

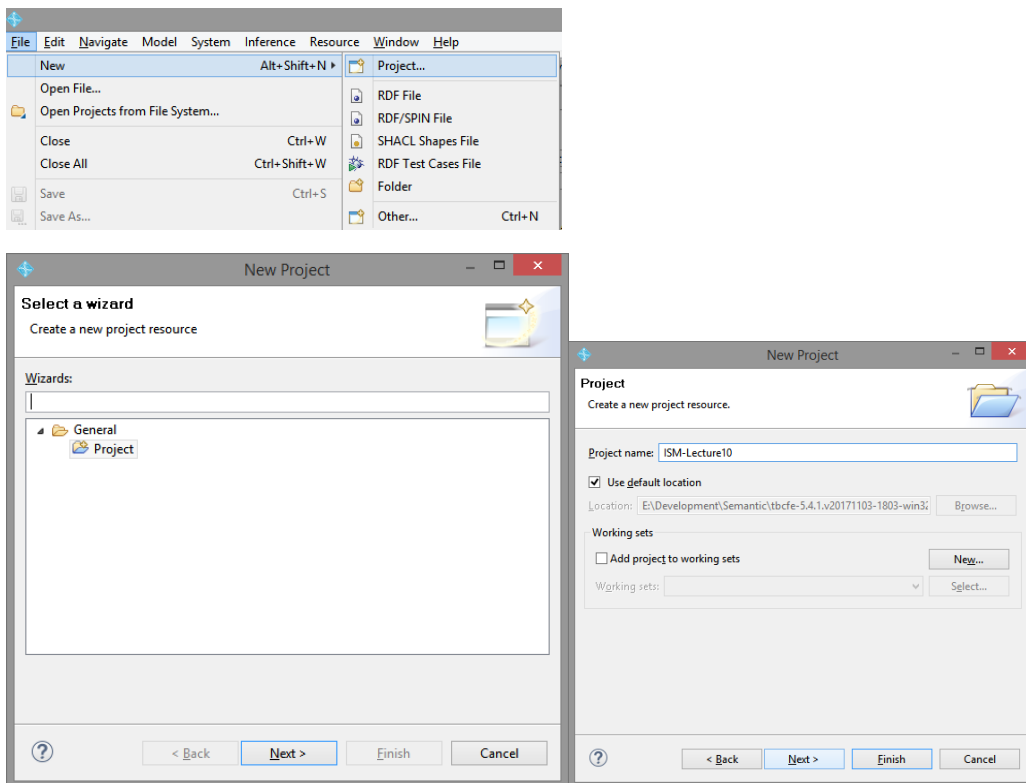
# Data models - design and validation

## Introduction to SHACL

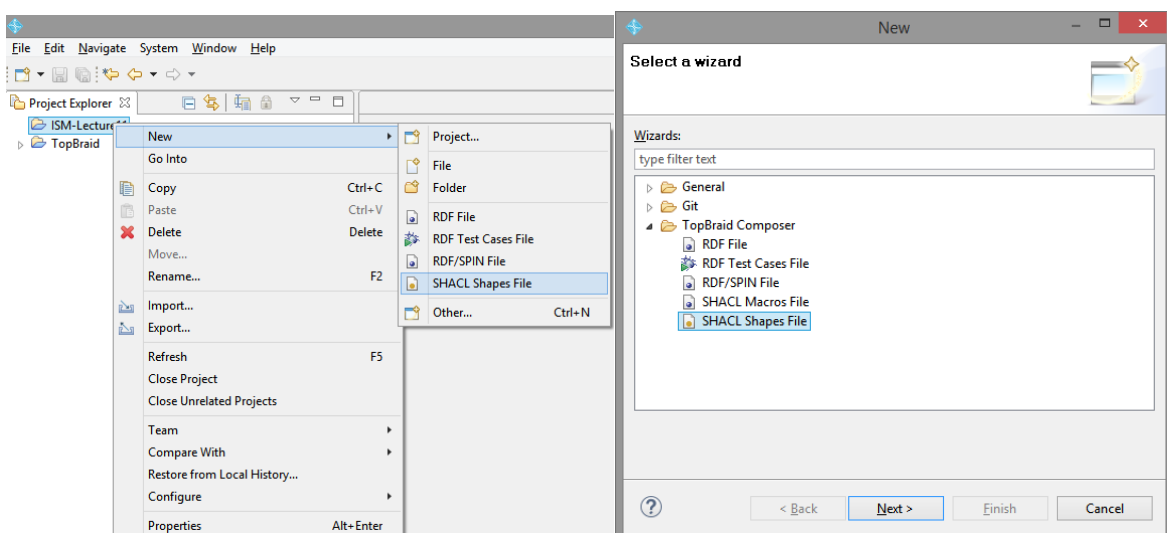
(based on <https://www.topquadrant.com/technology/shacl/tutorial/>).

RDF graphs can be validated with the use of SHACL. This can be demonstrated in TopBraid Composer

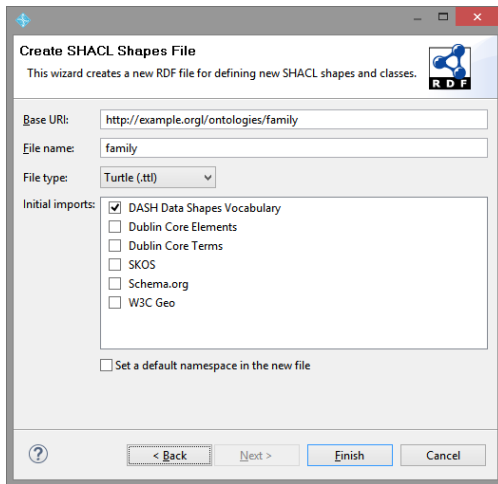
### 1. Open new Project



### 2. Create a new shape file in it (File->New->SHCL ShapeFile or File->New->Other...)



naming it correctly and, additionally, selecting DASH support (<https://datashapes.org/dash/>):



## DASH Data Shapes Vocabulary

Unofficial Draft 30 September 2021

### More details about this document

**Latest published version:**  
<https://www.w3.org/dash/>

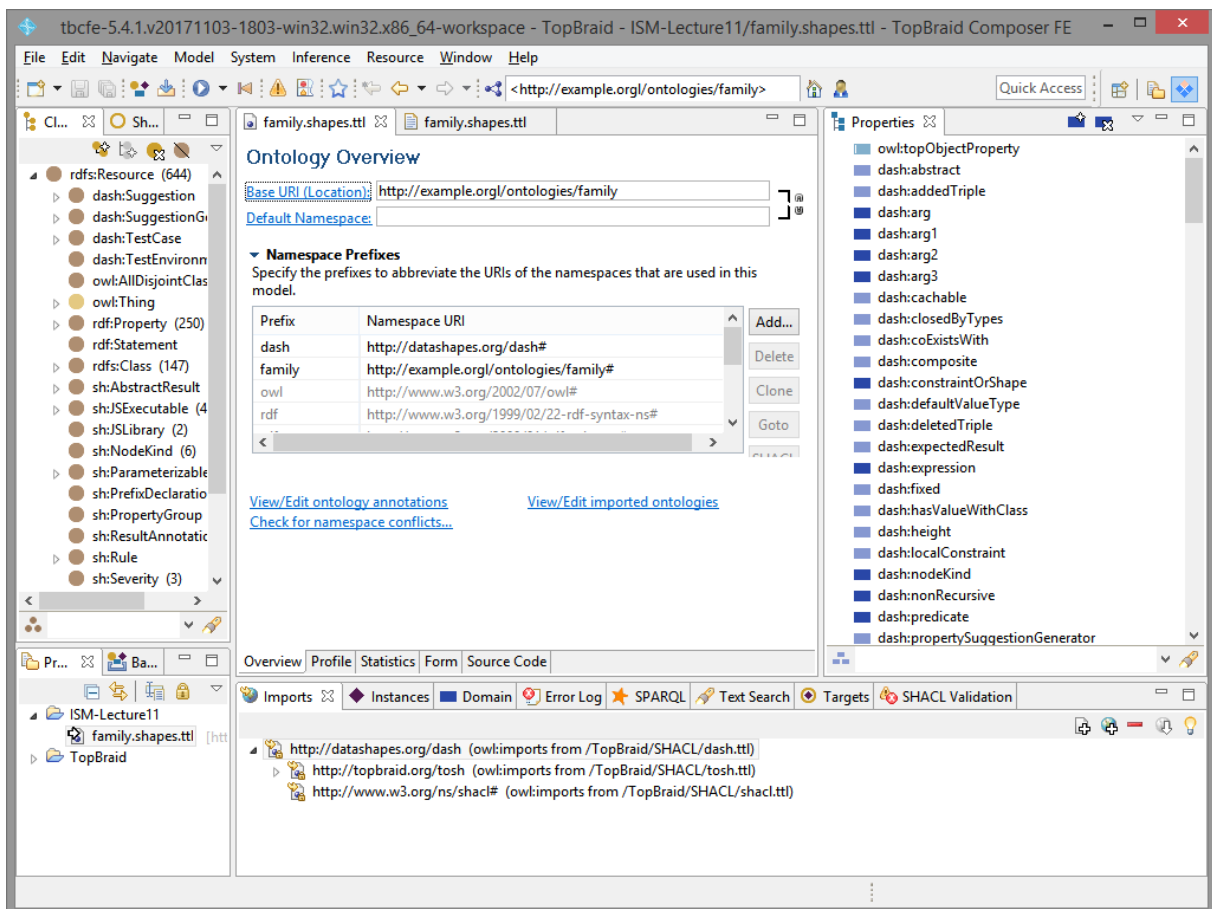
**Latest editor's draft:**  
<http://datashapes.org/dash.html>

**Editor:**  
[Holger Knublauch \(TopQuadrant, Inc.\)](#)

### Abstract

This document introduces the DASH Data Shapes Vocabulary, a collection of reusable extensions to SHACL for a wide range of use cases. In addition to a library of new SHACL constraint and target types, DASH also includes components for representing test cases, suggestions to fix constraint violations and an extended validation results vocabulary. Finally, DASH serves as a reference implementation of SHACL in SPARQL by providing default validators. DASH is intended to evolve as a standards-compliant open source vocabulary.

After that you will get:



```
# baseURI: http://example.org/ontologies/family
# imports: http://datashapes.org/dash
# prefix: family
```

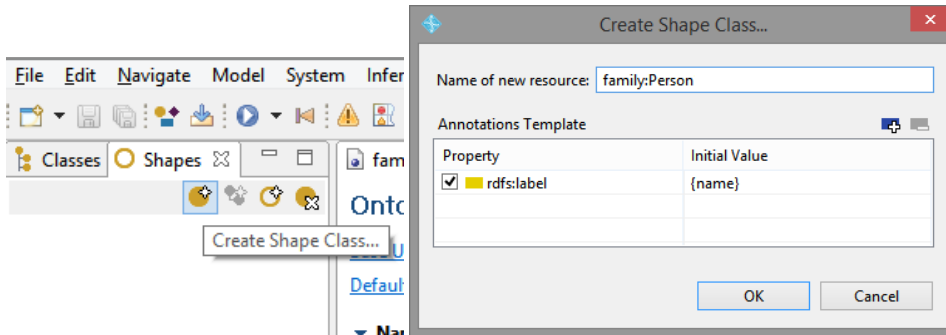
```
@prefix dash: <http://datashapes.org/dash#> .
@prefix family: <http://example.org/ontologies/family#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```

<http://example.org/ontologies/family>
  rdf:type owl:Ontology ;
  owl:imports <http://datashapes.org/dash> ;
  owl:versionInfo "Created with TopBraid Composer" ;

```

### 3. Create family:Person shape class:

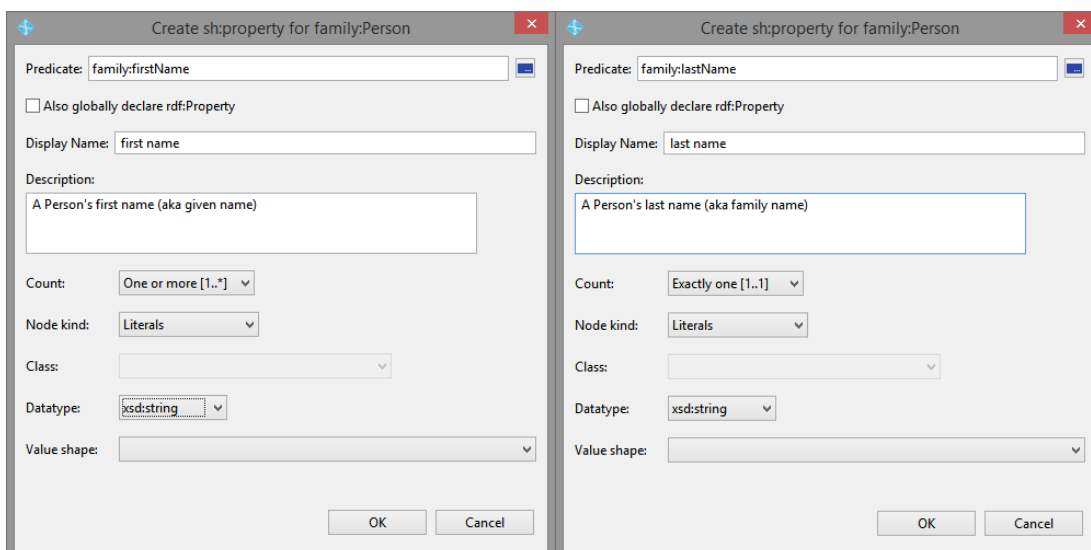
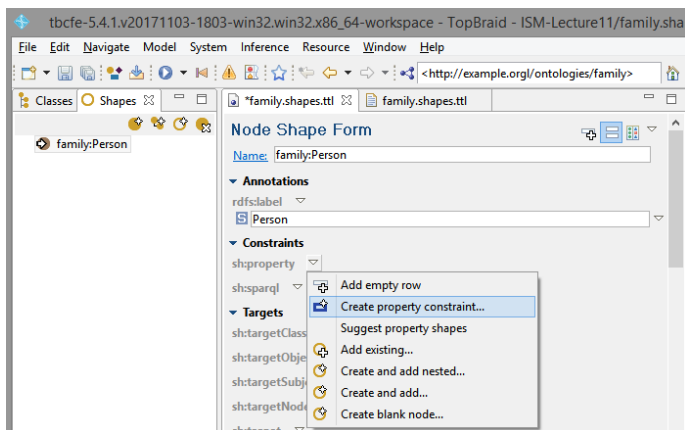


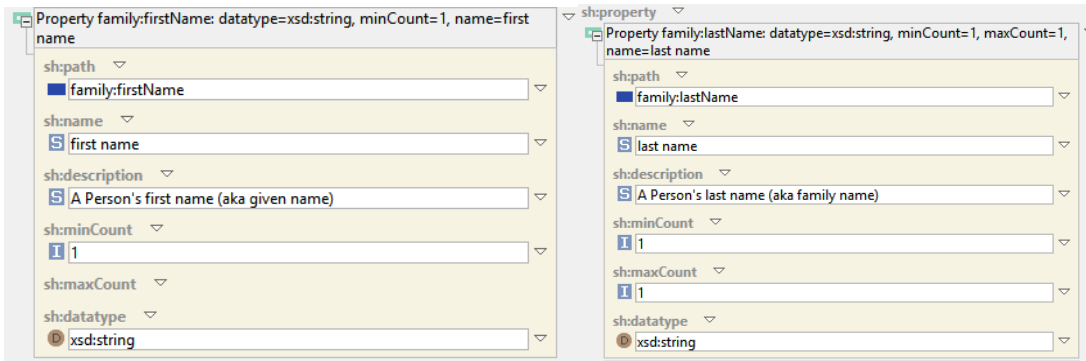
```

family:Person
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:label "Person" ;
  rdfs:subClassOf rdfs:Resource ;

```

### 4. Create property constraint family:firstName and family:lastName (from the context of sh:property):

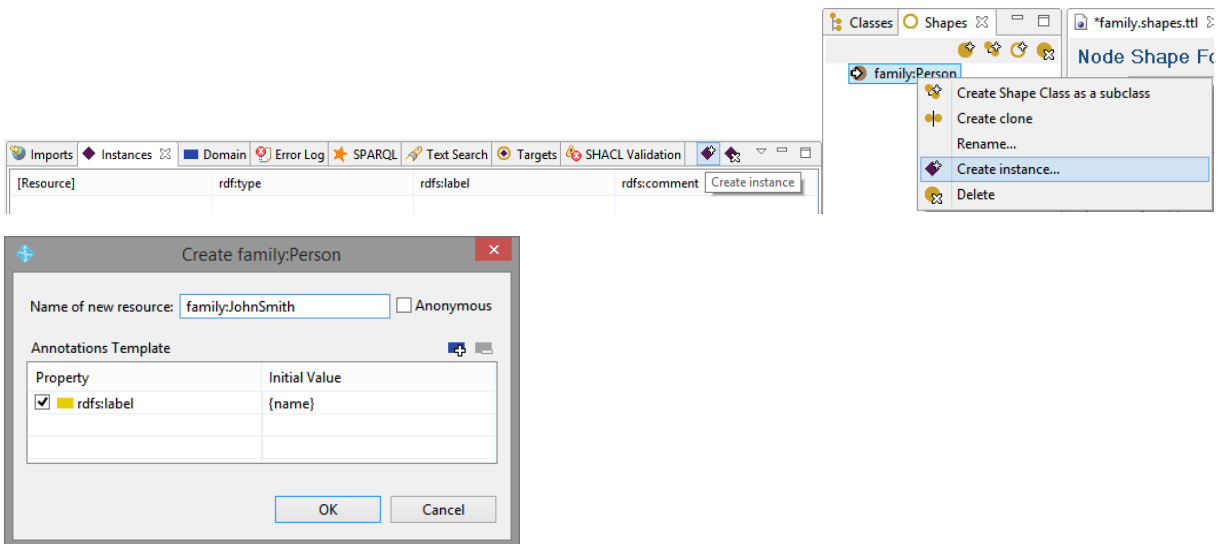




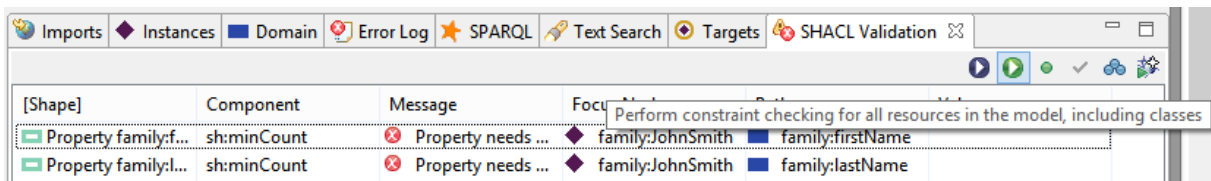
Note, that for “Also globally declare rdf:Property”

If checked and the predicate has no rdf:type yet, it will get a global rdf:type rdf:Property triple, and instead of sh:name and sh:comment, the global property will have rdfs:label and rdfs:comment.

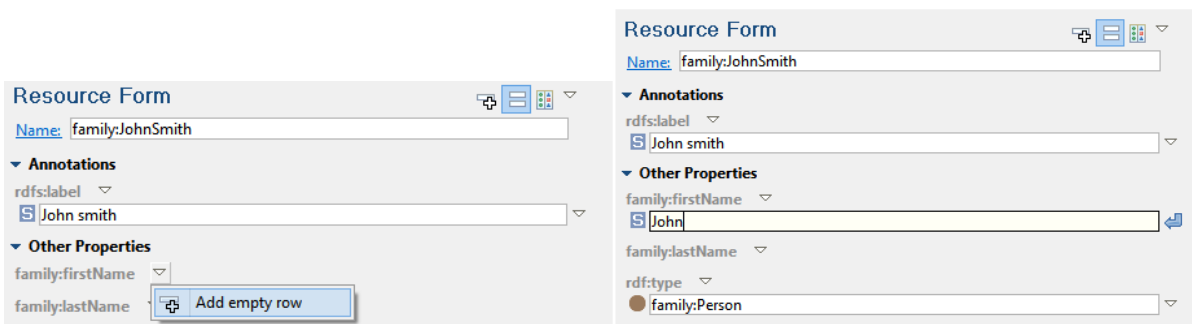
5. Now you may try to validate your triples. Thus create an instance of a Person class (you have two possibilities to start the wizzard).



You can check, that without assigning first name and last name property to this instance the graph will not validate correctly



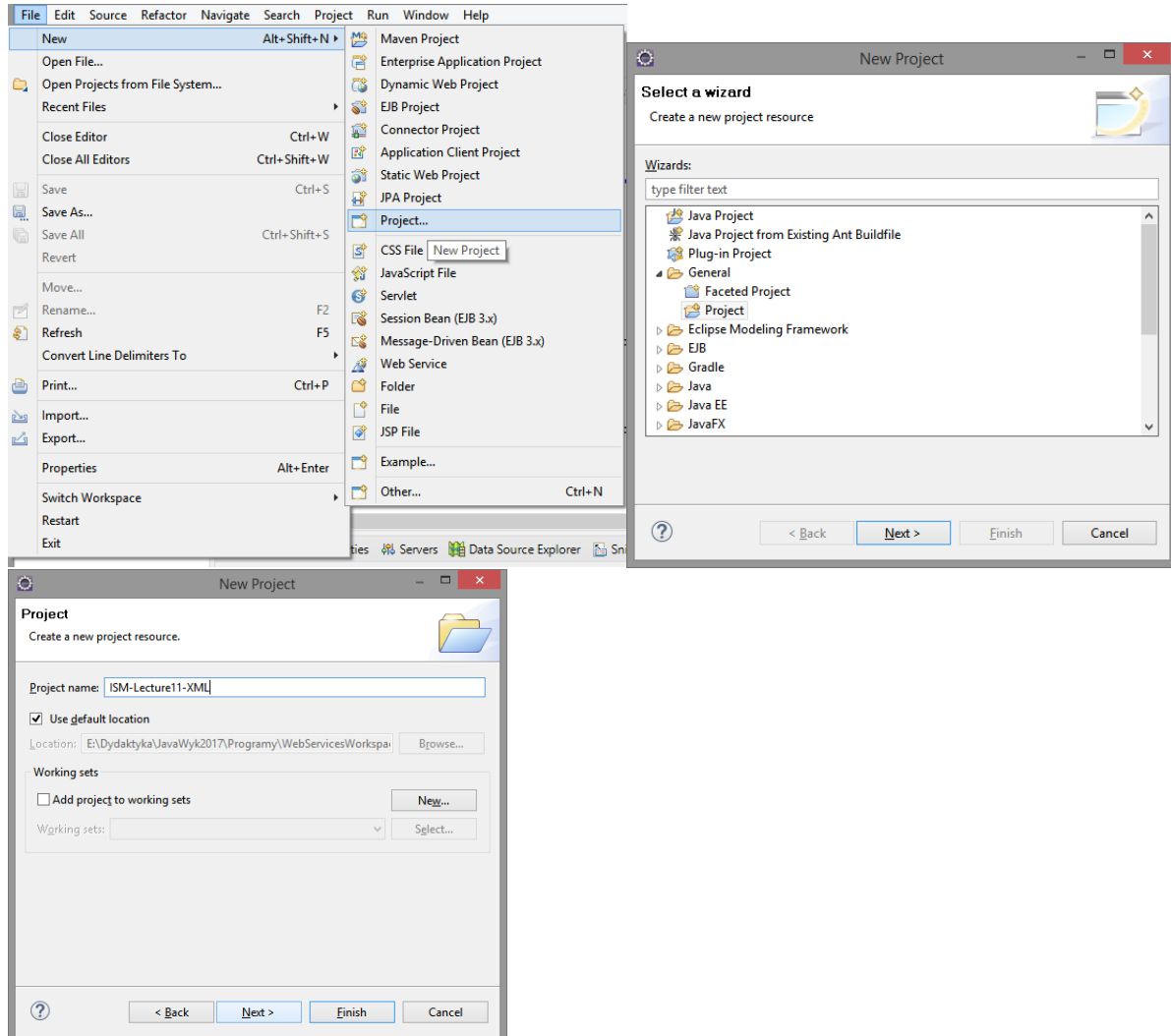
The errors will disappear after addition of missing properties



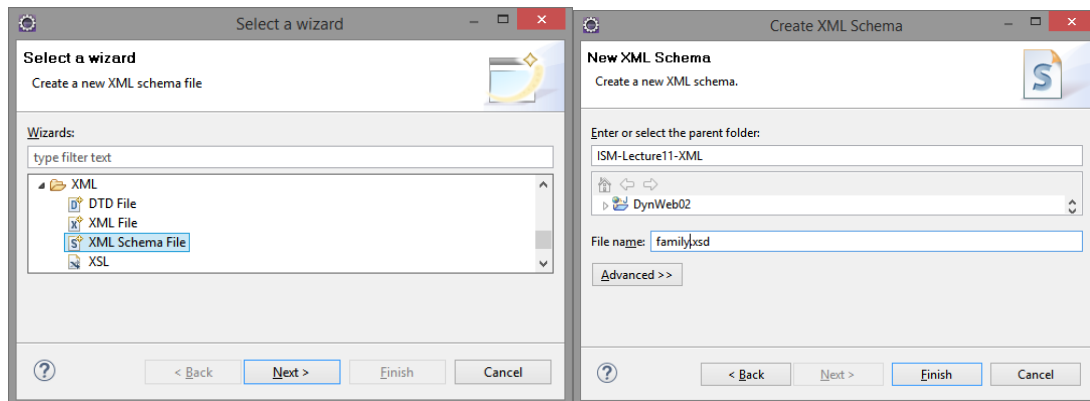
# Introduction XML Schema

XML data can be validated with the use of XML Schema. This can be demonstrated with the use of eclipse (with proper plug-ins).

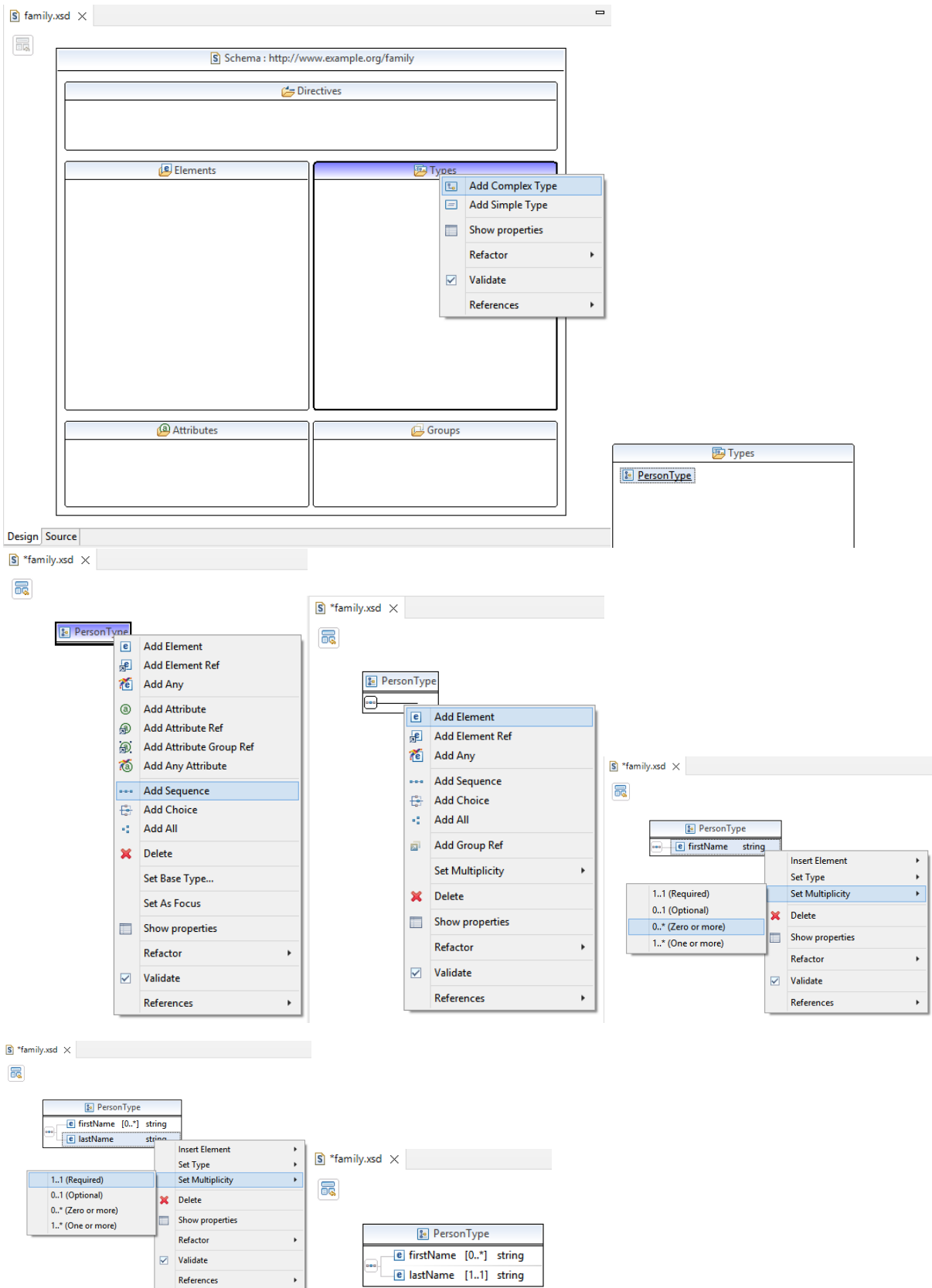
## 1. Create general project



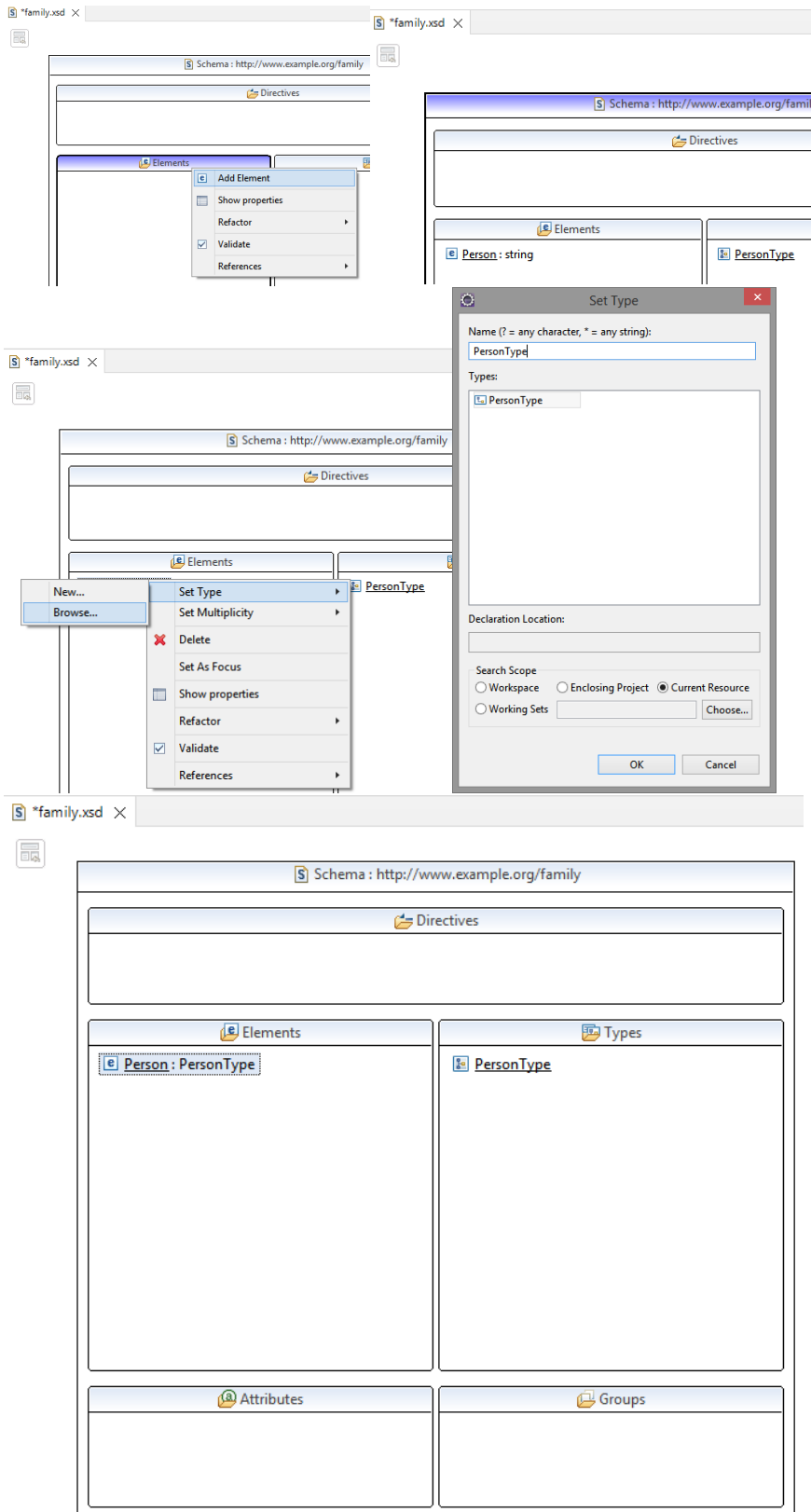
## 2. Add XML Schema file (File->New->Other...)



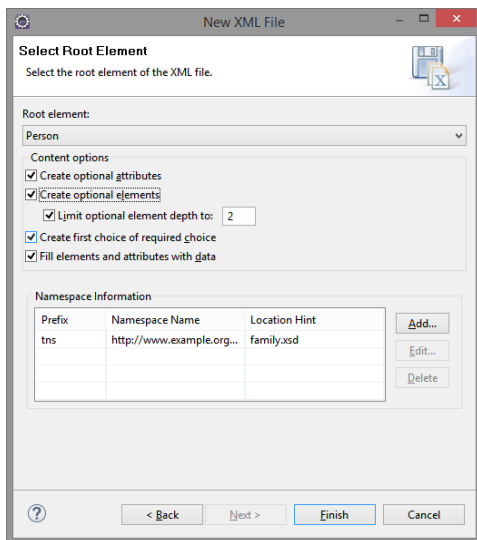
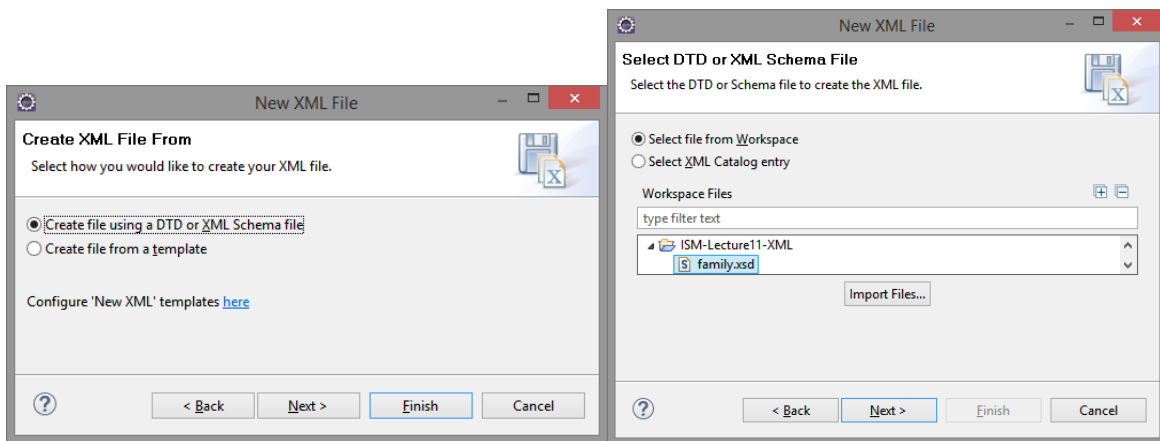
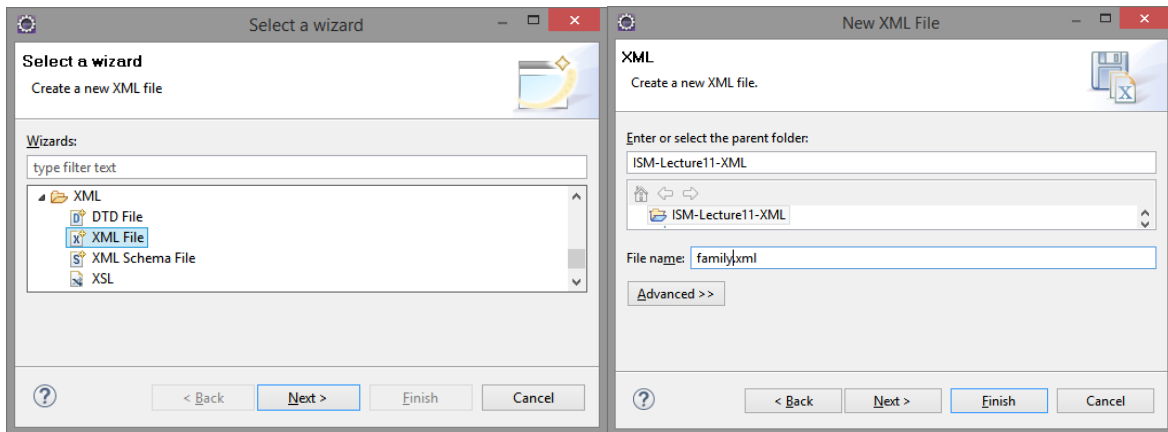
## 2. Declare schema, starting from adding complex type



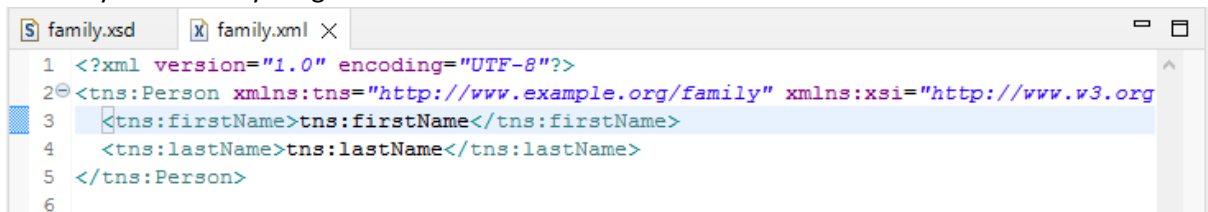
and then adding an element



2. Declare xml data file (based on designed schema). Start with File->New->Other ...



2. Now you can edit your generated xml data file.





```
family.xsd  *family.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <tns:Person xmlns:tns="http://www.example.org/family" xmlns:xsi="http://www.w3.org
3   <tns:firstName>John</tns:firstName>
4   <tns:lastName>Smith</tns:lastName>
5 </tns:Person>
6
```

After removing required elements

```
family.xsd  *family.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <tns:Person xmlns:tns="http://www.example.org/family" xmlns:xsi="http://www.w3.org
3   <tns:firstName>John</tns:firstName>
4 </tns:Person>
```

you will be warn about data inconsistency

```
family.xsd  *family.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <tns:Person xmlns:tns="http://www.example.org/family" xmlns:xsi="http://www.w3.org
3   <tns:firstName>John</tns:firstName>
4 </tns:Person>
```

Child elements are missing from element: - tns:Person The following elements are expected: - firstName - lastName Error indicated by {http://www.example.org/family}:firstName, {http://www.example.org/family} with code: