nsx
NORMALIZED SYSTEMS

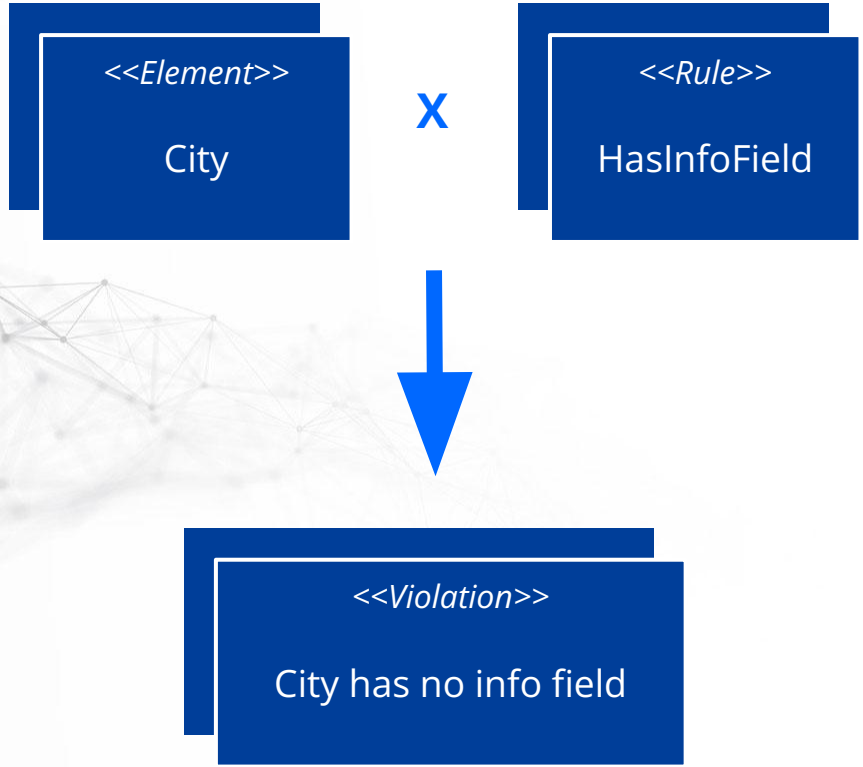
2022-01-26

Overview R&D 2022

Koen De Cock & Frédéric Hannes

1. Validations
2. Transmutations
3. ModelLoadingListeners
4. Expanders
5. OptionTypes
6. TestModelBuilder
7. Metamodels
8. µRadiant
9. NS Scripting
10. NS Initializer
11. Docker Images
12. Product Development
13. New Foundation

Validations



Validations

Integrated in µRadiant

zooApp::1.0.0 4 | animals 1 | + | settings

Filter el

Right-click to interact

- animals 1
  - dataElemer
    - Animal
      - fields
        - + nai
        - + spe
    - Species

edit | neighbors (in 0, out 1) | validations 1

animals::Animal *DataElement*

DataElementHasInfoFieldRule

Missing Defaults

A data element should have at least one info field. Without any info fields, no data will be visible in the UI tables.

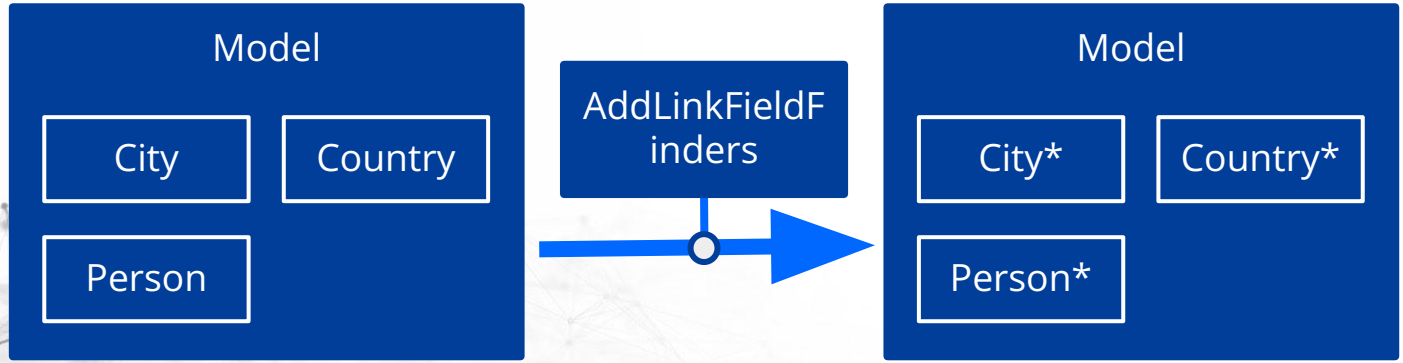
<DataElement animals::Animal> has no isInfoField field

Validations

Custom Rules

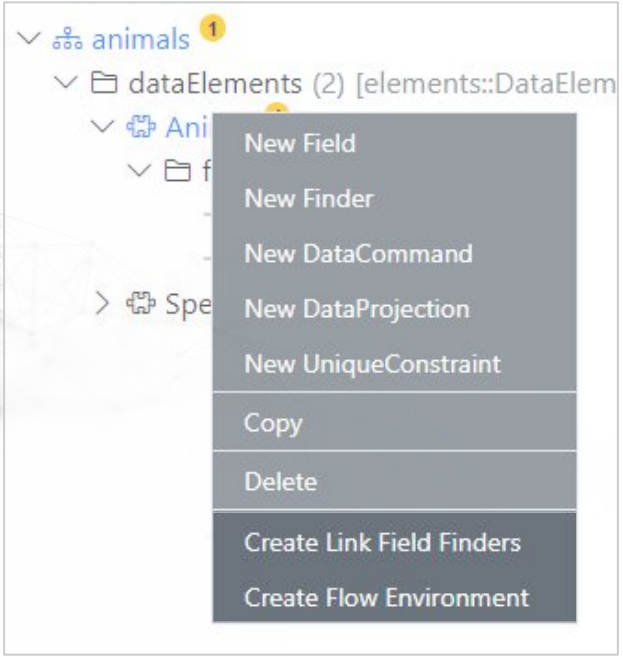
```
@ValidationRule(
   element = "elements::Field",
   severity = ValidationSeverity.SUGGESTION,
   reason = "Field names should start with lowercase letters.",
   description = "The first two letters of a Field's name should be lowercase characters."
)
@ValidationGroup("Naming")
public class FieldNamingRule implements ValidatableRule<FieldComposite> {
 public void validate(FieldComposite field, Critiques critiques) {
   String name = field.getName();
   if (name.length() == 0 || !Character.isLowerCase(name.charAt(0))) {
     critiques.add(field, "{element} name must start with lowercase letter");
   }
   if (name.length() >= 2 && Character.isUpperCase(name.charAt(1))) {
     critiques.add(field, "{element} name's second character must be lowercase");
   }
   if (name.contains("_")) {
     critiques.add(field, "{element} name should not contain underscores");
   }
 }
}
```

Transmutations



Transmutations

Integrated in µRadiant



Transmutations

Custom Transmuter

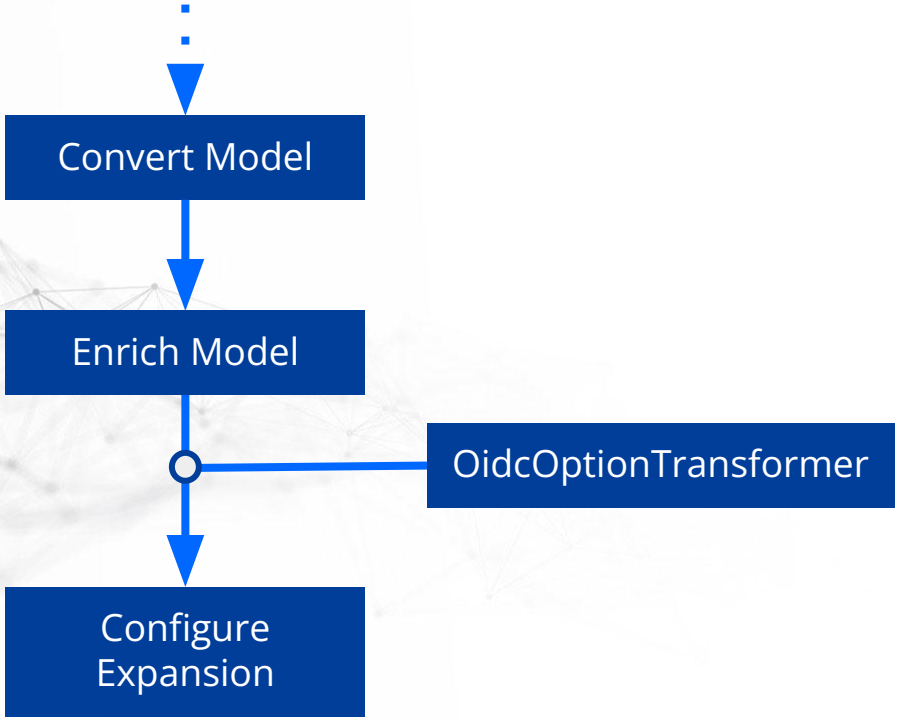
```
@Transmutation(
   element = "elements::DataElement",
   description = "Adds a FlowElement and status field to the target DataElement"
)
public class CreateFlowEnvironment implements ModelTransmuter<DataElementComposite> {

 @Override
 public void transmute(DataElementComposite dataElement, ModelTransmutationContext context) {
   ComponentCompositeBuilder.merge(dataElement.getComponent(), context, comp -> {
     comp.flowElements(flowElement -> {
       flowElement.name(dataElement.getName() + "Flow");
       flowElement.packageName(dataElement.getPackageName());
       flowElement.statusField("status");
       flowElement.targetClass(dataElement.getPackageName() + "." + dataElement.getName());
     });
   });
 }
}
```

ModelLoader

1. Read Expansion Settings
2. Resolve Resources
3. Extract Resources
4. Read Model
5. Convert Model
6. Enrich Model
7. Configure Expansion

ModelLoader

ModelLoadingListeners



ModelLoadingListeners

```
<modelLoadingListeners>
 <modelLoadingListener name="net.democritus.expander.OidcOptionTransformer">
   <modelLoadingStep name="EnrichModelStep"/>
   <implementation>net.democritus.expander.OidcOptionTransformer</implementation>
 </modelLoadingListener>
</modelLoadingListeners>
```

```
public class OidcOptionTransformer implements ModelLoadingStepListener<EnrichModelResult> {

  @Override
  public void afterStep(EnrichModelResult result, ModelLoadingContext context) {
    //…
  }
}
```

Expanders

Cleanup

- Vector 💀
- new Long(String) 💀
- Long.longValue() 💀
- Java 7 💀 → Java 17 (Java 21 LTS in september 2023)
- new Long(0) 💀
- Legacy methods 💀
- EngineService#collector 💀
- *target.element.projection* / *target.class.custom* replaces targetClass 💀

Expanders

Improvements

- Cached Workflow Configuration
- DetailsWithoutRefs Projection
- StateTask timeout
- *Run now* button for workflow
- Finder *in* operator
- ContextRetriever allows Context updates in Control Layer
- *myComp.ns.properties* file

Expanders

Improved Model Loading Errors

```
net.democritus.elements.ElementNotFoundException: Cannot find Field(registration::Person::Status)

    at FlowElementTreeToComposite.getStatusField (statusField=Status, targetElement=<DataElement registration::Person>)
    at FlowElementTreeToComposite.mapTree (tree=<FlowElement registration::PersonFlow>)
    at ComponentTreeToComposite.mapFlowElement (flowElement=<FlowElement registration::PersonFlow>)
    at ComponentTreeToComposite.mapTree (tree=<Component registration>)
    at ModuleCompositeModelLoader.loadModules (moduleType=<ModuleType elements::JeeComponent>)
    at ExpansionCompositeModelConverter.convert (program=<Application demo::1.0.0>, programType=<ProgramType elements::JeeApplication>)
    at ModelLoader.performStep (step=ConvertModelStep)
    at ModelLoader.loadModel (expansionSettings=F:\NSF\workspace\demo\conf\expansionSettings.xml)
```

Expanders

Unique Constraints

µRadiant

demo reload

demo::1.0.0 2 | cars 2 | demoComp 1 | + | settings

Filter elements...

Right-click to interact

- cars 2
  - dataElements (3) [elements::DataElement]
    - Car 1
      - fields (4) [elements::Field]
        - licensePlate: String
        - carBrand: CarBrand[Ln01]
        - notes: StringLong
        - status: String
      - finders (2) [elements::Finder]
        - findByCarBrandEq
        - findByStatusSe
      - uniqueConstraints (1) [elements::UniqueConstraint]
        - licensePlate
    - CarBrand

edit | validations

74f7437e-0a26-46c7-9fd3-737f2c163bee *UniqueConstraint*

fields

- [x] cars::Car::licensePlate
- [ ] cars::Car::carBrand
- [ ] cars::Car::notes
- [ ] cars::Car::status

options

Add option | Type option type name...

OptionTypes

Define OptionTypes in expansion-resources

- Enumerate valid ElementTypes
- Value (no value/required/regex)
- Cascade
- Alias
- Deprecate (with validUntil)
- Always enabled (with default value)

```
<optionType name="ejb3.useEjbInControl">
 <description>
    Adds ejb3.1 as a technology
    to the control layer together with
    a dependency on `javax:javaee-api`.
 </description>
 <elementTypes>
   <elementType component="elements"
                name="Application"/>
   <elementType component="elements"
                name="ApplicationInstance"/>
   <elementType component="elements"
                name="Component"/>
 </elementTypes>
 <valueConstraint>
   <noValue>true</noValue>
 </valueConstraint>
</optionType>
```

TestModelBuilder

Replaces ModelSpecBuilder to add integration with ModelLoader

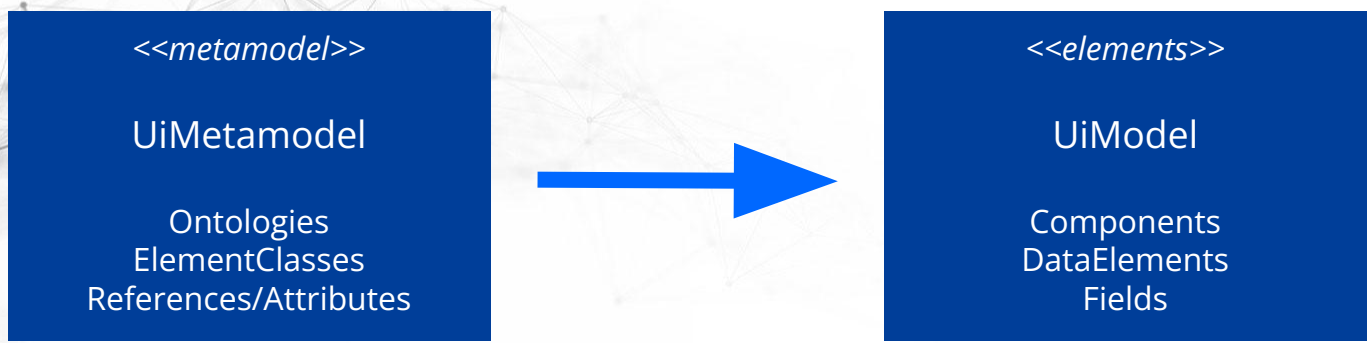
- Includes ModelLoadingListeners
- Supports OptionType cascade, alias ...

```
ModelSpecBuilder specBuilder=
    new ModelSpecBuilder();
return specBuilder.buildAndFind(
   component("testComp",
       dataElement("City")),
   dataElement("testComp::City"));
```

```
TestModelBuilder modelBuilder=
    new TestModelBuilder();
return modelBuilder.buildModelAndFind(
   component("testComp",
       dataElement("City")),
   dataElement("testComp::City"));
```

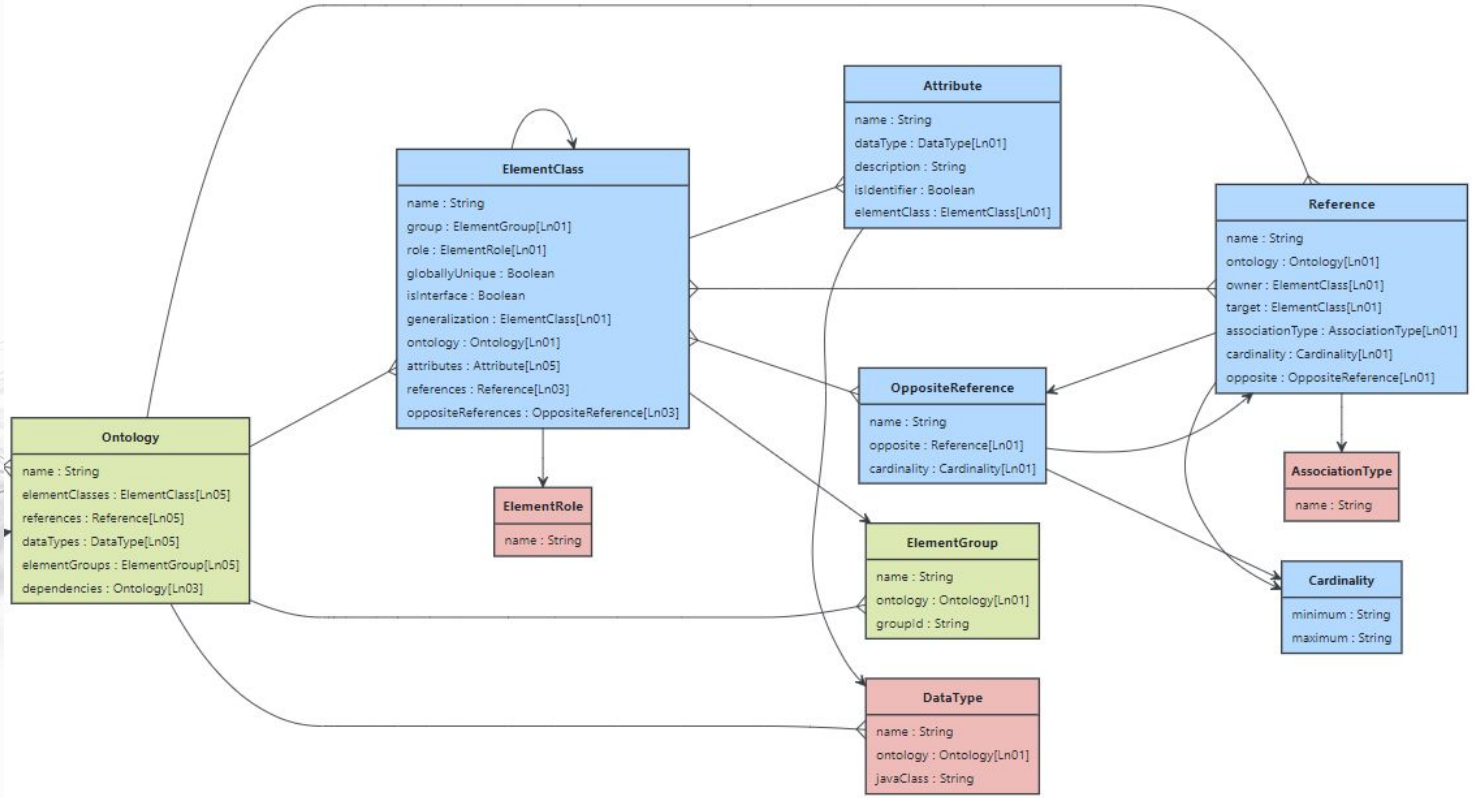
Metamodels

- Designed for creating metamodels
- Can better represent bi-directional Relationships
- Supports inheritance with Polymorphic Interfaces

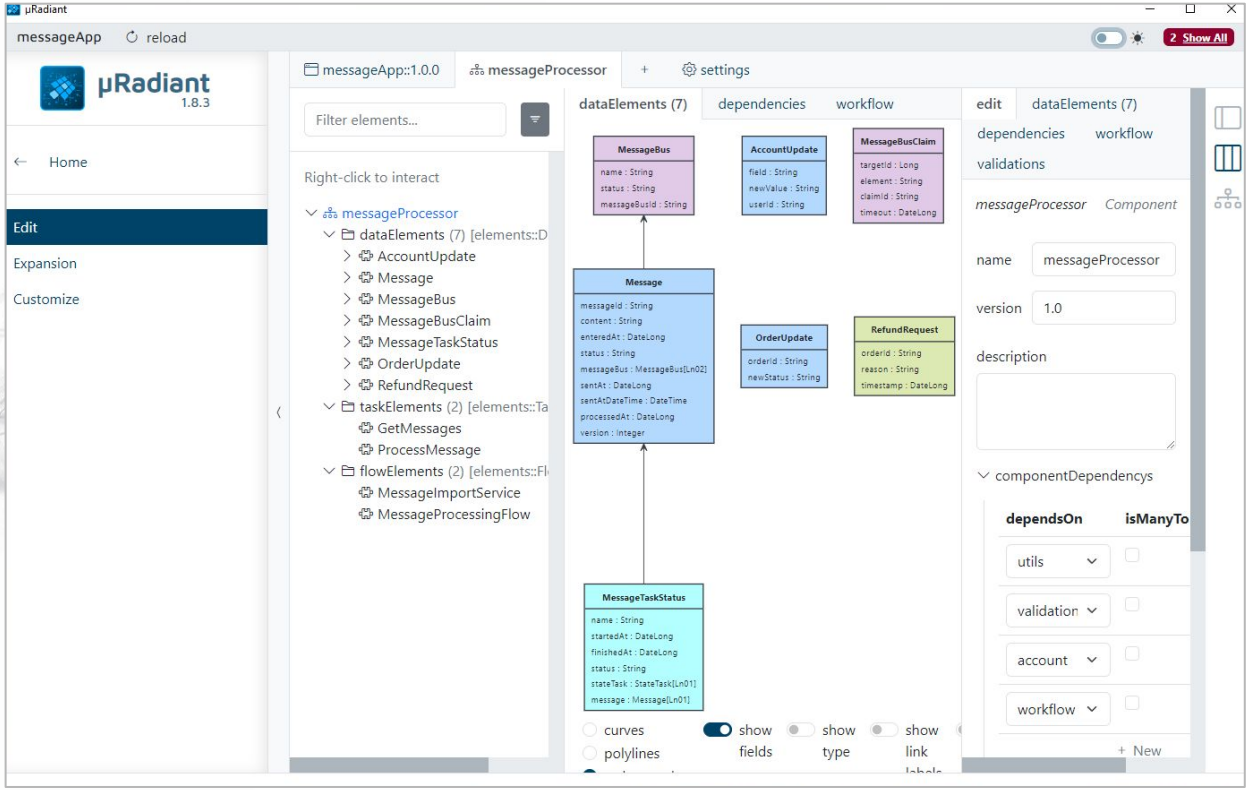


Metamodels

Custom Rules



µRadiant


µRadiant
messageApp
reload
2 Show All
µRadiant
1.8.3
Home
Edit
Expansion
Customize
messageApp::1.0.0
messageProcessor
settings
Filter elements...
Right-click to interact
messageProcessor
dataElements (7) [elements::D
AccountUpdate
Message
MessageBus
MessageBusClaim
MessageTaskStatus
OrderUpdate
RefundRequest
taskElements (2) [elements::Ta
GetMessages
ProcessMessage
flowElements (2) [elements::Fl
MessageImportService
MessageProcessingFlow
dataElements (7)
dependencies
workflow
MessageBus
name : String
status : String
messageBusId : String
AccountUpdate
field : String
newValue : String
userId : String
MessageBusClaim
targetId : Long
element : String
claimId : String
timeout : DateLong
Message
messageId : String
content : String
enteredAt : DateLong
status : String
messageBus : MessageBus[Ln02]
sentAt : DateLong
sentAtDateTime : DateTime
processedAt : DateLong
version : Integer
OrderUpdate
orderId : String
newStatus : String
RefundRequest
orderId : String
reason : String
timestamp : DateLong
MessageTaskStatus
name : String
startedAt : DateLong
finishedAt : DateLong
status : String
stateTask : StateTask[Ln01]
message : Message[Ln01]
curves
polylines
show fields
show type
show link
edit
dataElements (7)
dependencies
workflow
validations
messageProcessor Component
name
messageProcessor
version
1.0
description
componentDependencys
dependsOn
isManyTo
utils
validation
account
workflow
+ New

NS Scripting

Platform independent scripts for NS development processes

- Based on the Kotlin language engine
- Simplified scripting syntax inspired by Jenkins
- Platform independent commands
  - Platform specific tooling resolution
- API tailored to NS project development
  - Expanders, Maven, Git, …
- Scoped contexts (e.g. directory scoping)
- 2023: Script argument API, NS project I/O, …

Kotlin

```
#!/usr/local/bin/env nss

ns {
 expand {}
}

dir("expansions/myApp") {
 mvn {
   task("clean")
   task("package")
 }
}
```

NS Initializer

Evolution

1.x.x
2021Q2

Custom templating engine with basic variable replacement
Application & expander projects

2021Q3
2.x.x

Direct interfacing with expander framework with standardized expander definitions (string template + mapping)

3.x.x
2021Q4

Initializer metamodel to model projects and their arguments with individual expansion resources for initializers
Added initializer-initializer and dynamic CLI and interactive mode

2022
3.x.x/4.x.x

Added metamodel project
Further refinement and servicing of existing project initializers
Distribution with installers and package managers

4.x.x/5.x.x
2023

Initializer project modules (like components in applications) to reuse, combine and extend initializers

NS Initializer

µRadiant Integration

µRadiant 1.8.3

← Home

Import Existing Project

New Application Project

New Metamodel Project

Project Directory:

Select F:/NSF/workspace/demo

application

demo

The name of the application.

applicationInstance

demo

The name of the application instance. If not provided, the application name is used.

component

demoComp

The name of the main component.

groupId

org.normalizedsystems

The groupId that will be used in the POM files.

dockerImage

nsapp/demo

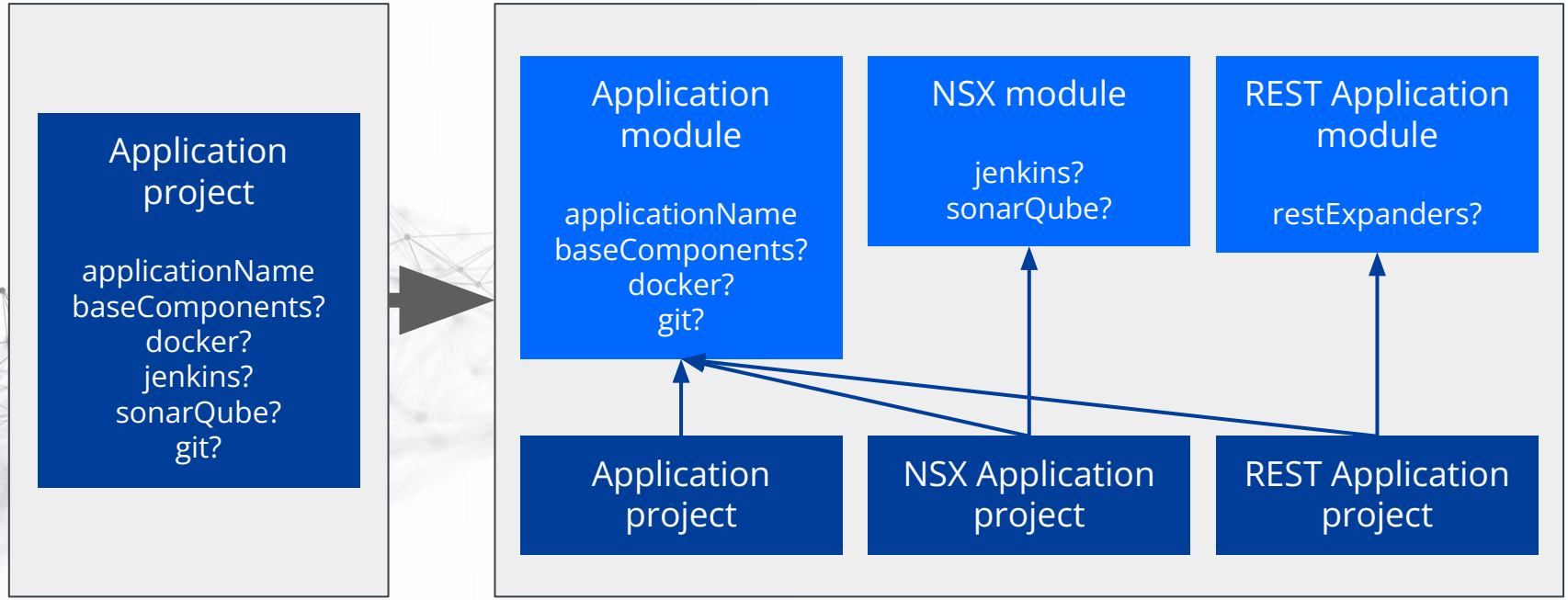
The Docker image name.

> Advanced

- [x] jenkins
  Create required files to build the project on Jenkins.
- [x] baseComponents
  Add the base components as a dependency for the application.
- [x] git
  Initialize a local git repository for the project and create an initial commit.
- [x] sonar
  Add SonarQube integration to the project.
- [x] modelValidation
  Generate configuration to run model validations.
- [x] docker
  Generate files to package the application as Docker image and deploy locally in Docker with docker-compose.
- [x] scripts
  Generate utility/demo scripts.
- [ ] aws
  Generate Terraform scripts to deploy to Amazon Web Services.
- [ ] examples
  Generate example content for the project.

NS Initializer

Planned: Project modules



Docker images

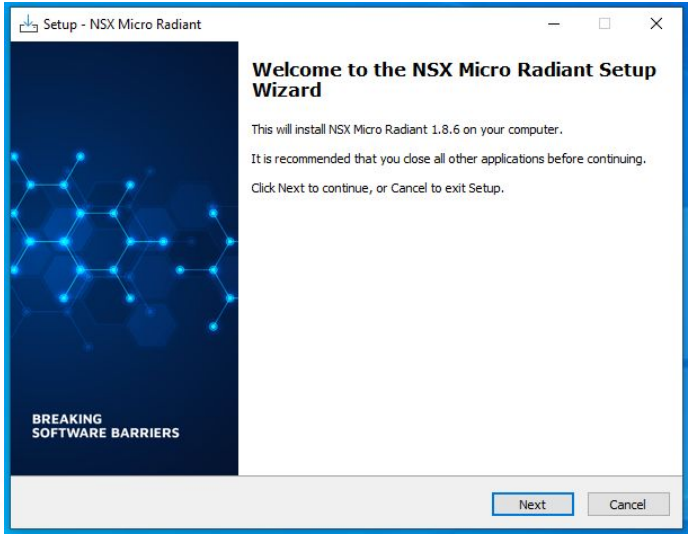
Simplified and consistent releases

Tag	Example	Fixed	Description
`latest`	`latest`	✗	The latest version of the image, latest supported version of JRE and latest version TomEE.
`<tomee-major-version>`	`8`	✗	The latest version of the image, latest supported version of JRE and latest major version TomEE.
`<tomee-version>`	`8.0.13`	✗	The latest version of the image, latest supported version of JRE and fixed version TomEE.
`<tomee-version>-<image-version>`	`8.0.13-2.13.0`	✓	A fixed version of the image, latest supported version of JRE and fixed version TomEE.

ApacheTomEE HIBERNATE Flyway

Product development

- More internal and external users
  - Increase UX and accessibility of tooling
  - Better documentation
  - Simplified distribution and setup of tooling
    - Windows installers
    - Ubuntu packages
    - Chocolatey packages



New Foundation

Foundation Tools Releases Development Blog

Search CTRL K

Normalized Systems

BREAKING SOFTWARE BARRIERS

NS Application Development relies on 4 principles, based on engineering concepts such as system theoretic stability and thermodynamic entropy, to create evolvable software.

Data Version Transparency

Data structures can have multiple versions without affecting the processing functions that consume or produce them.

Action Version Transparency

It should be possible to upgrade a processing function without affecting processing functions that call it.

Separation of Concerns

A processing function should not address more than one concern and should thus not include more than one task.

Separation of States

Tasks should clearly expose the state of their execution, so that errors can be correctly attributed to their causes.

New Foundation

- Deprecated content pruned
- Improved/rewritten existing content
- Centralized all official documentation
- Blog with release notes and migration guides
- Powerful search functionality:
  - Carefully indexed
  - Weighted results
  - Includes Maven plugin documentation