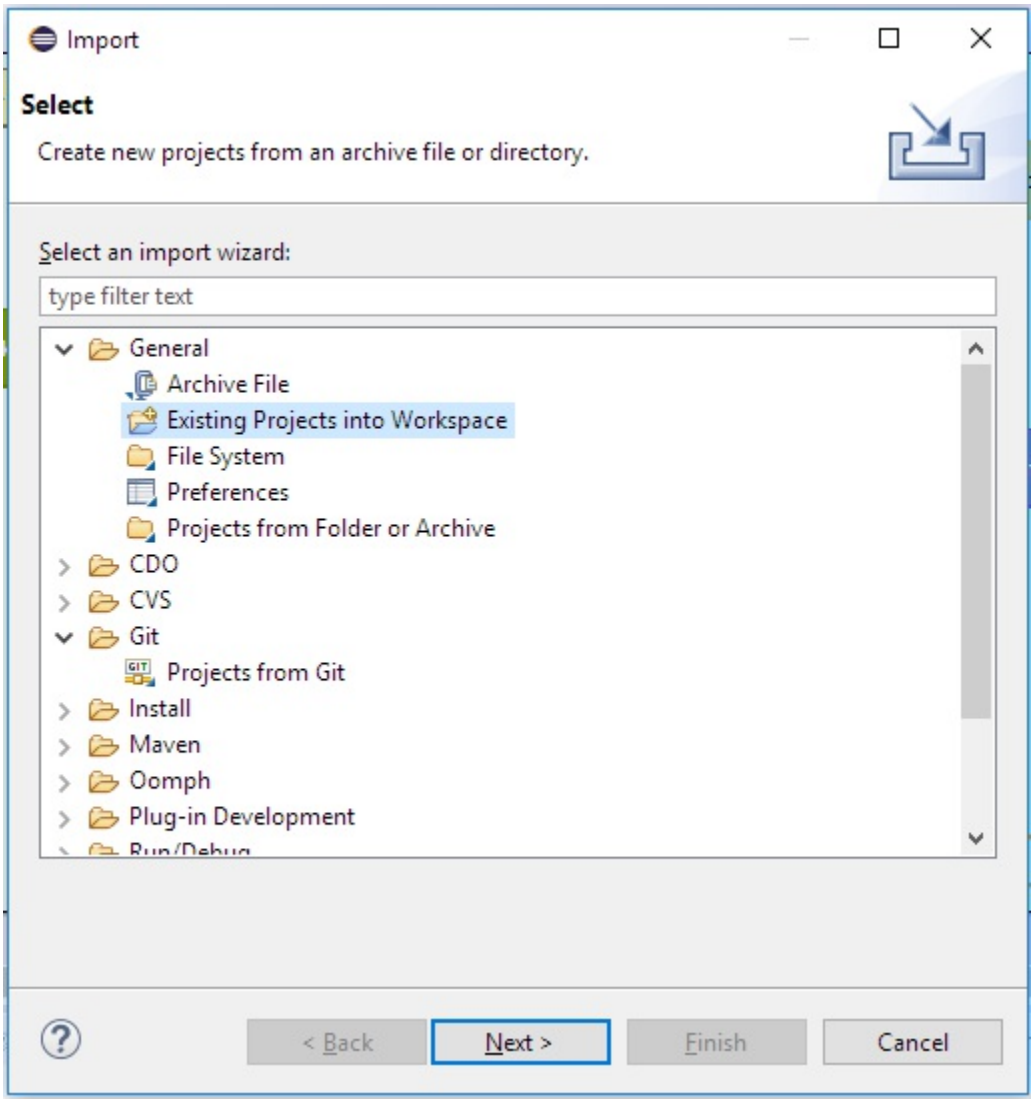
Download Eclipse Modelling tools

Currently Oxygen 3a is the latest stable release of Eclipse. The complete package can be downloaded here. Choose **Eclipse Modelling Tools** from the list. Or you can download the Eclipse installer from the general download page and select the Eclipse Modelling Tools from the list of options after running the executable.

Setting up Eclipse Modelling Tools

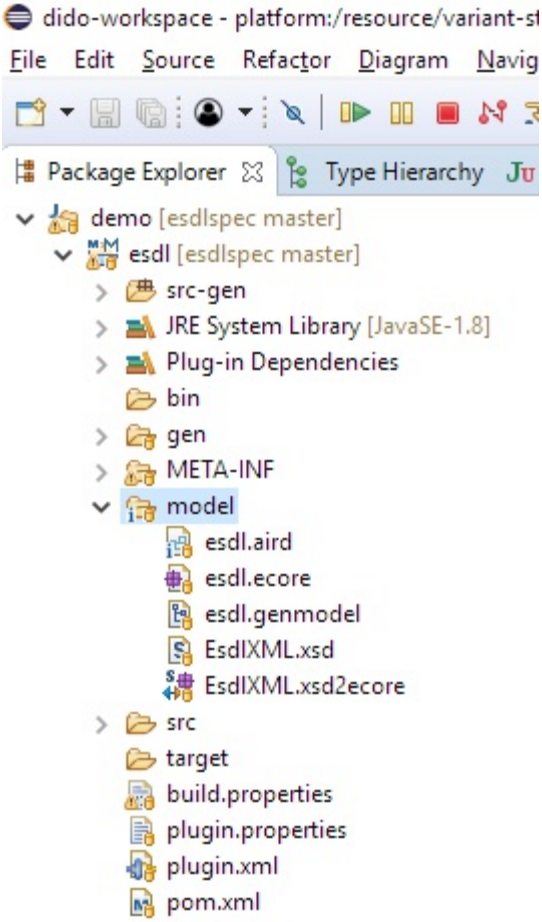
After downloading and installing Eclipse and Eclipse Modelling Tools, you start Eclipse Modelling Oxygen by double clicking on the created icon on the Desktop (if you are using Windows...). It will ask you where you want to place your workspace files. Select a logical location and you end up with a welcome screen.

If you did not clone the repository yet, now is a good moment with your preferred GIT tool or use Eclipse to import the repository directly.



Next step is importing the ESDL git repository by using File -> Import... If you already cloned the repository locally, use *Existing Project into Workspace* and the project will be imported. If you want to directly import the project from Git, use Git->Projects from Git.

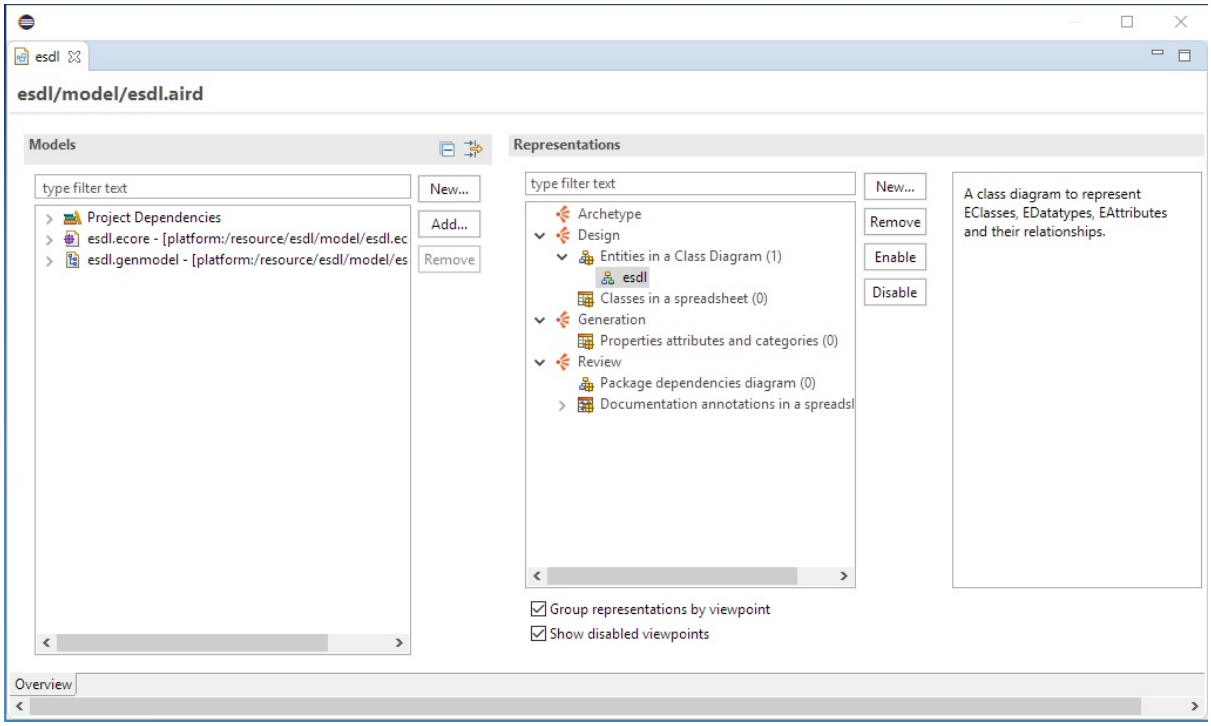
After importing, the *Package explorer* will show the following files and directories:



In the model folder you see several files:

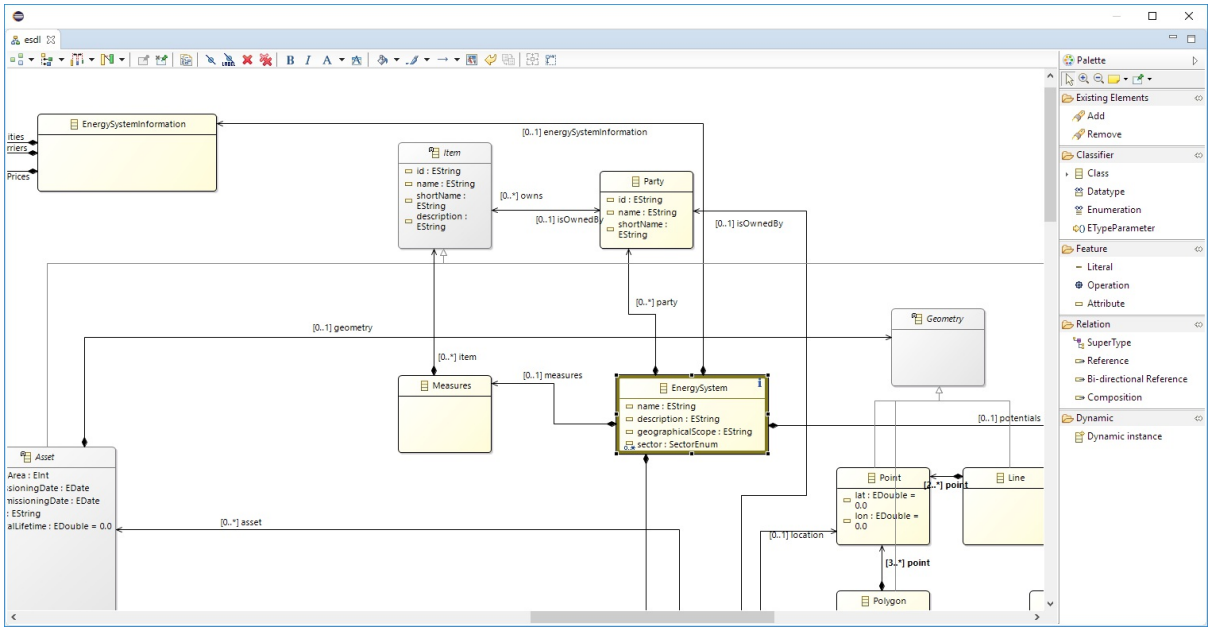
- `esdl.aird` - In this file the graphical representation of the ESDL concepts are documented, such as the (x,y) location of the classes, the colour of the classes, the font, etc. It allows us to visually modify the contents of the ESDL model. If you double click you can open the ESDL model diagram editor and manipulate the model.
- `esdl.ecore` - This is ESDL. The UML-based model describing all ESDL concepts: the vocabulary and grammar of an Energy System. If you double click you can edit the ESDL model using a tree-based structure (less convenient than the diagram editor)
- `esdl.genmodel` - This file describes how to generate Java-files and XSD Schemas from an `ecore`-file. It contains a dozen configuration options.
- `esdlXML.xsd` - this is the XML Schema file for ESDL
- `esdlXML.xsd2ecore` - maps the XSD to the `Ecore` model.

Double clicking on the aird-file opens the following dialog:



The AIRD file shows the model dependencies and representation. The Design->Entities in a Class Diagram is the most interesting. Double click on `esdl` to open this representation.

There is also a Review representation that shows all the documentation in the model. (although this is work in progress).



Now you're setup to edit the Energy System Description Language!

Generating model, edit and editor code (TODO)

Using the gen-model

Creating the jars

Using ESDL to model an energy system

At the moment there are three initiatives to view and edit/generate ESDL instances (they need to be documented however)

- Eclipse based ESDL tree editor
- Graphical ESDL viewer (TODO)
- Eclipse based visual ESDL editor (TODO)

ESDL tree editor

Introduction

This document describes the following topics:

- Installing Eclipse ESDL editor
- Defining your first ESDL model

Installing Eclipse ESDL editor

Step 0: Download Oracle JDK (if necessary)

Go to www.oracle.com/technetwork/java/javase/downloads/index.html and press the “Download” button under the “JDK” text to download the JDK.

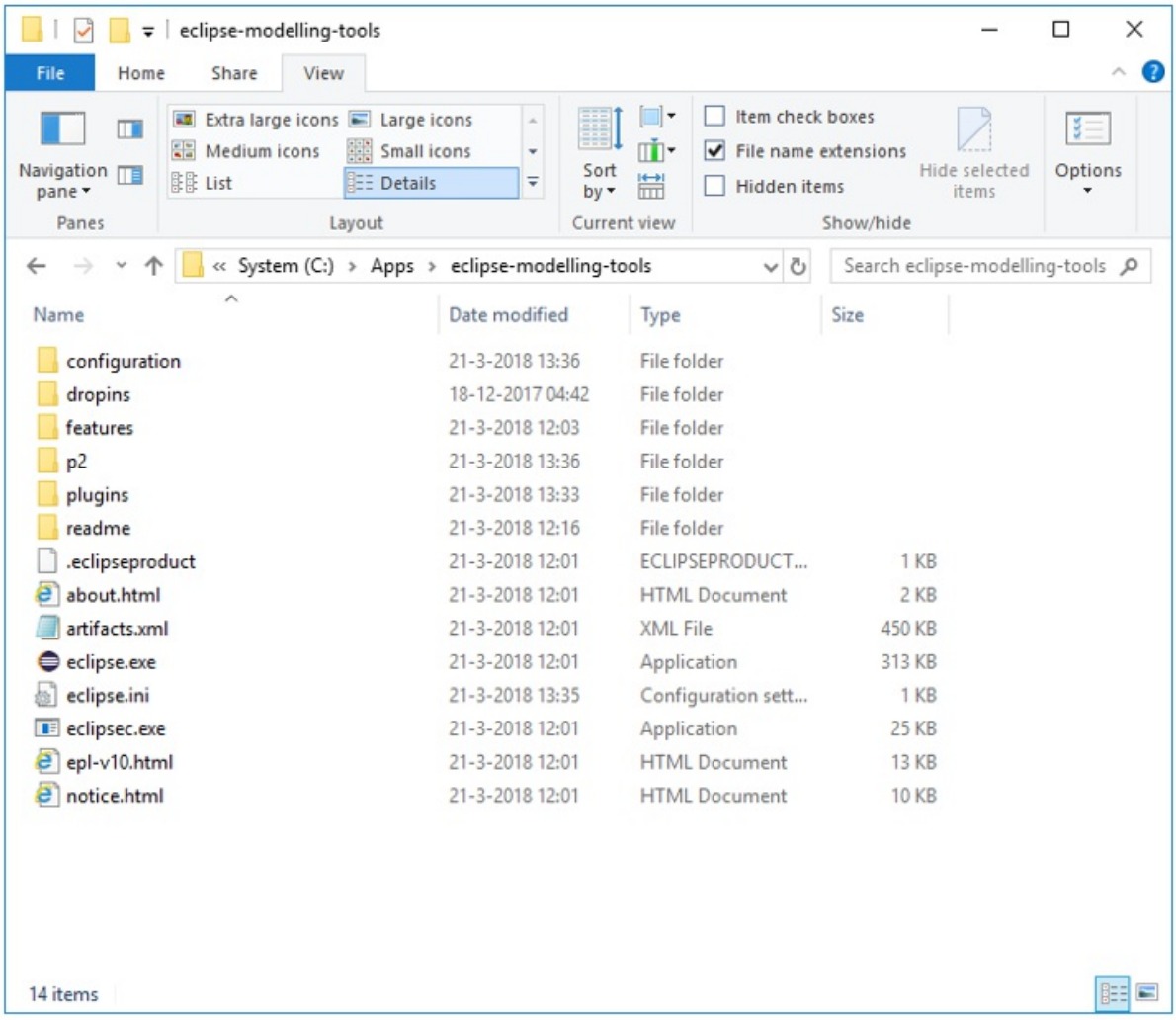
Step 1: Download Eclipse Modelling Tools

From: http://www.eclipse.org/downloads/packages/

Select for example “Windows 64-bit”: http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/oxygen/2/eclipse-modelling-oxygen-2-win32-x86_64.zip

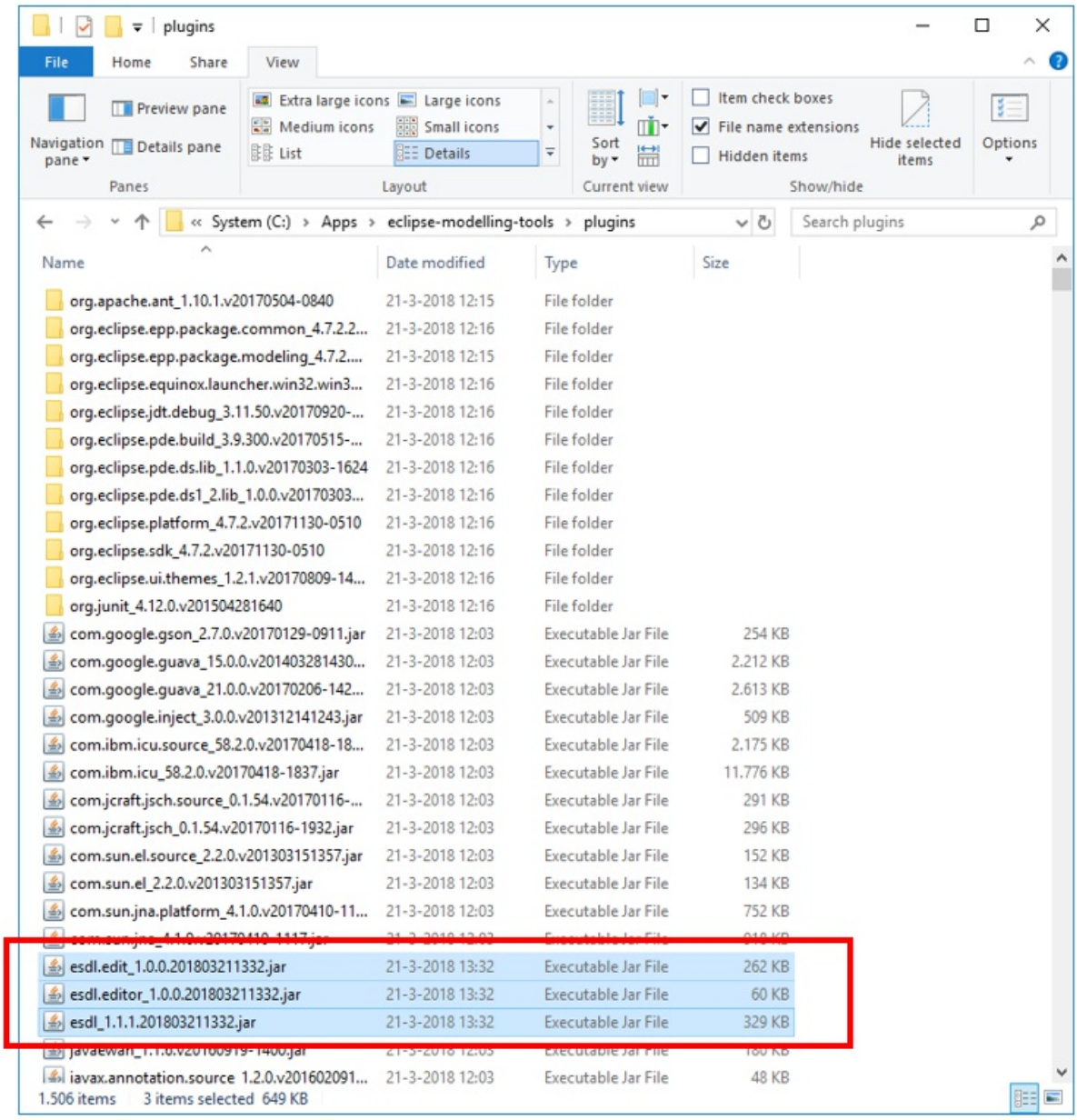
Step 2: Unzip zip-file on local hard drive

Unzip for example to C:\Apps\eclipse-modelling-tools



Step 3: Add ESDL plugin files to plugins directory

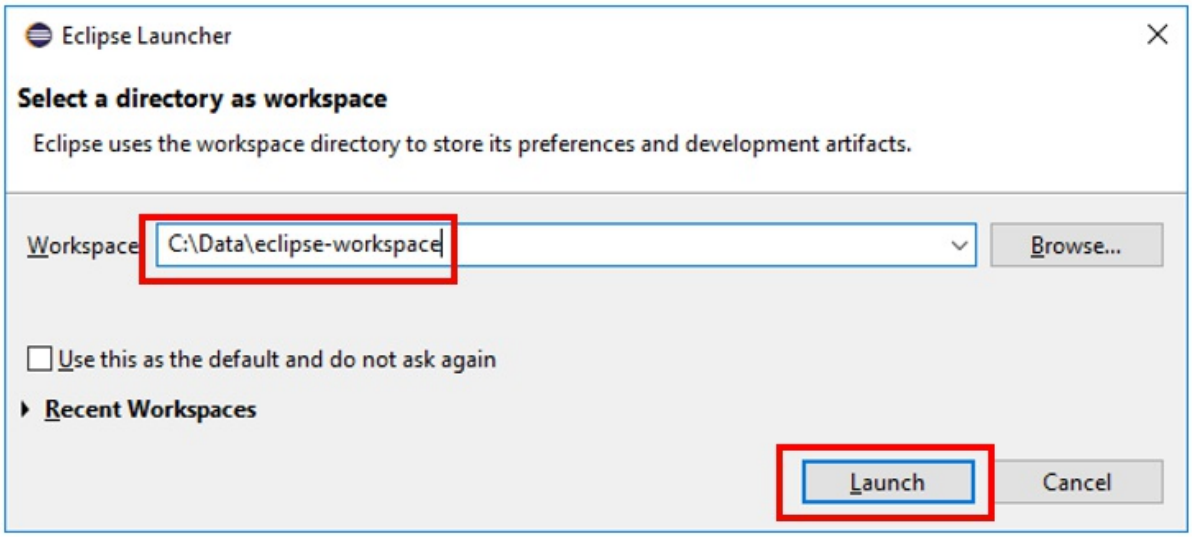
A zip file with the ESDL plugins can be found here.



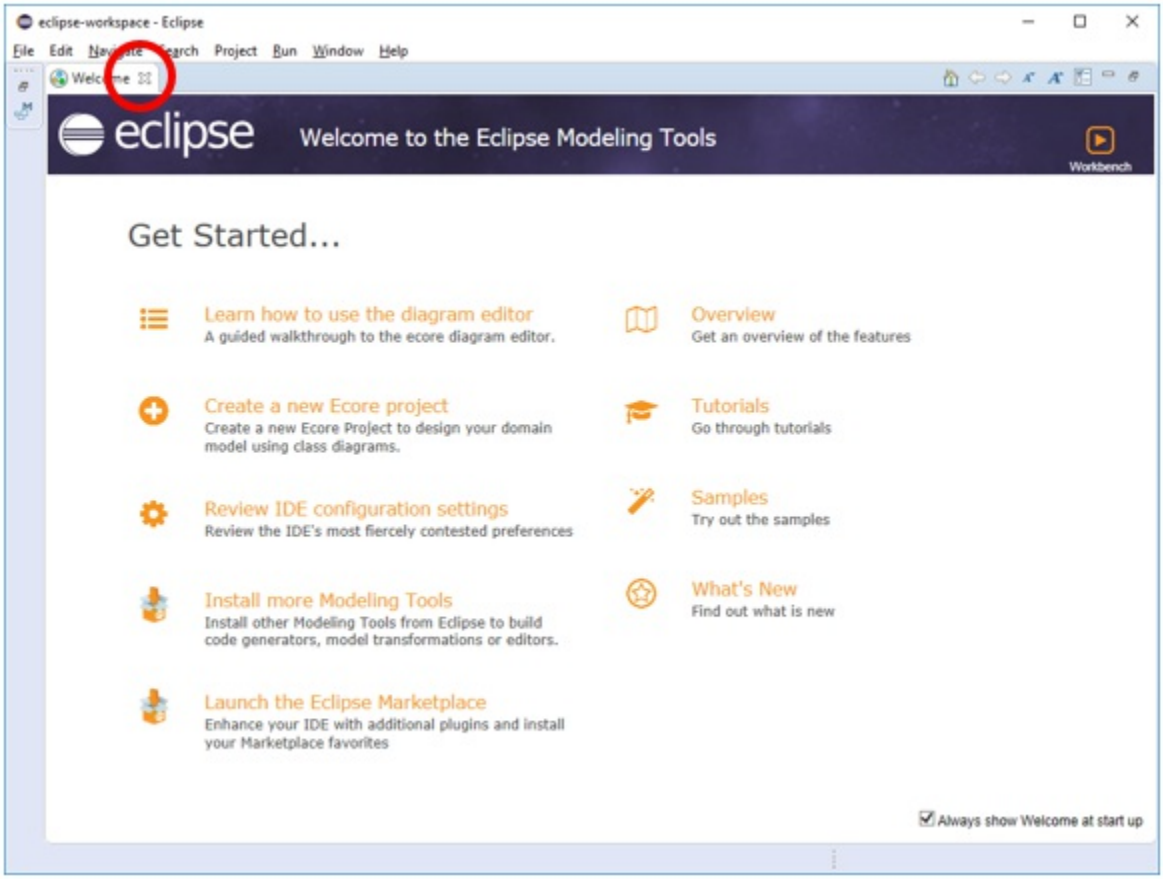
Defining your first ESDL model

Step 1: Start Eclipse and select workspace

Start eclipse by executing eclipse.exe from the folder your installed eclipse in Select a directory to store all your eclipse projects and press the “Launch” button:

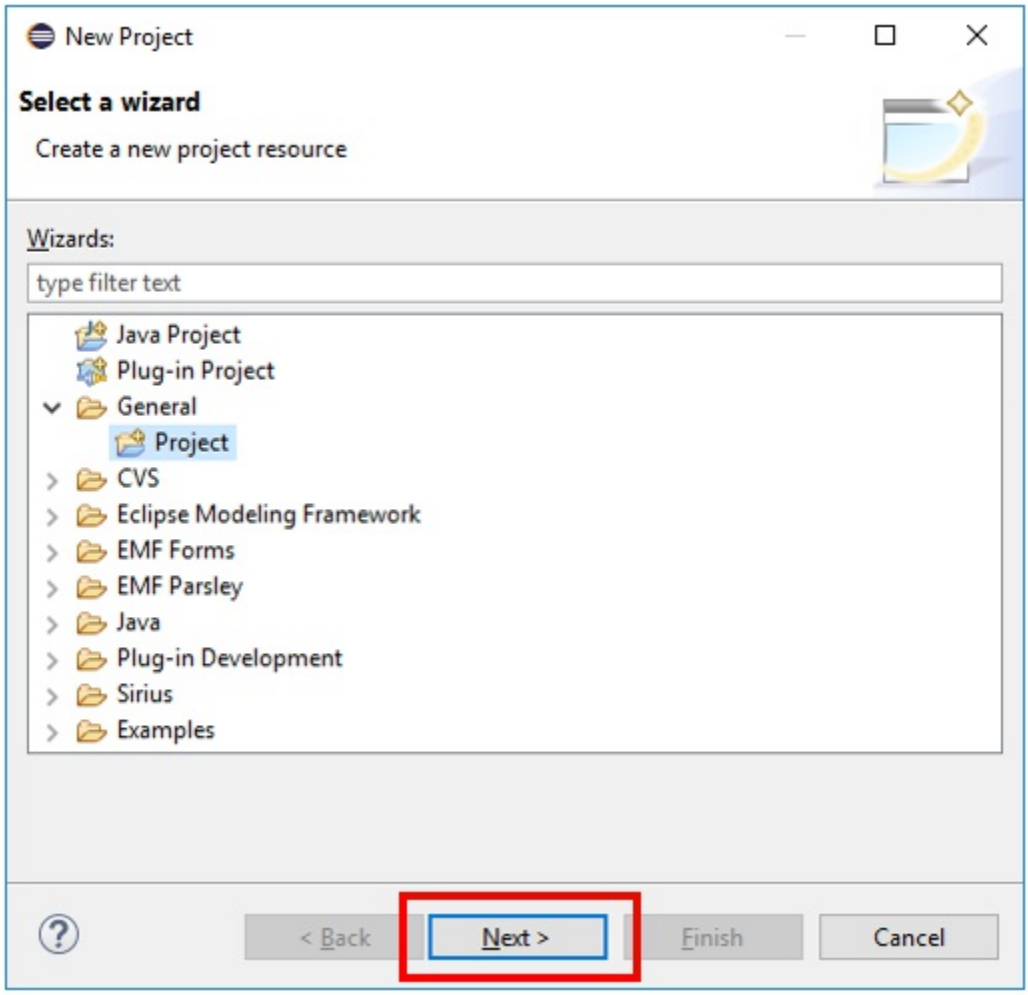


Press the “X” in the welcome window:

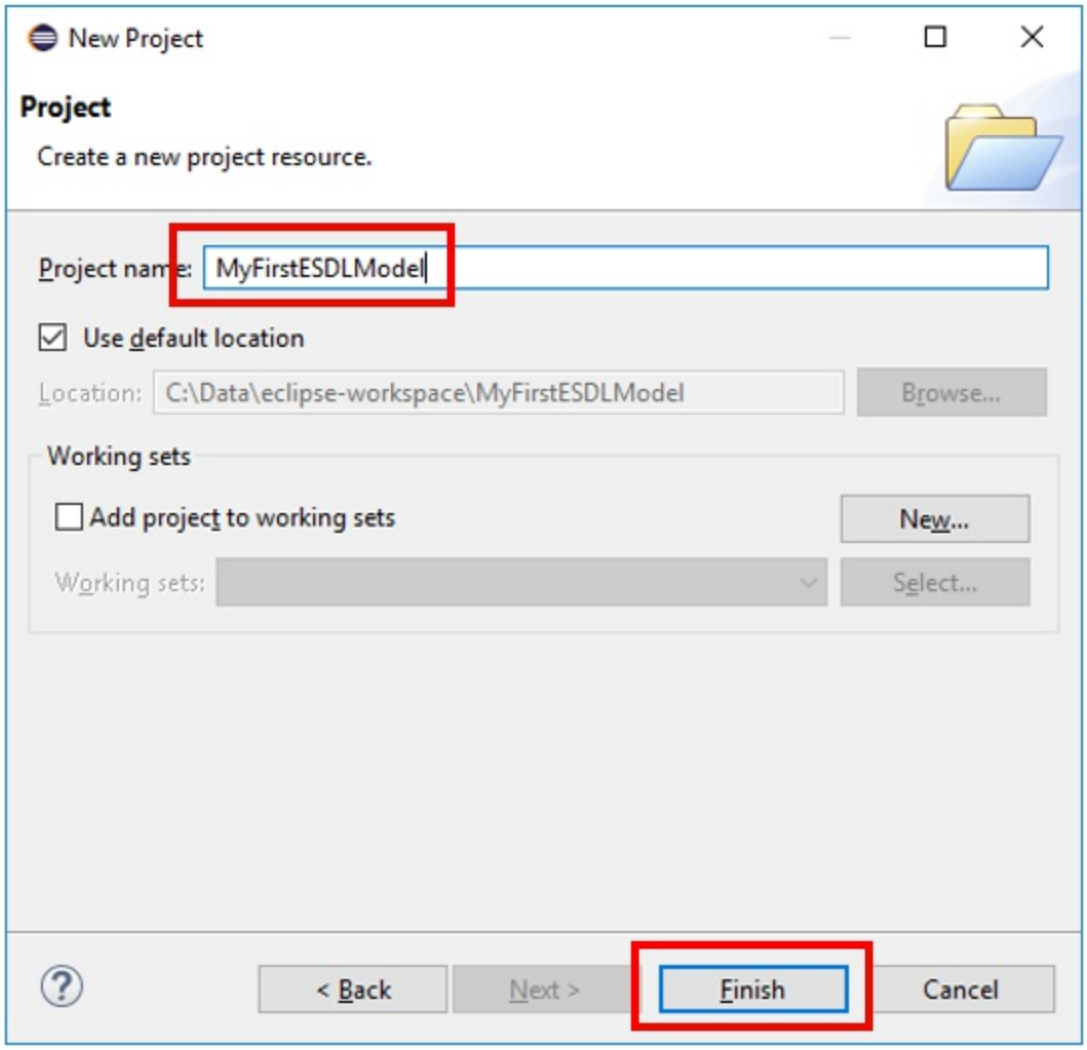


Step 2: Create new project

Select “File” --> “New” --> “Project…” Select “Project” from the “General” folder and click “Next”

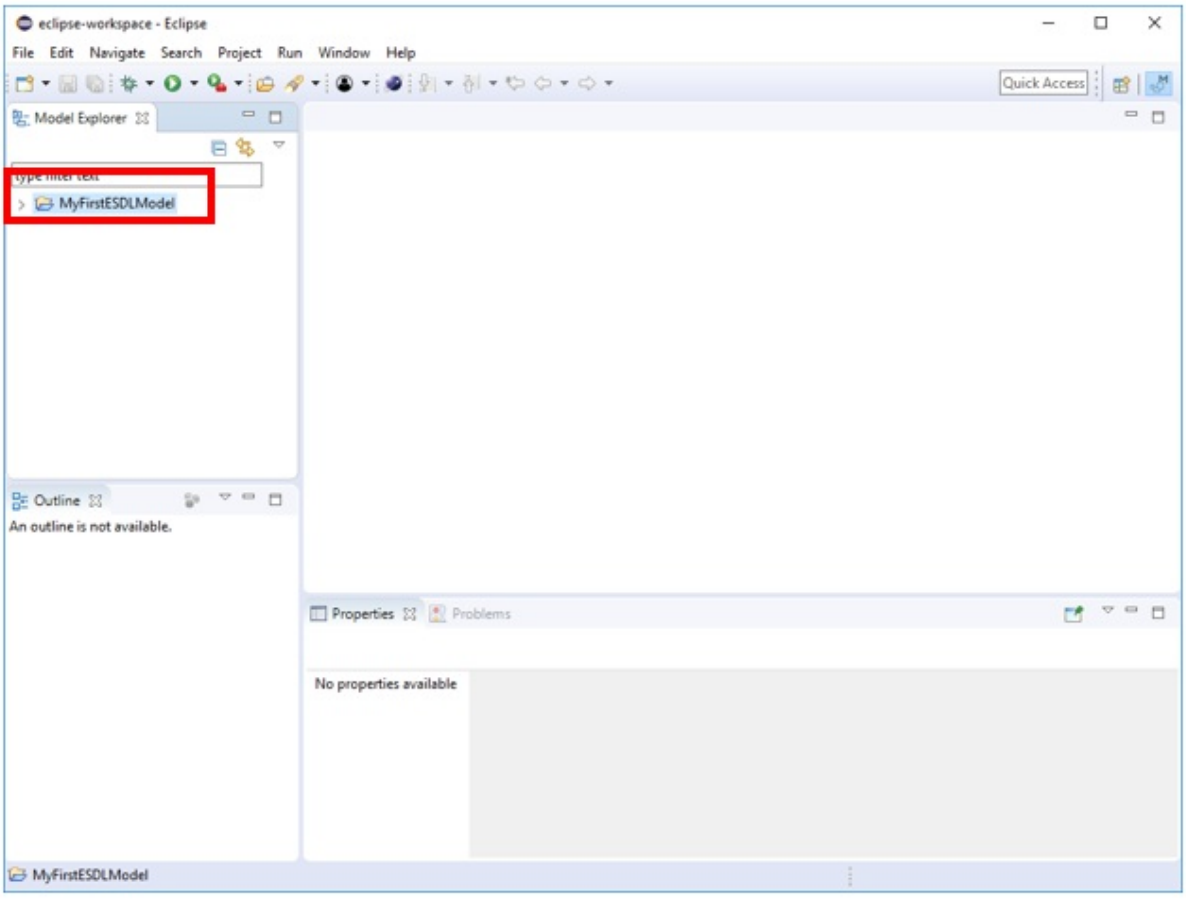


Give your project a name and click “Finish”:

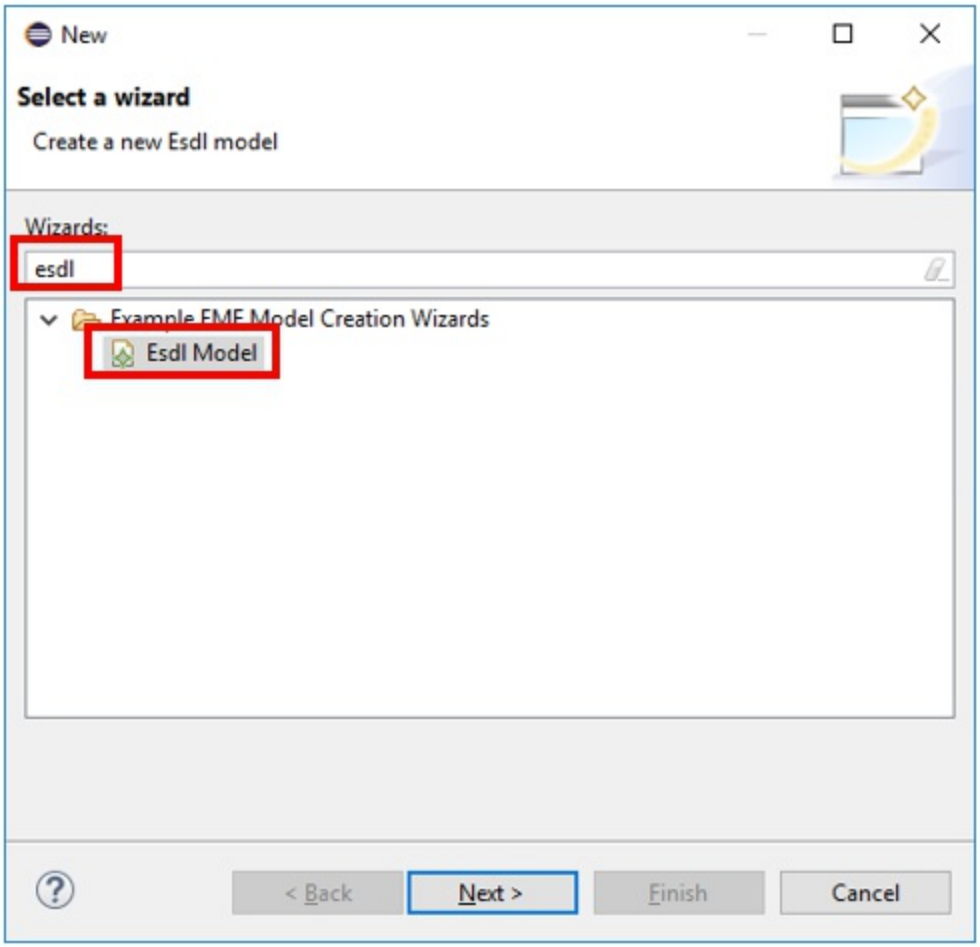


Step 3: Create new ESDL model within the project

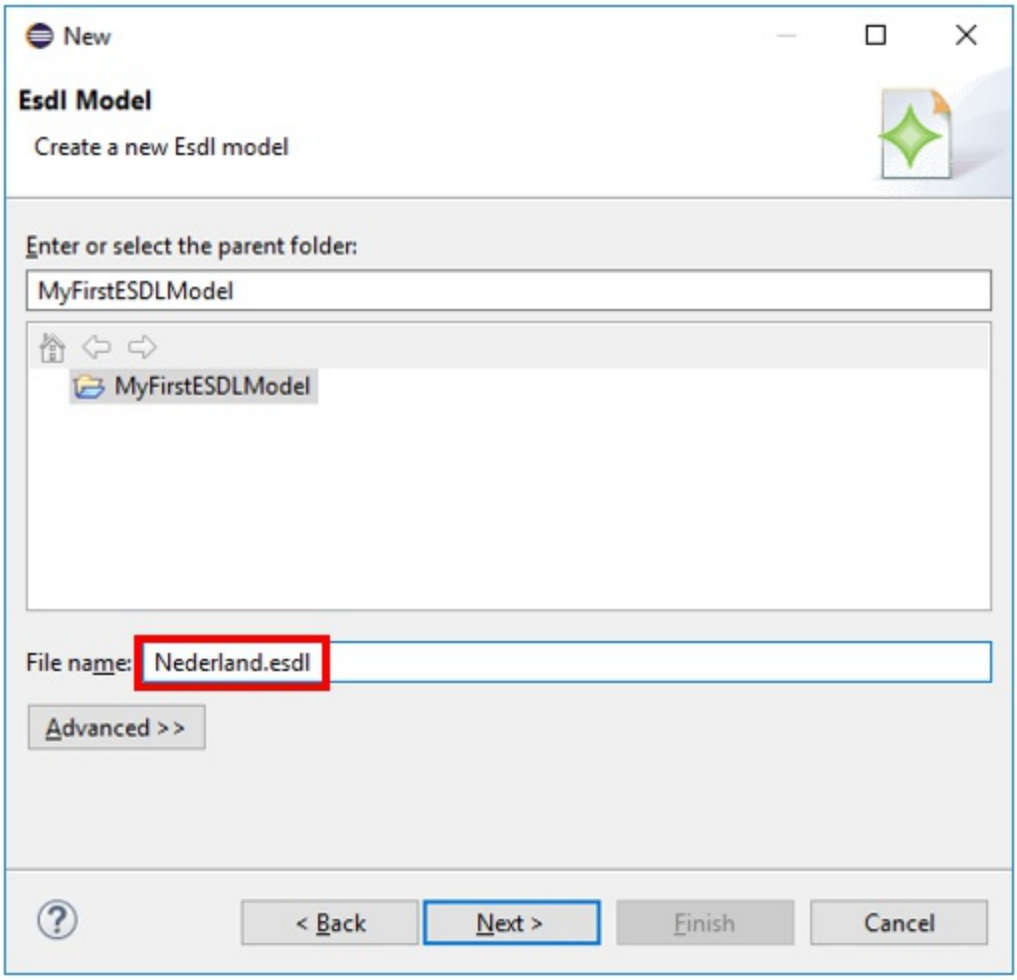
Right click on your project and select “New” --> “Other…”



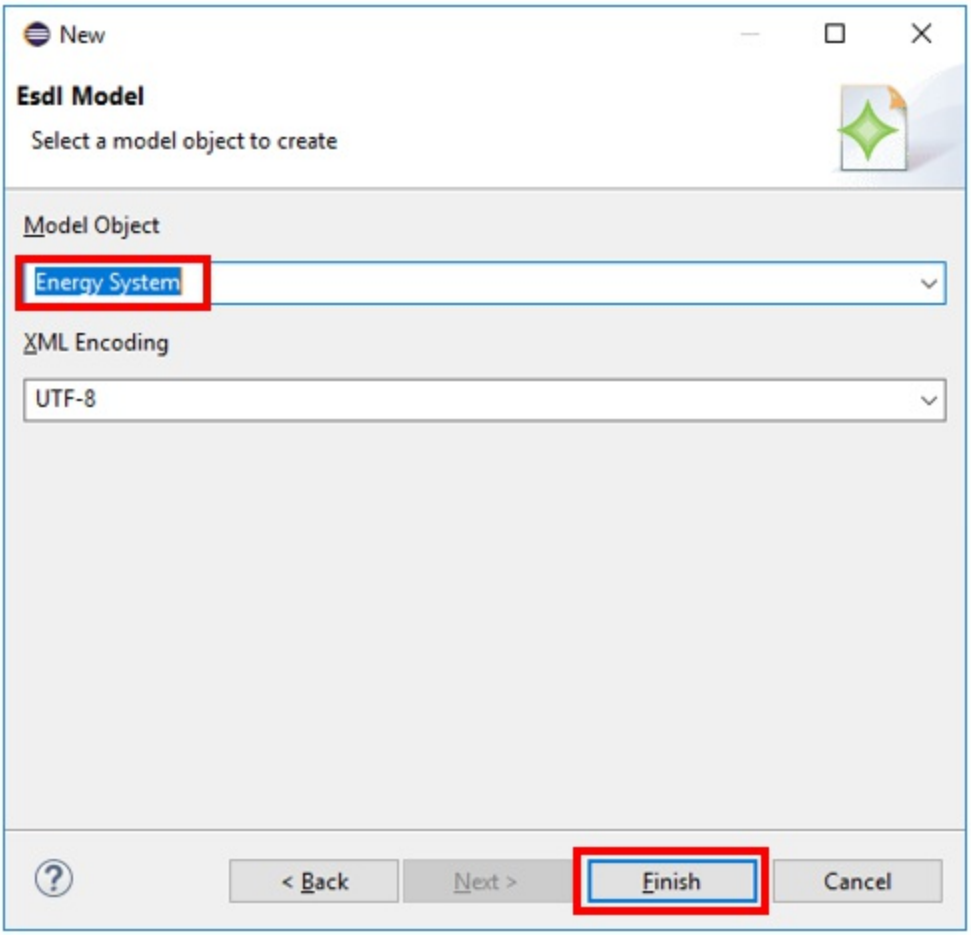
Enter “esdl” in the filter textbox and select “ESDL Model” from the available wizards:



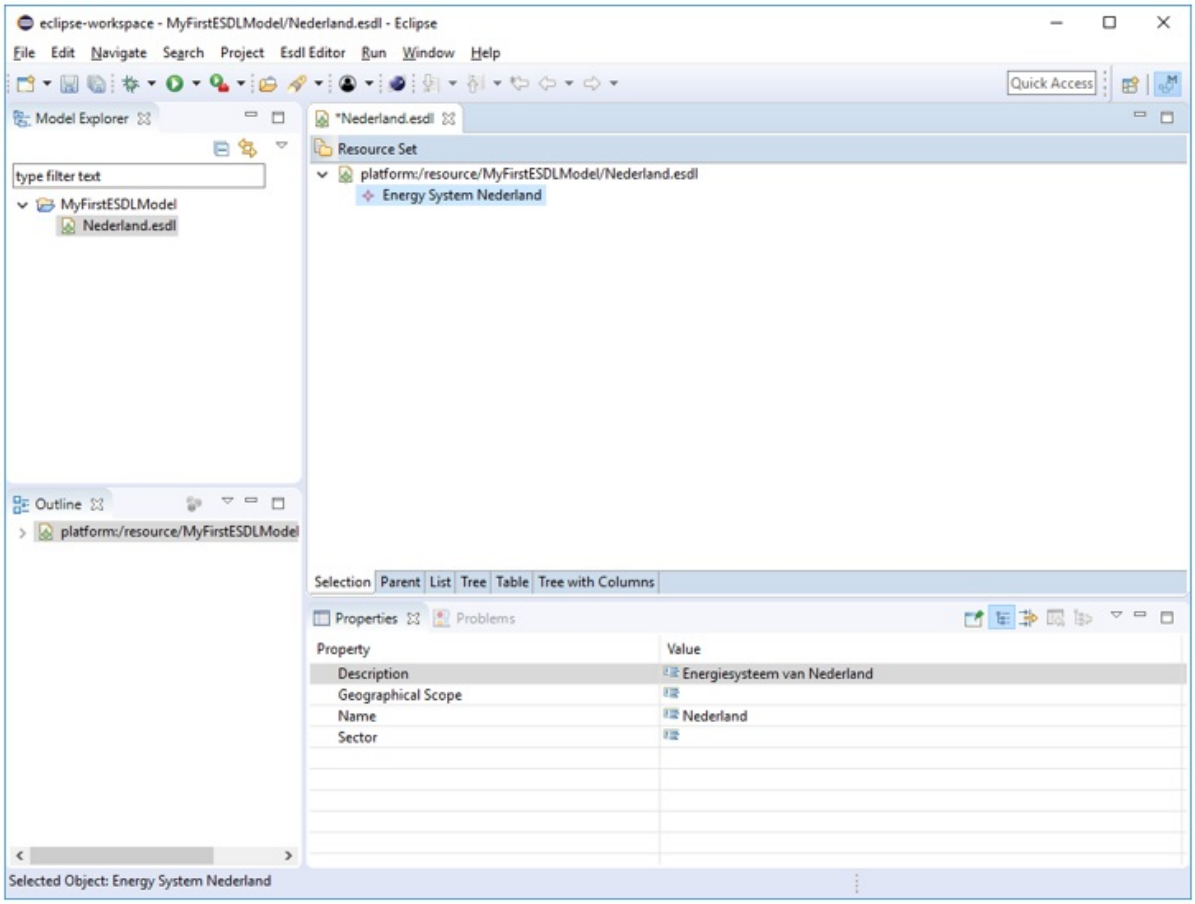
Give your ESDL model a name:



Select “Energy System” as a model object to create and click “Finish”:

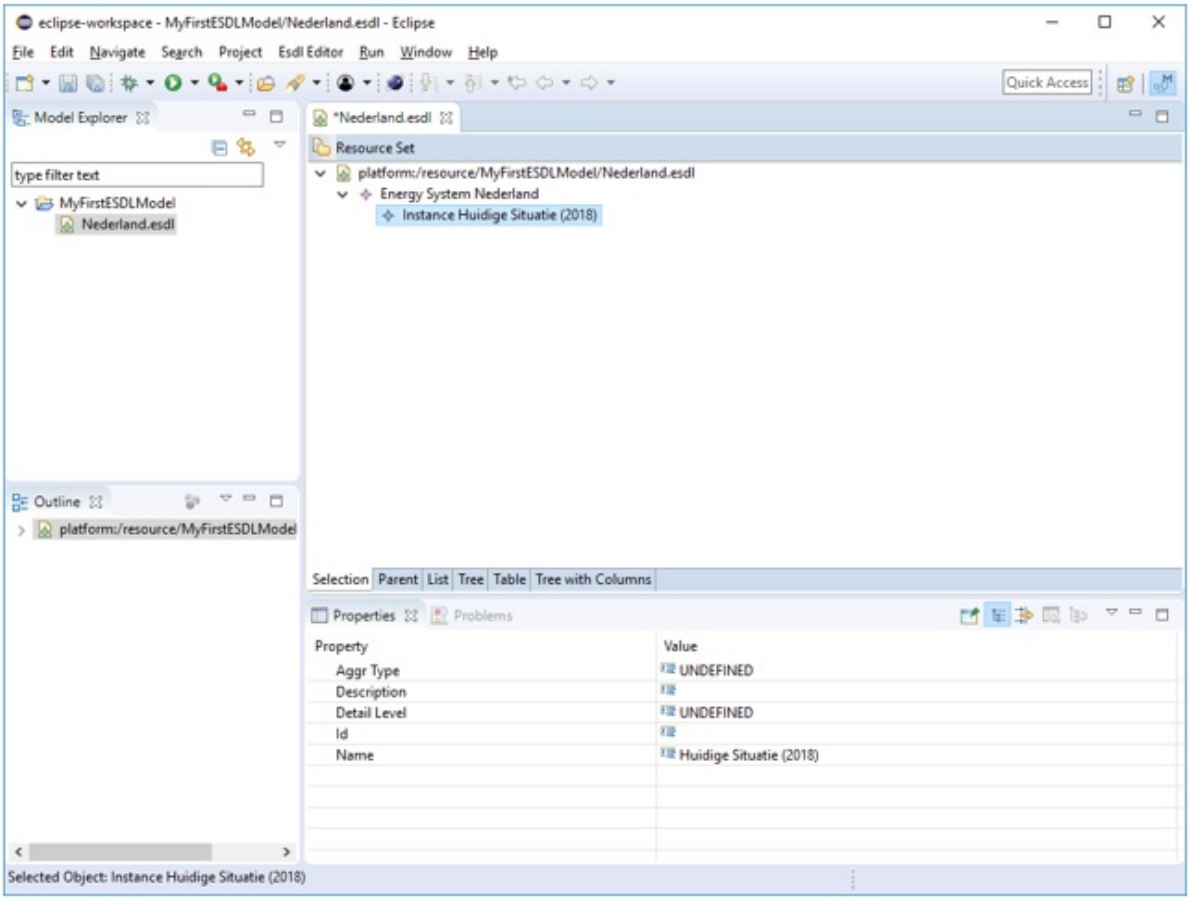


This opens de ESDL editor for your newly generated ESDL model:

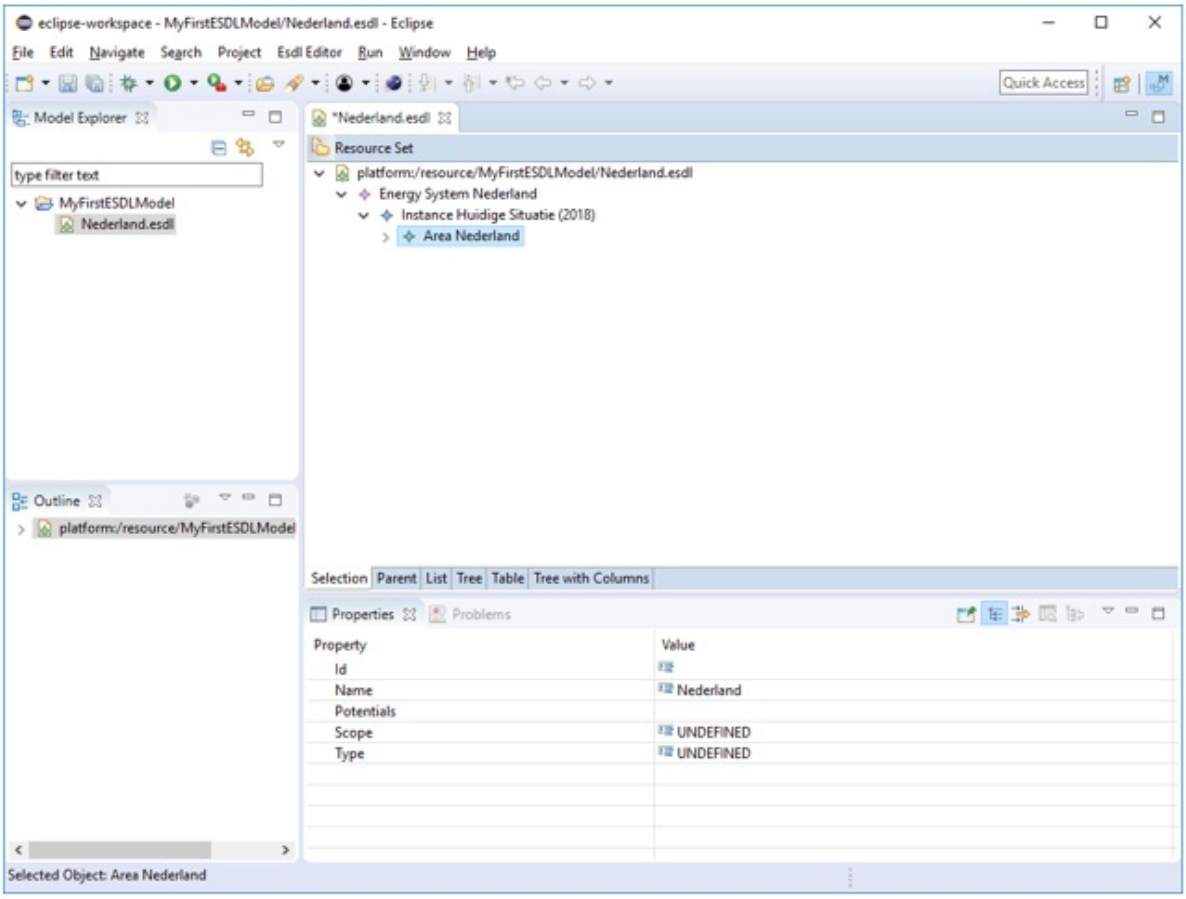


Step 4: Edit ESDL model

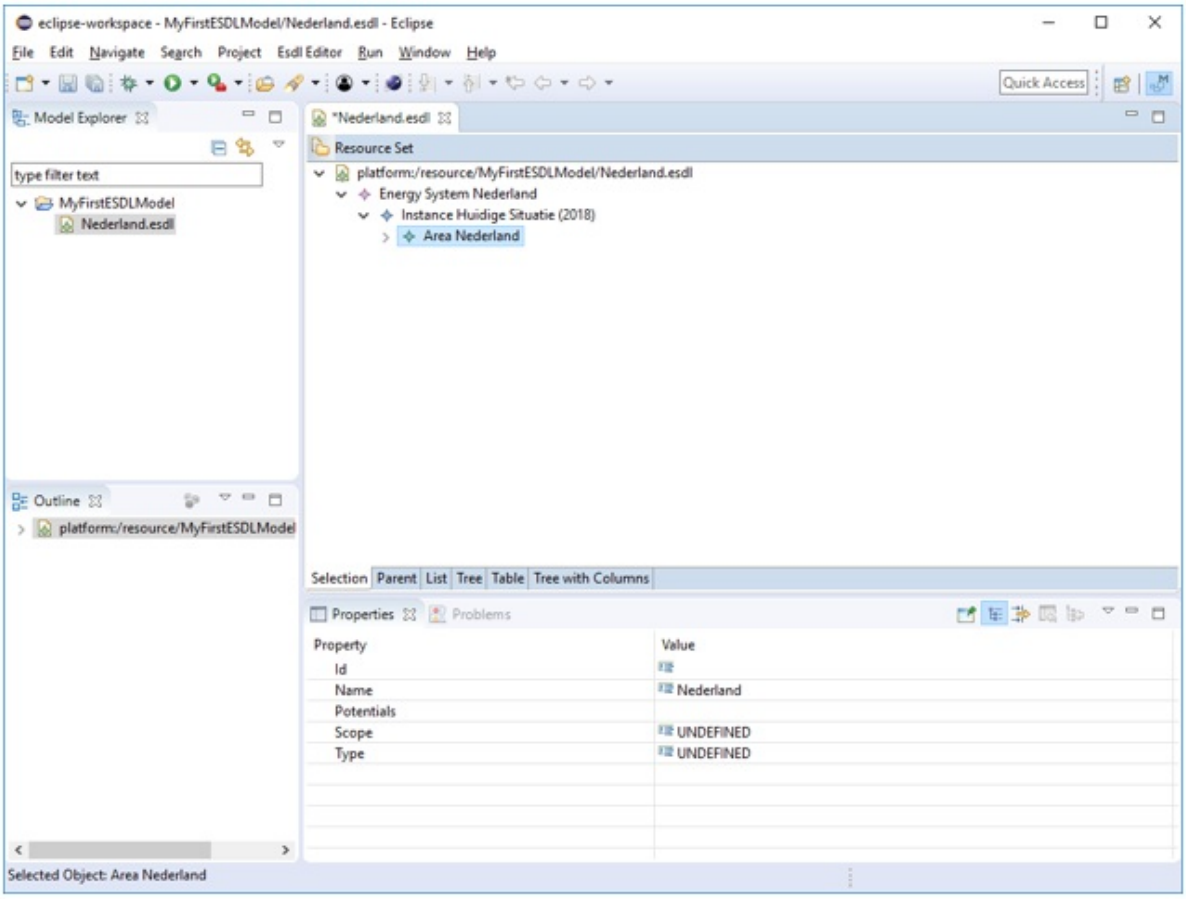
Right click on the Energy System and select “New Child” --> “Instance” and give the instance a name in the properties panel:



Add an area to the instance and give the area a name:



Add more areas or energy assets according to your needs… Start playing around!


eclipse-workspace - MyFirstESDLModel/Nederland.esdl - Eclipse
File Edit Navigate Search Project Esdl Editor Run Window Help
Quick Access
Model Explorer
type filter text
MyFirstESDLModel
Nederland.esdl
*Nederland.esdl
Resource Set
platform:/resource/MyFirstESDLModel/Nederland.esdl
Energy System Nederland
Instance Huidige Situatie (2018)
Area Nederland
Outline
platform:/resource/MyFirstESDLModel
Selection Parent List Tree Table Tree with Columns
Properties
Problems
Property
Value
Id
Name
Nederland
Potentials
Scope
UNDEFINED
Type
UNDEFINED
Selected Object: Area Nederland

ESDL graphical editor (TODO)

ESDL web-based viewer (TODO)

Using ESDL in models (TODO)

Generating and using the ESDL XSD

Code Generation (Java)

Code Generation (Python)

Introduction

Using the generateDS tool with the XSD generated from the ESDL model, you can generate python code to work with ESDL. The generated code contains all the python classes for each ESDL concept and functionality to instantiate a new model, add assets to it, read an ESDL description from disk or save one.

Generate the XSD from the ESDL model

The ESDL GitHub project contains the XML schema definition (XSD) file in the model subdirectory.

To generate a new XSD from an updated ESDL model, right click the `esdl.genmodel` file and select the 'Export model...' option from the drop down menu. In the wizard select the 'XML Schema' model exporter and follow the steps in the wizard to create the XSD.

Create Python classes using GenerateDS

The GenerateDS project provides us with a tool to generate Python data structures and an XML parser from an XML schema definition (XSD).

You can find the GenerateDS project here.

Using the following command, you can create the python code from the XSD file:

```
generateDS -o esdl.py .\esdl\EsdlXML.xsd
```

Using the generated Python code

TODO...

Examples (TODO)

Contact

See for contact information this site.