Nasdanika Codegen

Get Started

Overview

- Concepts
- Installation & Update
- Create a modeling project
- Create a model & select Codegen View Point
- Populate the model
- Generate model
  - IDE
  - CLI
  - Server

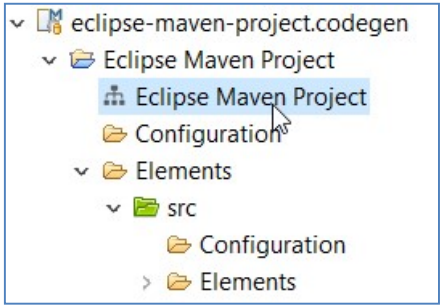
Concepts

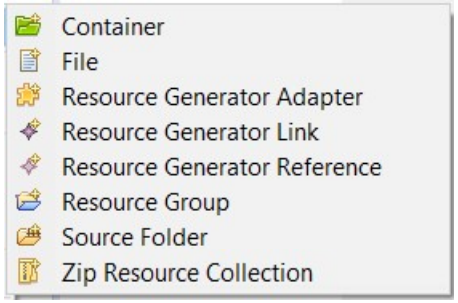
- Generates one or more resources – files, directories
- Input data * Generation Model => Generated resources
- Generation from
  - IDE
  - Command line
  - Server over HTTP
- Input data – Context
  - Properties – retrieved by name
  - Services – retrieved by type, e.g. `ImportManager.class`
  - Property & Service computers – compute data on retrieval
  - Can be constructed from:
    - *YAML*
    - *JSON*
    - *Properties files*
    - *Command line options*
    - *External sources – context builders*
- Interpolation
  - Replacement of `${<key>}` token with value from the context
  - Widely used in generator models
- Iteration – execution of generation zero or more times based on the type and value of the iterator context property

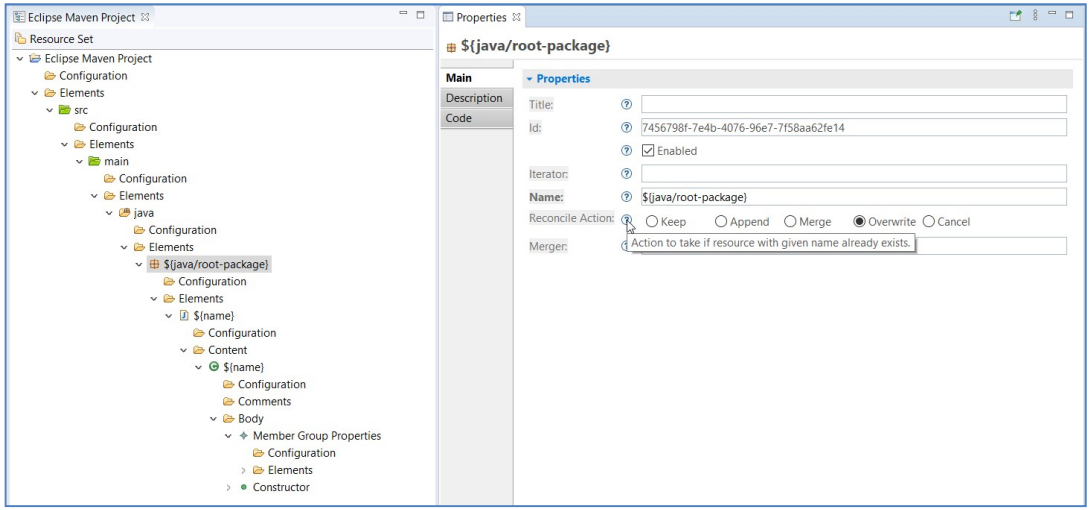
```
Usage: nsd codegen [-fhjV] [-b=<baseUri>] [-o=<outputDir>]
                   [-p=<progressOutput>] [-P=<parallelism>] [-t=<timeout>]
                   [-B=URL]... [-c=<String=String>]... [-C=URL]...
                   [-M=<String=String>]... [-r=URL]... URI
Generates resources from a codegen model
      URI                     Model (object) URI resolved relative to the
                                current directory. May include fragment to
                                address a non-root object
  -b, --base-uri=<baseUri>    Base URI for resolving and relativizing other
                                URI's. Resolved against the output directory
                                URI. Defaults to the output directory URI.
  -B, --context-builders=URL Context builders configuration resource URL
                                relative to the current directory.  See online
                                documentation at https://www.nasdanika.
                                org/builds/develop/doc/reference/cli/context-buil
                                ders.html for details.
  -c, --context-entry=<String=String>
                              Context entries.
                              Shadow entries in contexts and mounts.
  -C, --context=URL           Context resource URL relative to the current
                                directory. YAML, JSON, or properties. In
                                properties dots are treated as key path
                                separators. Type is inferred from the content
                                type header, if it is present, or extension.
                                Contexts are composed in the order of
                                definition, later context entries shadowing the
                                former
  -f, --file                  URI is a file path
  -h, --help                  Show this help message and exit.
  -j, --json                  Output progress in JSON
  -M, --context-mount=<String=String>
                              MappingContext resource URL relative to the
                                current directory. YAML, JSON, or properties. In
                                properties dots are treated as key path
                                separators. Type is inferred from the content
                                type header, if it is present, or extension.
                                Mounts shadow context entries.
  -o, --output=<outputDir>    Output directory, defaults to the current directory
  -p, --progress=<progressOutput>
                              Output file for progress monitor
  -P, --parallelism=<parallelism>
                              If the value greater than one then an executor
                                service is created and injected into the context
                                to allow concurrent execution.
  -r, --resource=URL          URL of a resource to load to the resource set
                                relative to the current directory.
  -t, --timeout=<timeout>     If parallelism is greater than one this option
                                specifies timout in seconds awaiting completion
                                of execution. Default value is 60.
  -V, --version               Print version information and exit.
Exit codes:
  0   Success
  1   Unhandled exception during execution
  2   Invalid input
  3   Diagnostic failed
  4   Execution failed or was cancelled, successful rollback
  5   Execution failed or was cancelled, rollback was not successful
  6   Executor service termination timed out
```

Generation Model

- A tree of resource generators and their contents generators
- Multiple representations per model
- Models can reference other models
- Resource generators:
  - Container – directory/folder
  - File
  - Adapter – calls Java extension to generate resources
  - Link – loads resource generator from URL,
    - *Reuse*
    - *Polymorphic generation with interpolation – link different models based on input*
  - Reference – points to a generator, e.g. a generator from another model
    - *Reuse*
  - Group – a construct for configuration and iteration
  - Zip resource collection – loads resources from a zip stream, which can be loaded by, say, HTTP Call
    - *Leveraging external generators*
- Reconciliation with pre-existing resources
  - Several options, default is Overwrite
  - Optional support for merging
    - *Ability to change generator input, re-generate and preserve manual modifications*
    - *Concurrent evolution of a generator and code generated by it – improve both at the same time and then combine (merge).*

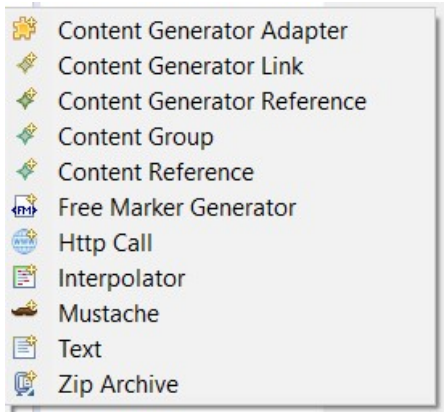


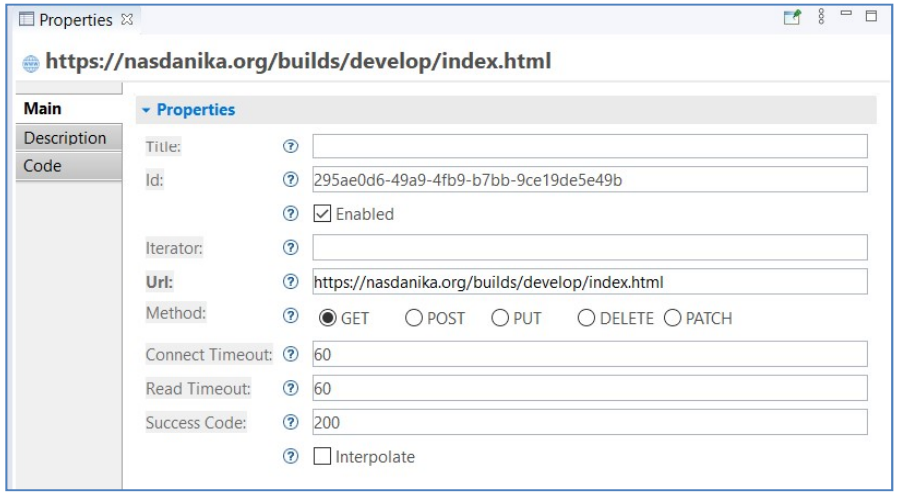


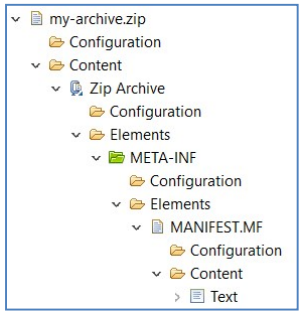


Content Generators

- Adapter – calls Java extension to generate content
- Link – loads content generator from URL
  - Reuse
  - Polymorphic generation with interpolation
- Generator Reference – points to a generator, e.g. a generator from another model
  - Reuse
- Group – a construct for configuration and iteration
- Content reference – loads content from a URL
- Free Marker Generator – uses Apache Free Marker
- HTTP Call
  - HTTP methods
  - Headers
  - Body
  - Response as content, can be interpolated
- Interpolator – expands `${<key>}` tokens.
- Mustache – evaluates Mustache template
- Text – static text with support of interpolation
- Zip archive – constructs Zip stream from contained resources

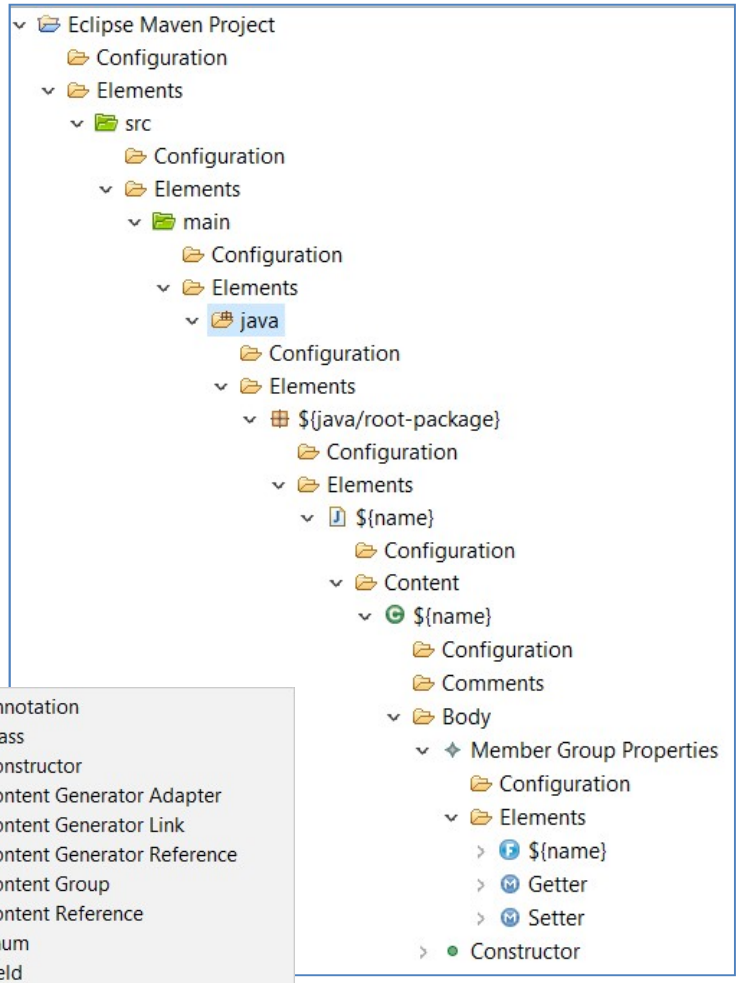


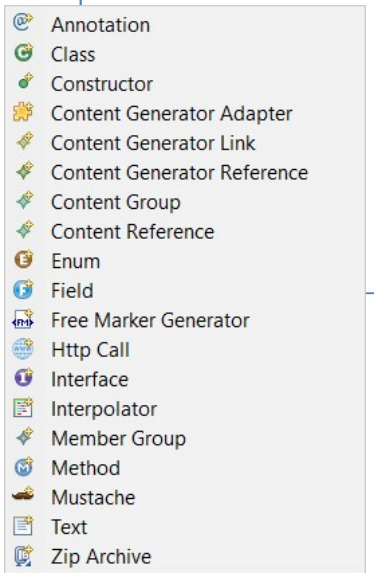




Java Support

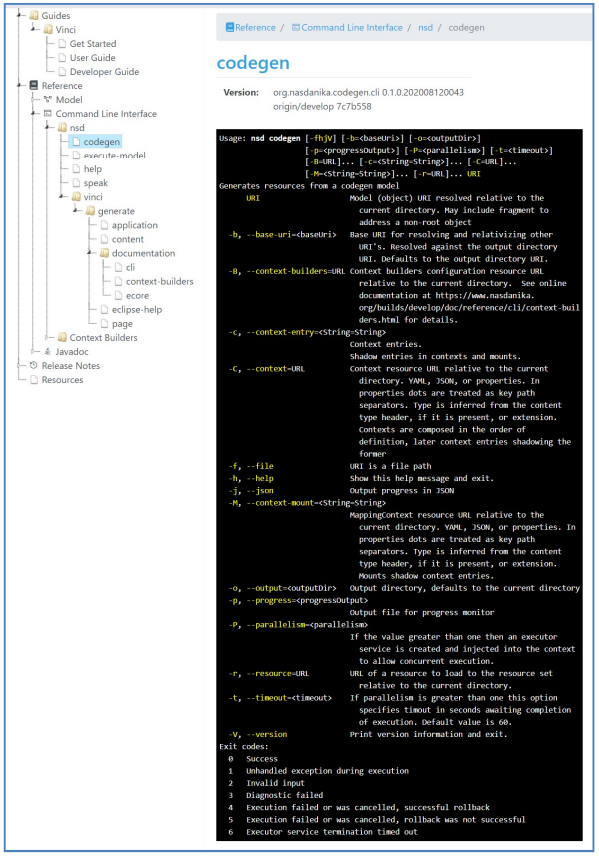
- Resource generators:
  - Source folder – a specialized container which can contain packages
  - Package – a specialized container
    - *Can contain Compilation Units*
    - *Uses Java naming for packages – dot-separated*
  - Compilation Unit – a specialized file
    - *Generates package declaration*
    - *Import manager*
      - Use `${import/<fully qualified type name>}` to replace FQN with short name
      - Generates imports after the package declaration
    - *Can contain types – Annotations, Classes, Enums, Interfaces*
    - *Source formatting*
    - *Native merger (JMerger from the EMF code generator)*
      - Regenerates only members with a "clean" `@generated` Javadoc tag – no value
      - Keeps members without the `@generated` tag or with a "dirty" tag, e.g. `@generated NOT`
- Content generators:
  - Types – specialized content generators. Some can contain members.
  - Members – types, fields, methods, and constructors

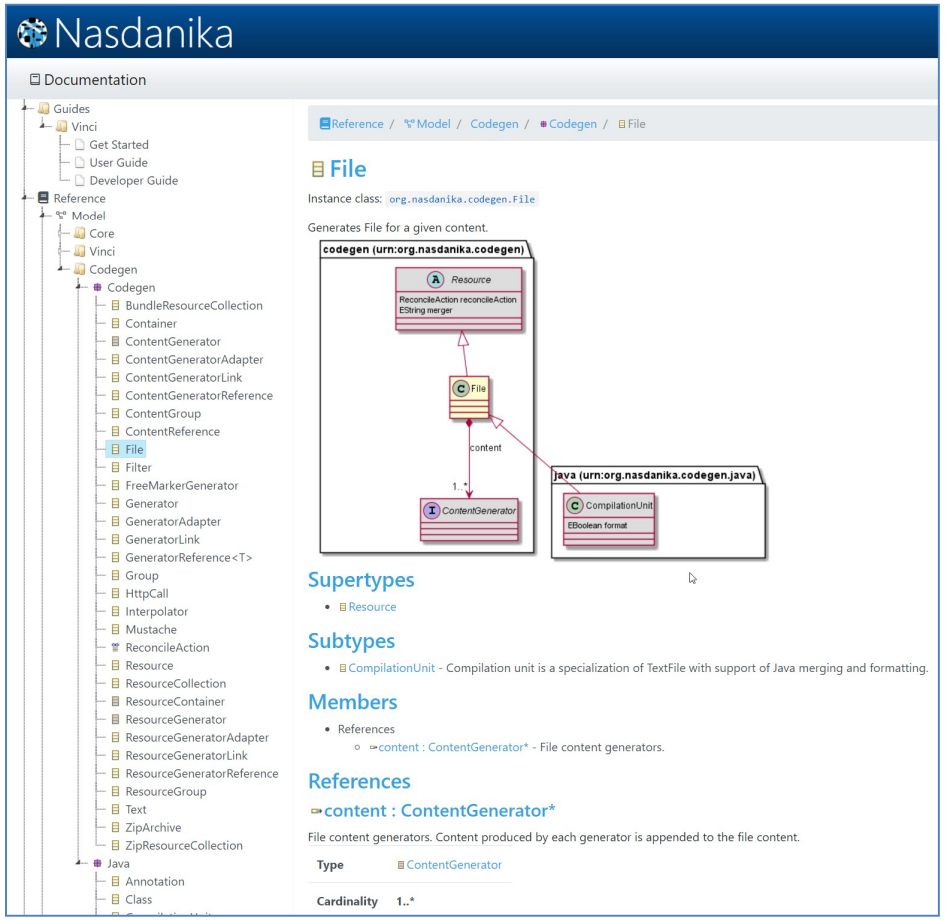




Documentation

- Online
- Integrated Eclipse Help
- Model reference
  - The primary source of information on how to use model elements
  - First sentence of attribute documentation is shown as a tooltip in properties view
- Command line reference



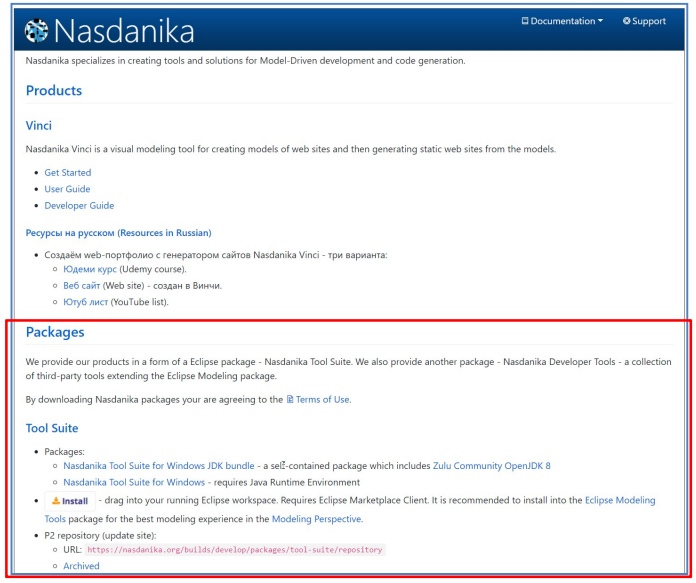


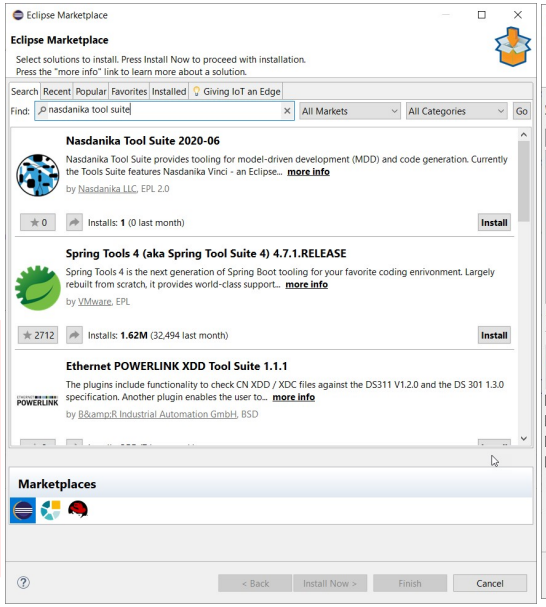
Installation prerequisites

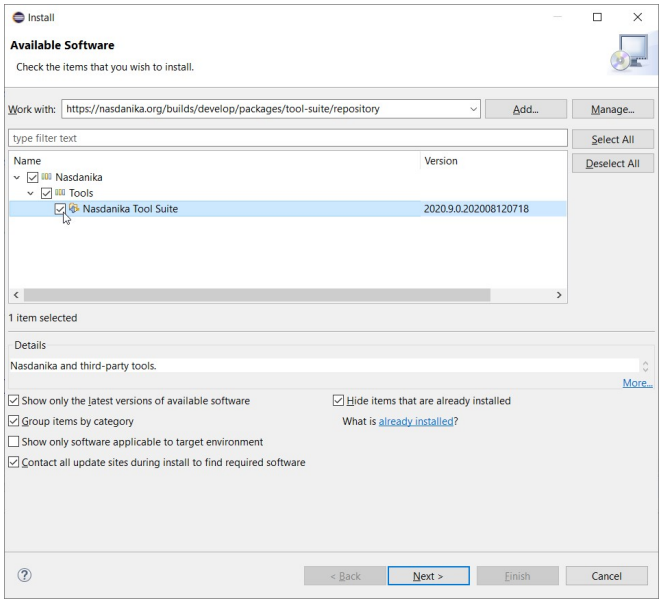
- Microsoft Windows 10 or 7
- Permissions
  - For binary packages:
    - *Download zip files*
    - *Extract zip files*
    - *Launch executables*
  - For installing into an existing Eclipse:
    - *Eclipse installation, Eclipse Modeling package is recommended.*
    - *Access to internet from Eclipse to install from the Eclipse Marketplace or Install New Software*
    - *Or download zip files (p2 archive)*

Installation

- Downloadable packages:
  - Nasdanika Tools Suite for Windows – based on Eclipse modeling 2020-03
  - Nasdanika Tools Suite for Windows & Open JDK 8 bundle
- Installing into an existing Eclipse:
  - Eclipse Marketplace
  - P2 repository (update site),
  - Archived P2 repository
    - *Download and install from archive*
    - *Team environment - extract and host on the intranet*

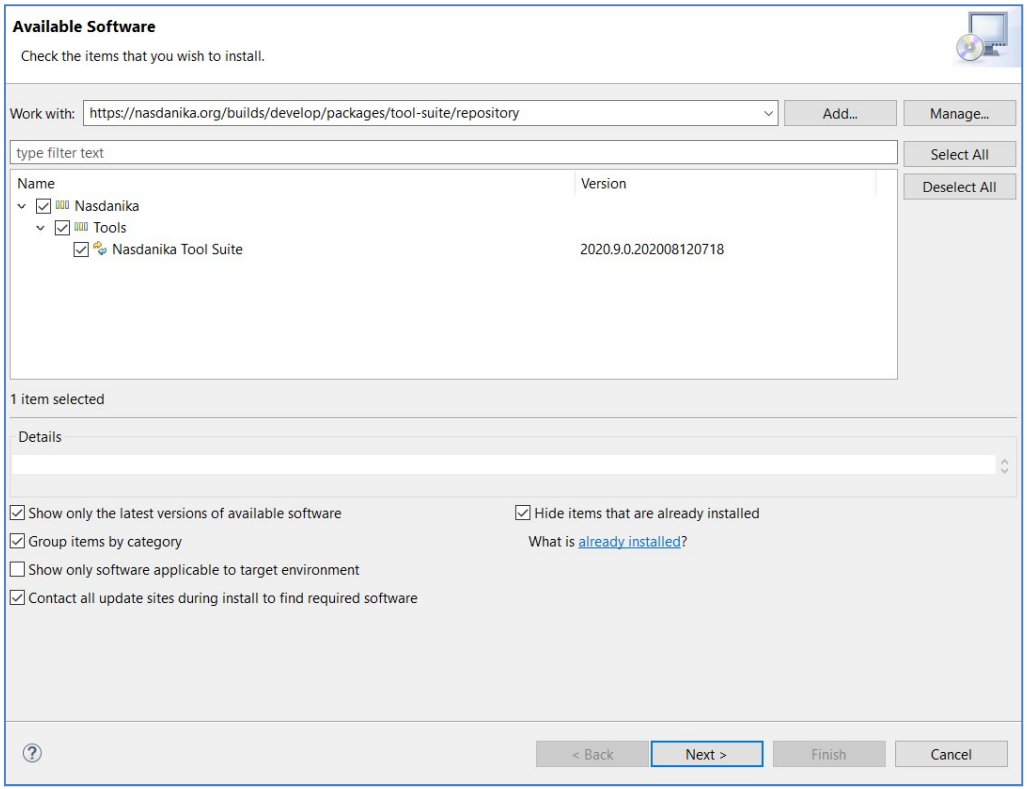






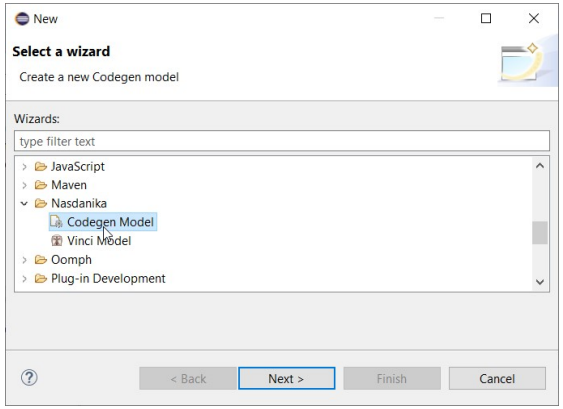
Updating

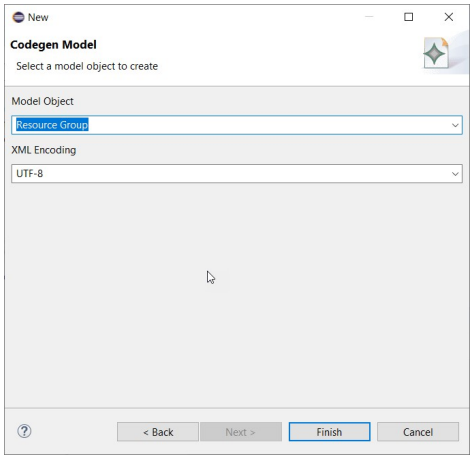
- First time
  - Help > Install new software
  - Work with – P2 site URL
- On-going – Help > Check for Updates

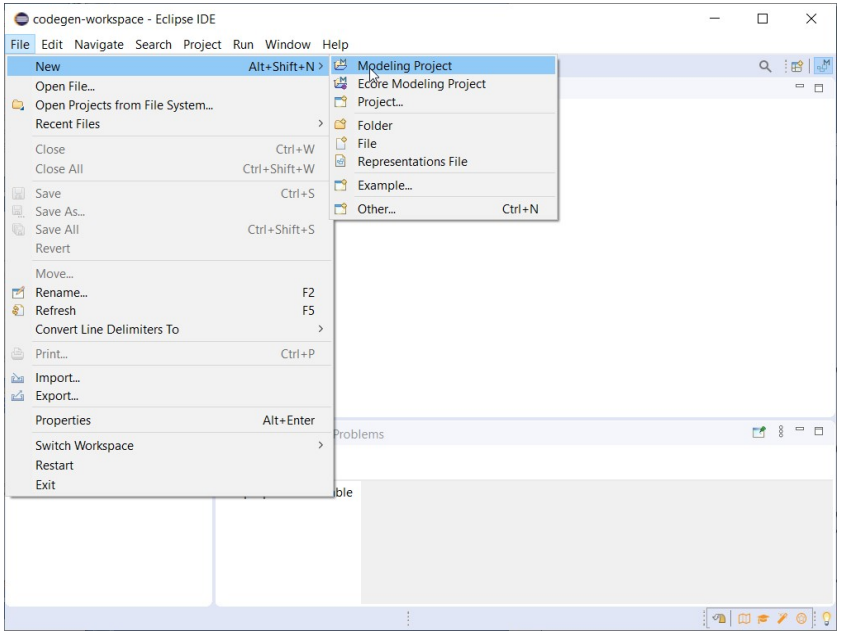


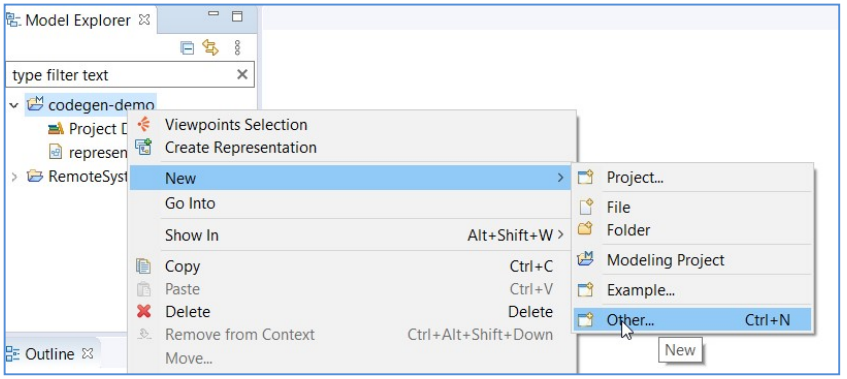
Create a modeling project and a model

- File > New > Modeling Project
- File > New > Other
- Nasdanika > Codegen Model
- Select Model Object – Resource Group

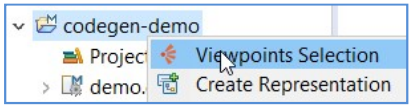


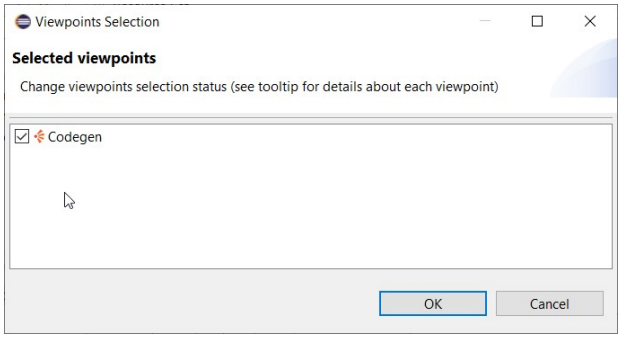






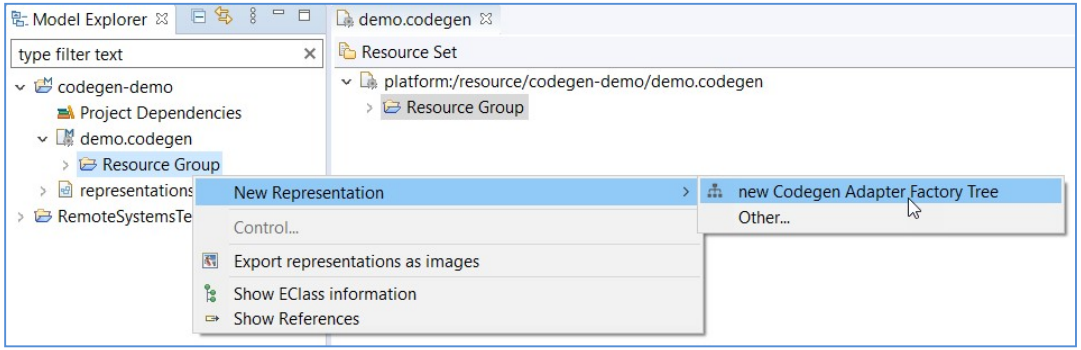
- Select Codegen Viewpoint

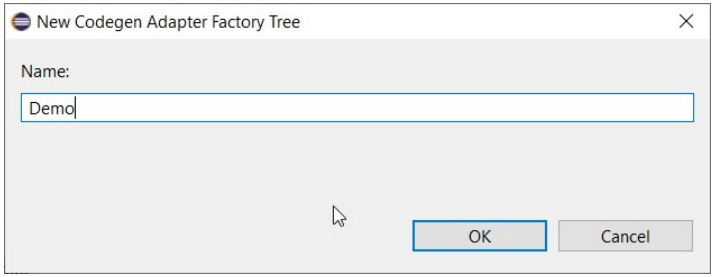


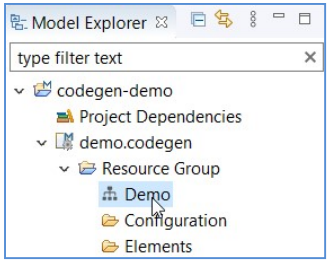


Modeling – create representations

- By default the generated editor is used
- It is recommended to use representations
  - Expand the model in the Model Explorer
  - Right-click on a model element of interest
  - New Representation > new Codegen Adapter Factory Tree
  - Name the representation
  - One model can have multiple representations
  - Opens a representation editor once created
  - Going forward double-click on a representation to open it



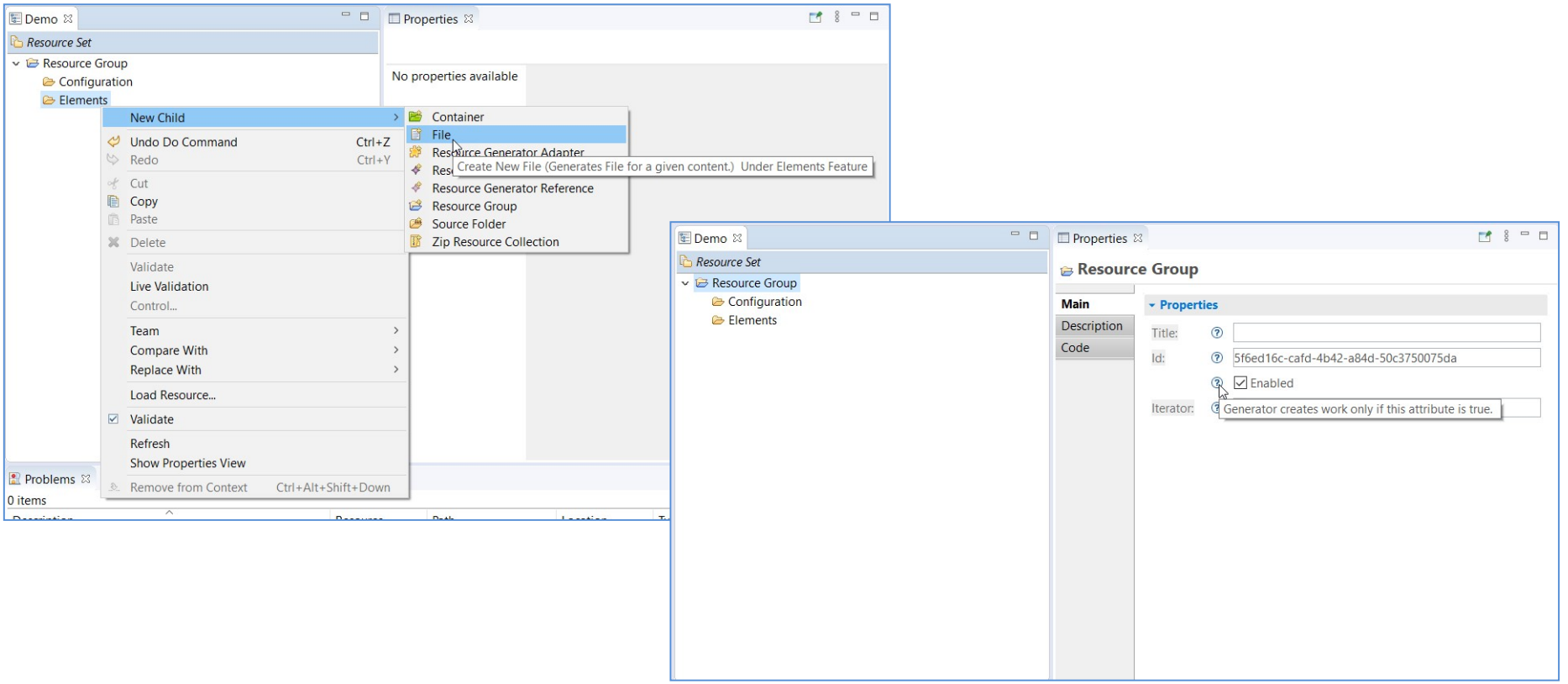




Important – do not edit properties of elements selected in the Model Explorer. These edits might be lost. Select elements in a representation editor.

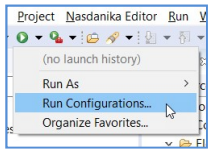
Modeling – populate the model

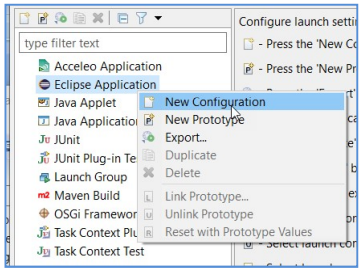
- Right click on model elements in the representation editor to add child elements
- Fill-out properties in the property view
- Ctrl-C/Ctrl-V to copy
- Drag-n-Drop to move from one element to another
- Right-Click > Load resource to reference model elements from external models

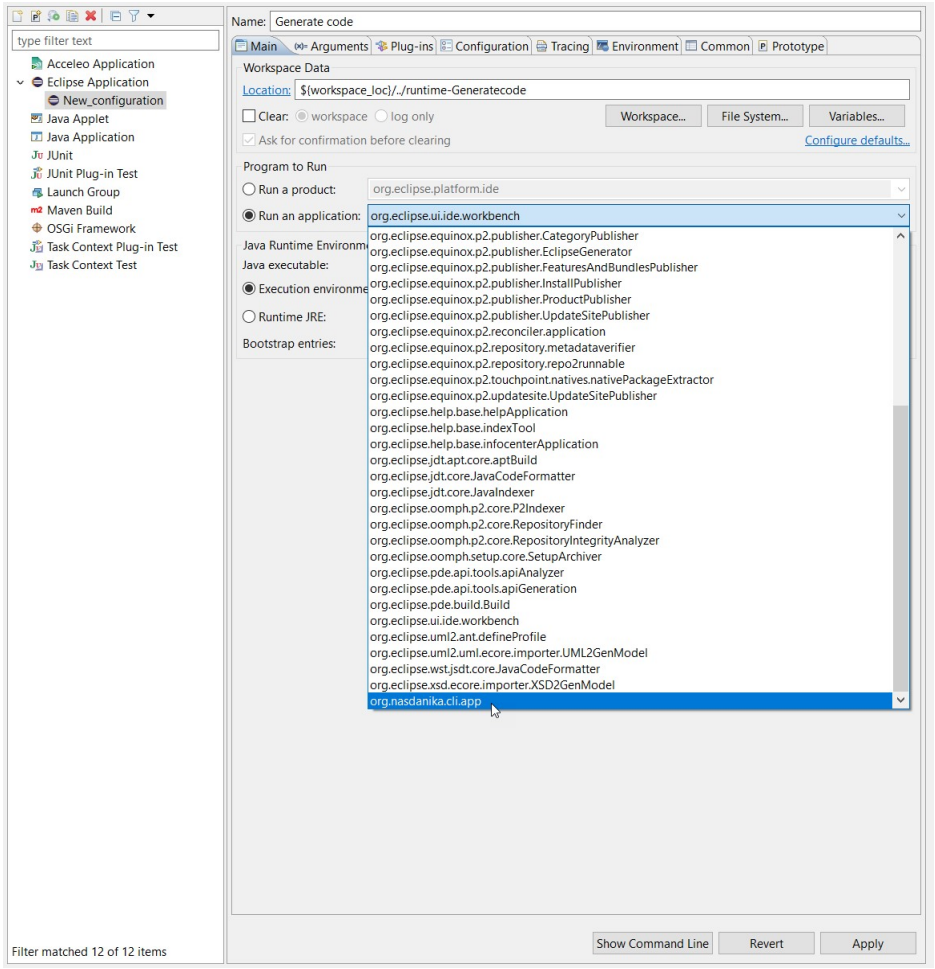


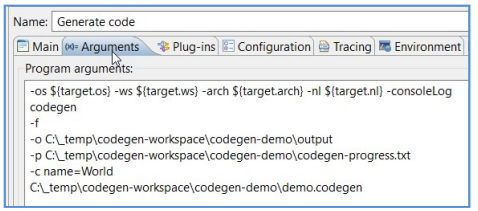
Generation – IDE – Option 1: Eclipse Application

- Create a new Run configuration:
  - Eclipse application
  - Run an application: org.nasdanika.cli.app
  - Arguments according to the CLI documentation



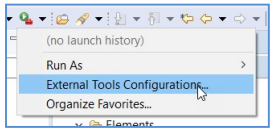


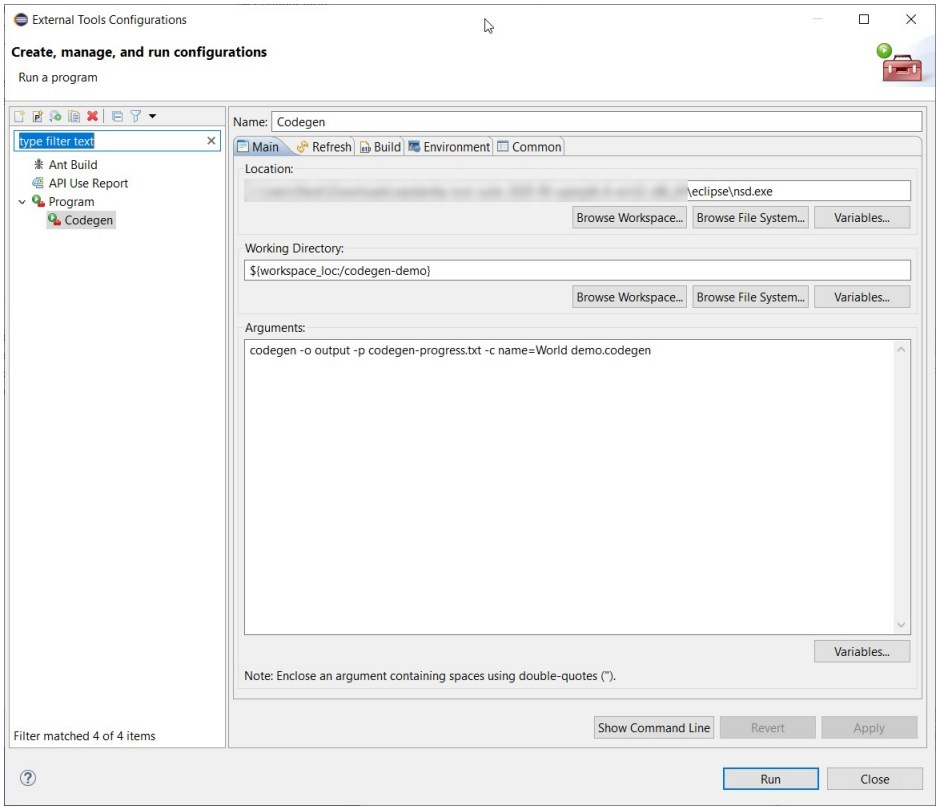




Generation – IDE – Option 2: External tool

- Create a new External tool configuration:
  - Program
  - Location – point to nsd.exe
  - Working directory – at your convenience,
    - *E.g. modeling project directory – shorter command line*
  - Arguments according to the CLI documentation





Generation – output

- Look for the exit code – 0 is success
- Ignore net4j exception stack trace
- Explore progress output if needed to troubleshoot – very verbose

Problems | Console

<terminated> Codegen [Program] [illegible] eclipse\nsd.exe [illegible] – 1

```
Exit code: 0
[WARN] Problem while deactivating CDOViewProviderRegistryImpl
java.lang.InterruptedException
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(AbstractQueuedSynchronizer.
        at java.util.concurrent.Semaphore.acquire(Semaphore.java:312)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.lock(Lifecycle.java:310)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.internalDeactivate(Lifecycle.java:118)
        at org.eclipse.net4j.util.lifecycle.ShareableLifecycle.internalDeactivate(ShareableLifecycle.java:52)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.deactivate(Lifecycle.java:168)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:235)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:225)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:251)
        at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.doStop(Activator.java:129)
        at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.stop(Activator.java:99)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:899)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:1)
        at java.security.AccessController.doPrivileged(Native Method)
        at org.eclipse.osgi.internal.framework.BundleContextImpl.stop(BundleContextImpl.java:891)
        at org.eclipse.osgi.internal.framework.EquinoxBundle.stopWorker0(EquinoxBundle.java:1029)
        at org.eclipse.osgi.internal.framework.EquinoxBundle$EquinoxModule.stopWorker(EquinoxBundle.java:370)
        at org.eclipse.osgi.container.Module.doStop(Module.java:658)
```

Generation – CLI

- Remove `-Dpicocli.ansi=true` from `nsd.ini` if garbled output instead of color highlighting – no ANSI support
- Look for the exit code – 0 is success
- Ignore net4j stack trace

```
C:\Windows\system32\cmd.exe

C:\_temp\codegen-workspace\codegen-demo>F:\eclipse\nsd codegen -o output -p codegen-progress.txt -c name=World demo.codegen
Exit code: 0
[WARN] Problem while deactivating CDOViewProviderRegistryImpl
java.lang.InterruptedException
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(AbstractQueuedSynchronizer.java:1306)
        at java.util.concurrent.Semaphore.acquire(Semaphore.java:312)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.lock(Lifecycle.java:310)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.internalDeactivate(Lifecycle.java:118)
        at org.eclipse.net4j.util.lifecycle.ShareableLifecycle.internalDeactivate(ShareableLifecycle.java:52)
        at org.eclipse.net4j.util.lifecycle.Lifecycle.deactivate(Lifecycle.java:168)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:235)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:225)
        at org.eclipse.net4j.util.lifecycle.LifecycleUtil.deactivate(LifecycleUtil.java:251)
        at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.doStop(Activator.java:129)
        at org.eclipse.emf.internal.cdo.bundle.Activator$Implementation.stop(Activator.java:99)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:899)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$4.run(BundleContextImpl.java:1)
        at java.security.AccessController.doPrivileged(Native Method)
        at org.eclipse.osgi.internal.framework.BundleContextImpl.stop(BundleContextImpl.java:891)
        at org.eclipse.osgi.internal.framework.EquinoxBundle.stopWorker0(EquinoxBundle.java:1029)
        at org.eclipse.osgi.internal.framework.EquinoxBundle$EquinoxModule.stopWorker(EquinoxBundle.java:370)
        at org.eclipse.osgi.container.Module.doStop(Module.java:658)
        at org.eclipse.osgi.container.Module.stop(Module.java:520)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.decStartLevel(ModuleContainer.java:1885)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.doContainerStartLevel(ModuleContainer.java:1760)
        at org.eclipse.osgi.container.SystemModule.stopWorker(SystemModule.java:275)
        at org.eclipse.osgi.internal.framework.EquinoxBundle$SystemBundle$EquinoxSystemModule.stopWorker(EquinoxBundle.java:202)
        at org.eclipse.osgi.container.Module.doStop(Module.java:658)
        at org.eclipse.osgi.container.Module.stop(Module.java:520)
        at org.eclipse.osgi.container.SystemModule.stop(SystemModule.java:207)
        at org.eclipse.osgi.internal.framework.EquinoxBundle$SystemBundle$EquinoxSystemModule$1.run(EquinoxBundle.java:220)
        at java.lang.Thread.run(Thread.java:748)

C:\_temp\codegen-workspace\codegen-demo>
```

Server Mode

- Adjust `server.ini` if needed
  - Port number
  - Context path
  - Base URL for models, defaults to the current directory
- Start `server.exe`
- HTTP POST
  - Input data as payload – YAML or JSON
  - Zip archive as response
    - *`output` directory contains generation results*
    - *`progress.txt` file contains progress information*
- TODO – CURL demo

Install Server as Windows Service

- TODO

Modeling Process

- Manually create an example of desired generation output – a reference implementation
- Analyze resources in the reference implementation to identify variation points – things which may change between generations, e.g:
  - Maven coordinates – group and artifact ID's
  - Project name
  - Project description
  - Root Java package
- Identify needed generation inputs and how variation points values will be derived from them, e.g. root Java package may be derived from Maven group ID and artifact ID
- Create sample input or inputs, e.g. a YAML file
- Create model elements, generate with provided input, compare with the reference implementation

Example

- Iteration over `java/beans` and then `fields`