

JSON and JSON Schema Support



Octavian Nadolu, Syncro Soft
octavian_nadolu@oxygenxml.com
@OctavianNadolu

schematron
Structured
editing
XML
review
XQuery
Publish
PDF
WebDAV
DTD
DocBook
oxygen
authoring
XML Editor
XSD
SCHXSD
Single
XPRRNCFO
frameworks
Profiling
WSDL
styles
visual
WebHelp
DITA
TEI
XSL
PHP
Ant
Js

JS
KML
XSLT
SVN
JSON
SVG
IDREFS
WebDAV
DocBook
oxygen
authoring
XML Editor
Single
Source
Database
XHTML
Cha
Col
Wa

© 2020 Syncro Soft SRL. All rights reserved.

Agenda

- JSON and JSON Schema in Oxygen
- Validating JSON with Schematron
- Convert between JSON and XML
- XSD to JSON schema converter
- Instance generator from a JSON Schema
- Generate JSON Schema from JSON instance



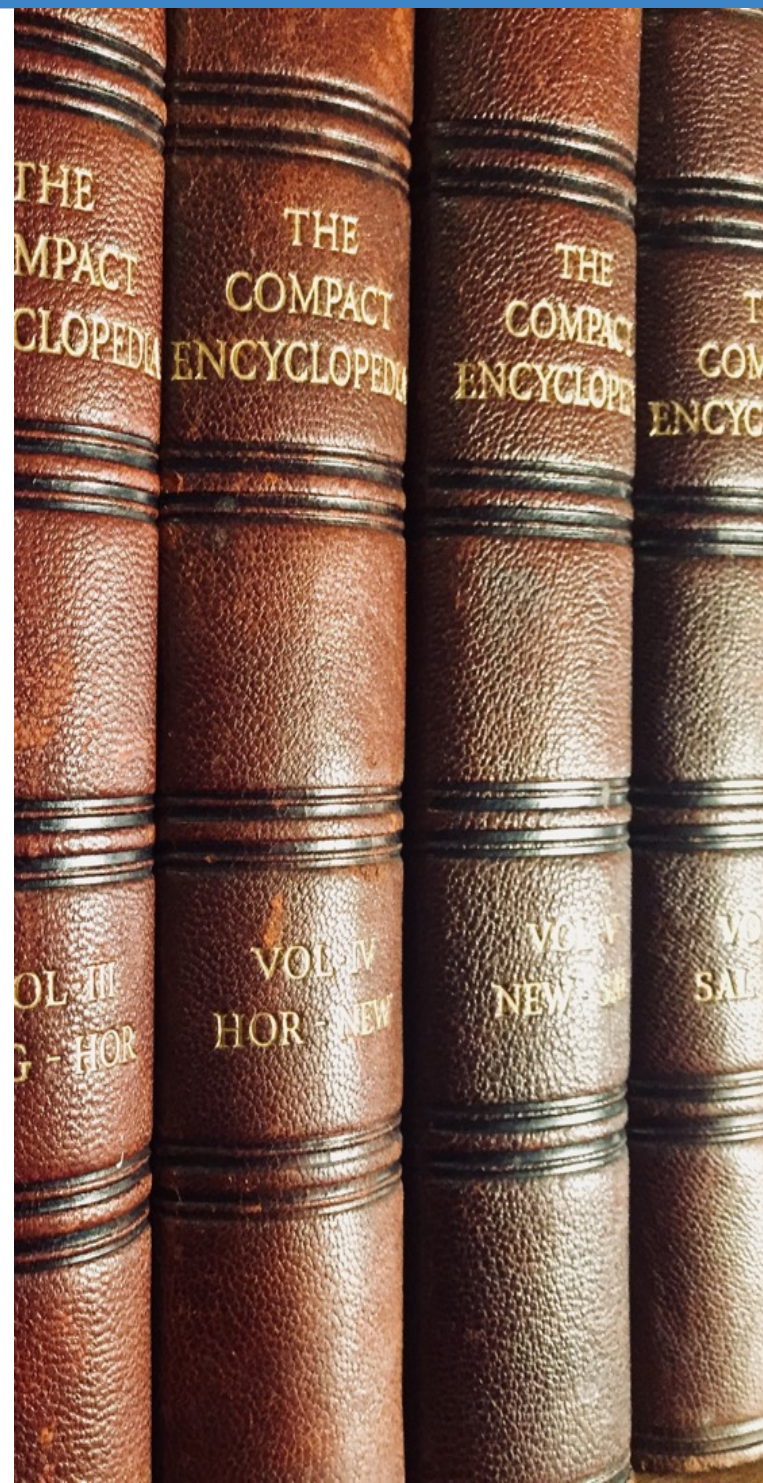
JSON

(JavaScript Object Notation)

- Data representation format
- Used for API and Configs
- Lightweight and easy to read/write

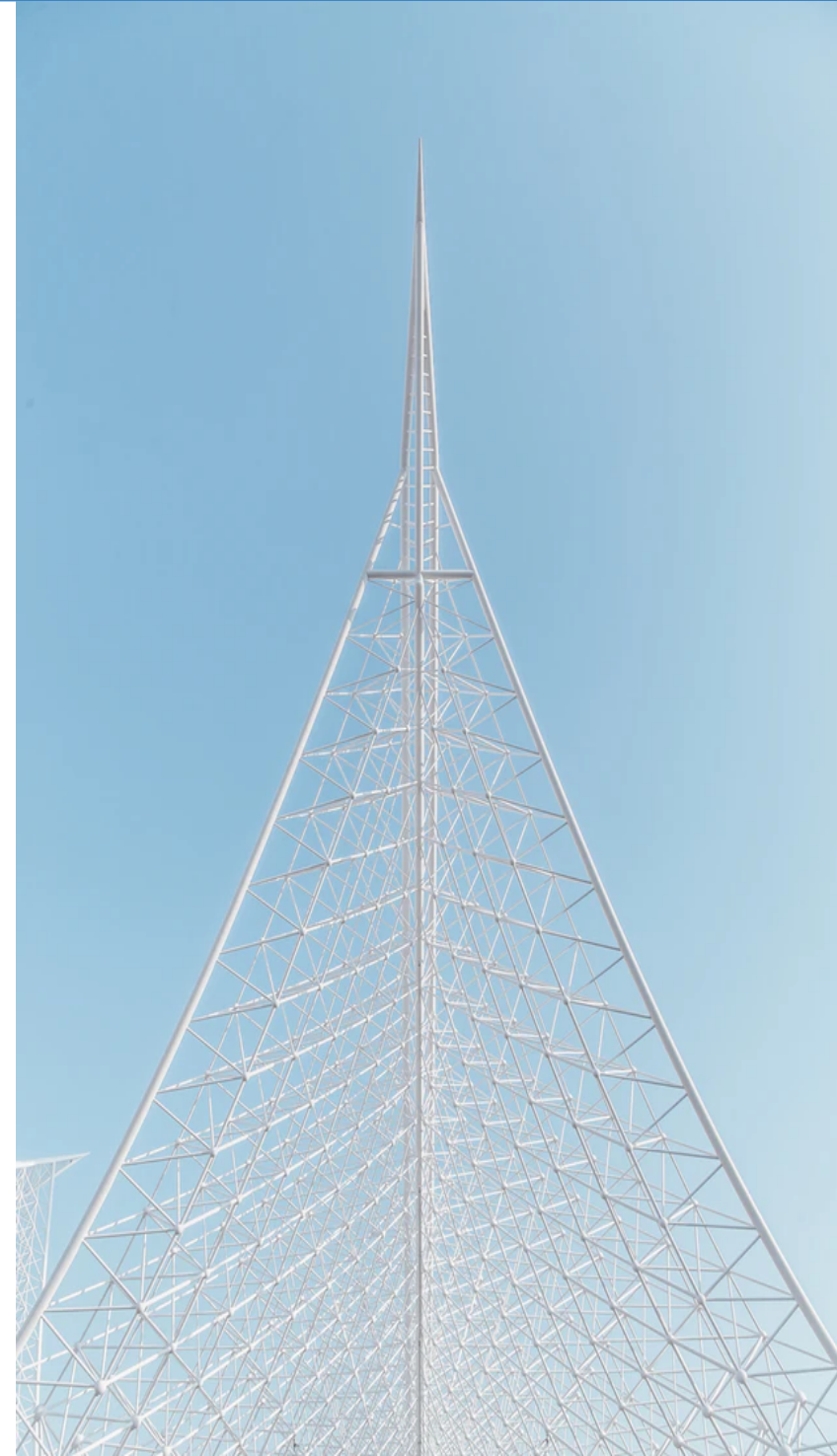


<http://json.org>



JSON Documents Structure

- JSON Data – name/value pair
- JSON Object {...}
- JSON Array [...]



Validating JSON Documents

- Checking **Well-Formedness** in JSON Documents
- **Validating JSON** Documents Against JSON Schema
- **Validating JSON Schema** According to the Specification

Checking Well-Formedness

Check if the JSON document respects the JSON specification

```
{  
  "id": "jane.doe",  
  "firstname": "Jane",  
  "lastname": "Doe",  
  "email": "jane@oxygenxml.com",  
  "age": 37  
}
```

ECMA-404 The JSON Data Interchange Standard

<http://json.org/>

Example

```
{
  "persons" [
    {
      "id": "jane.doe",
      "firstname": "Jane",
      "lastname": "Doe",
      "email": "jane@oxygenxml.com"
      "age": 37,
    },
    {
      "id": "john.jones",
      "firtname": "John",
      "lastname": "Jones",
      "email": "john@oxygenxml.com",
      "age": 42
    }
  ]
}
```

Expected a ':' after a key

Expected ',' character

Unexpected ',' character

Expected quotes

Expected '}'

JSON Schema

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents



<http://json-schema.org>



Defining JSON Schema

- Similar to XML Schema, RNG, or DTD
- Written in JSON
- Used to define the structure of a JSON data

```
{"type": "string"}
```

JSON Schema

```
"I'm a string"
```

JSON Instance

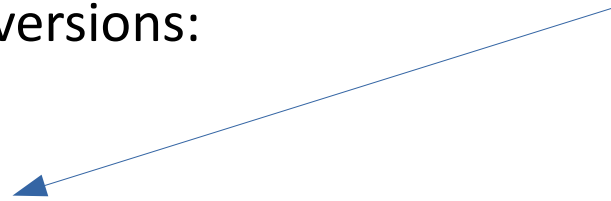
JSON Schema Definition

- It is recommended to have the schema definition on the first level

"\$schema": "<http://json-schema.org/draft-07/schema#>"

- JSON Schema used versions:

- Draft 4
- Draft 6
- Draft 7
- Draft 8 (2019-09)

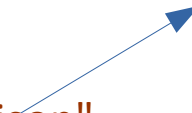


Associate JSON Schema

- Associating a Schema to JSON Documents
 - Directly in JSON document – using `$schema` property
 - In application options

```
{  
  "$schema": "person-schema.json",  
  "persons" [ ... ]  
}
```

Absolute or relative URI



Validating JSON Schema

- Check for well-formedness
- Validate accordingly to the Internet Engineering Task Force (IETF) Specification



Validating JSON with Schematron

- Specify the JSON structure using JSON Schema
- Express co-constraints and define custom rules using Schematron



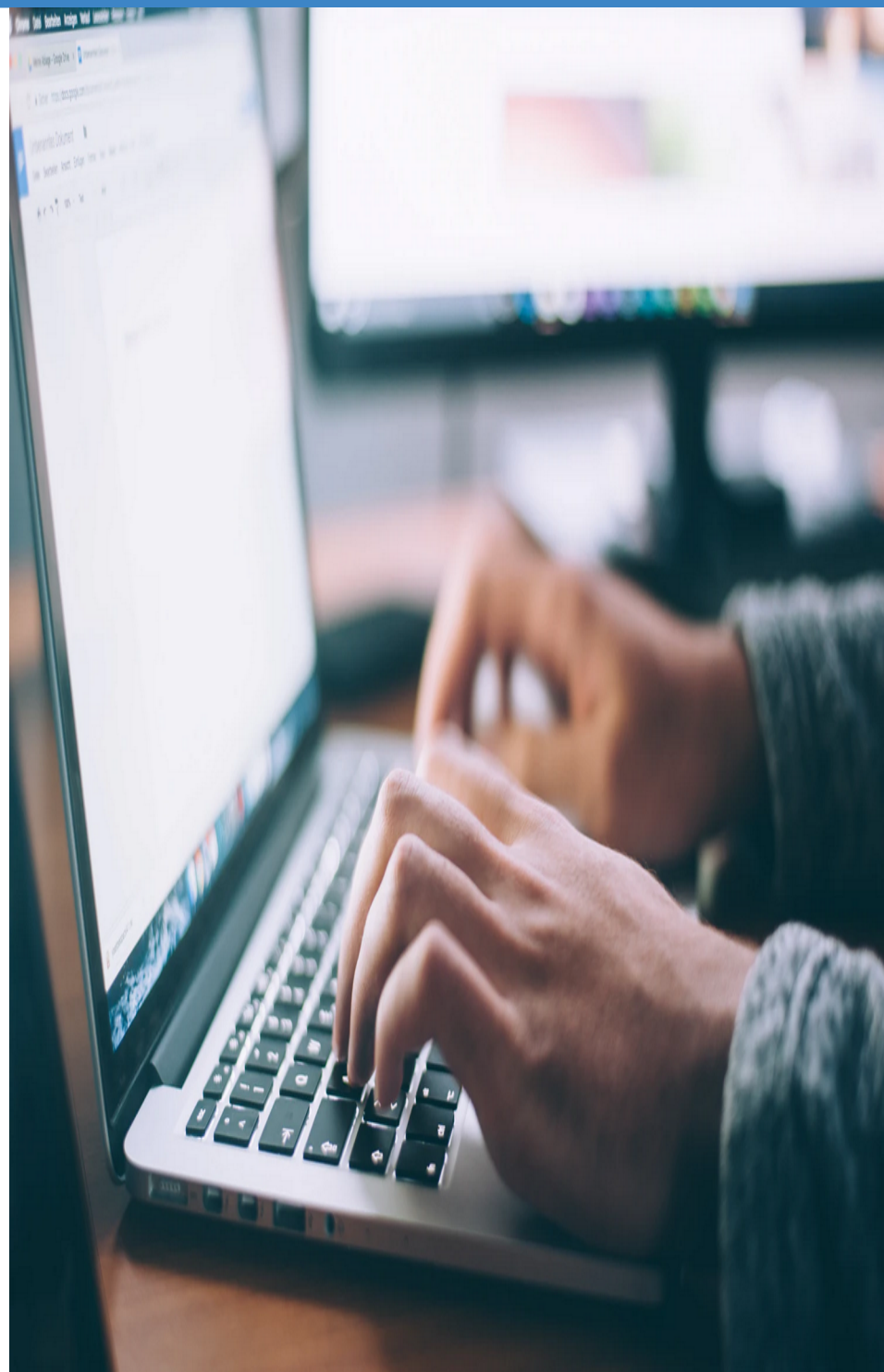
Schematron for JSON

- Use XPath to express the rules

```
<sch:rule context="persons">  
  <sch:assert test="number(age) > preceding-sibling::node()/age ">  
    The person age must be greater than the previous person age </sch:assert>  
</sch:rule>
```

Editing JSON

- JSON Text and Grid Editing Mode
- JSON Outline View
- Content Completion Assistant in JSON
- Associate JSON Schema to JSON Documents



Text and Grid Editing Mode

The image displays two side-by-side screenshots of a JSON editor interface, demonstrating different editing modes for a JSON file named 'personal.json'.

Left Screenshot (Text Editing Mode): The editor shows the raw JSON text with line numbers 1 through 22. The text is as follows:

```
1 {  
2   "$schema": "personal-schema.json",  
3   "personnel": {  
4     "person": [  
5       {  
6  
7         "id": "Big.Boss",  
8         "name": {  
9           "family": "Boss",  
10          "given": "Big"  
11        },  
12        "email": "chief@oxygenxml.com",  
13        "link": {  
14        "subordinates": [  
15          "one.worker",  
16          "two.worker",  
17          "three.worker",  
18          "four.worker",  
19          "five.worker"  
20        ]  
21      }  
22    }  
23  ]  
24 }
```

The 'Text' tab is selected at the bottom.

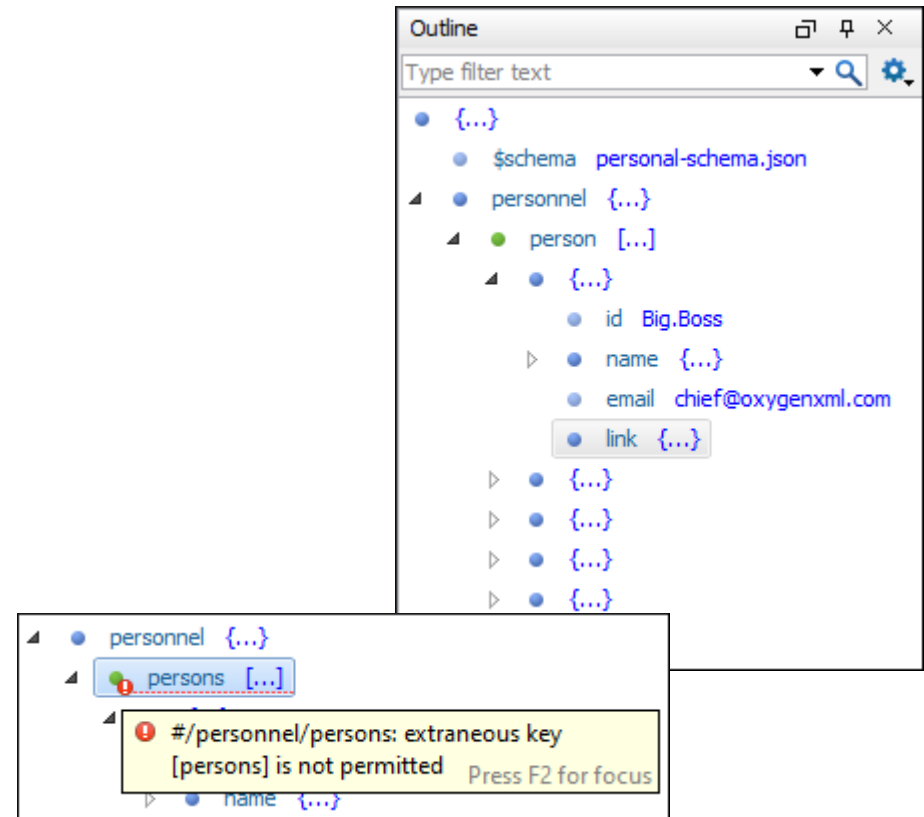
Right Screenshot (Grid Editing Mode): The editor displays the same JSON data in a structured grid view. The root object is expanded to show 'personnel' and 'person'. The 'person' array (6 rows) is shown as a table with columns for 'id', 'name', and 'email'. The 'name' column is further expanded to show 'family' and 'given' sub-properties.

id	name	email
1 "Big.Boss"	name family "Boss" given "Big"	"chief@oxygenxml.com"
2 "one.worker"	name family "Worker" given "One"	"one@oxygenxml.com"
3 "two.worker"	name family "Worker" given "Two"	"two@oxygenxml.com"

The 'Grid' tab is selected at the bottom.

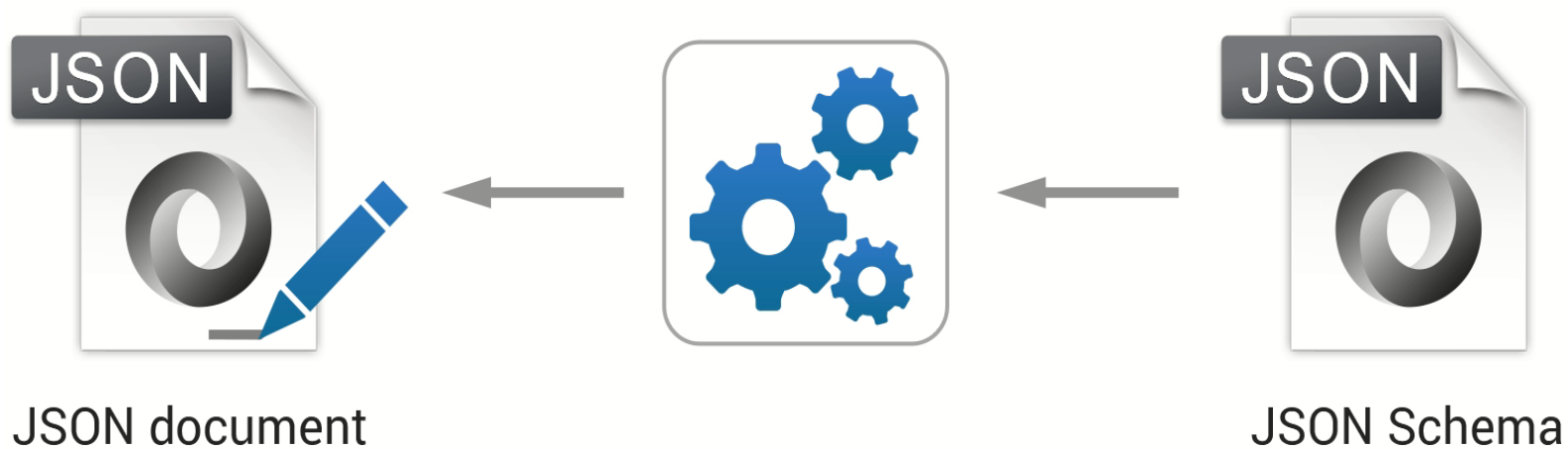
JSON Outline View

- Document Overview
- Synchronization with Editor
- Filtering
- Error Markers



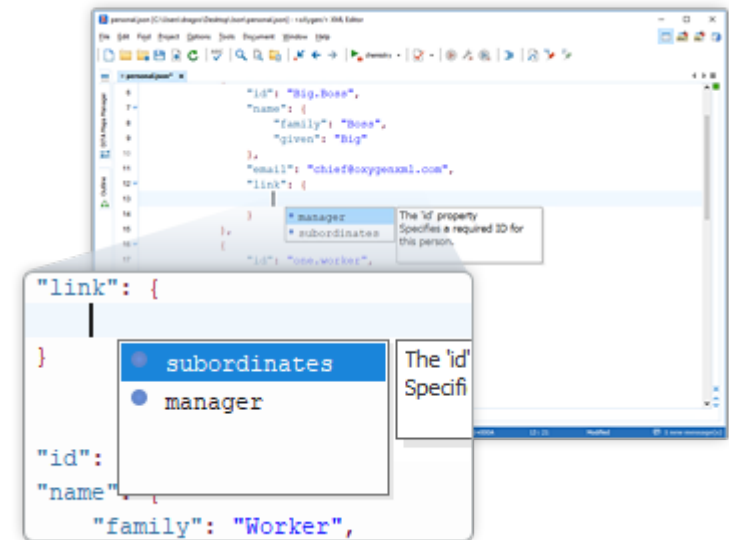
Associate JSON Schema

- Associating a Schema to JSON Documents
 - Directly in JSON Documents
 - Through a Validation Scenario



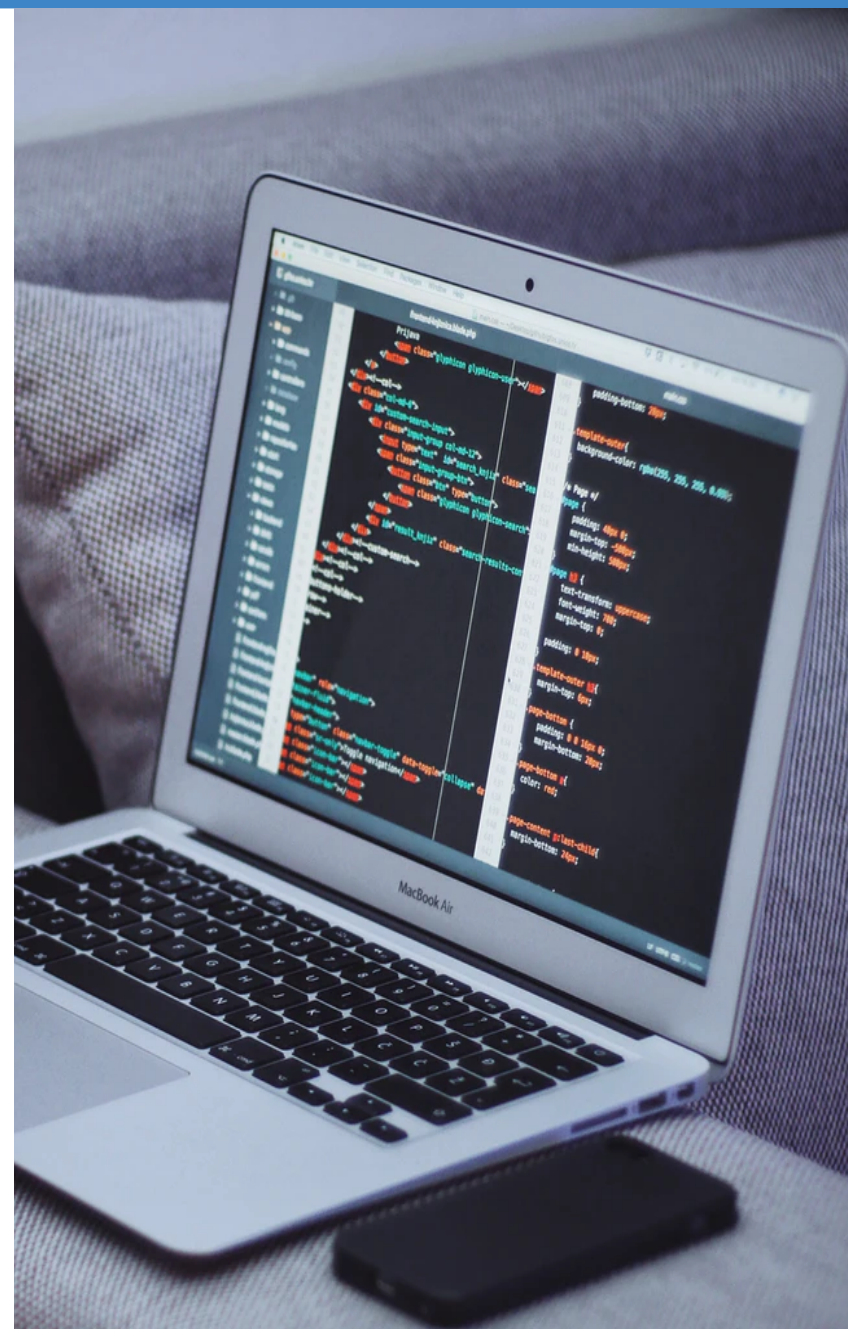
Content Completion

- Context-Sensitive
- Valid Proposals
- Documentation from Schema
- Code Templates



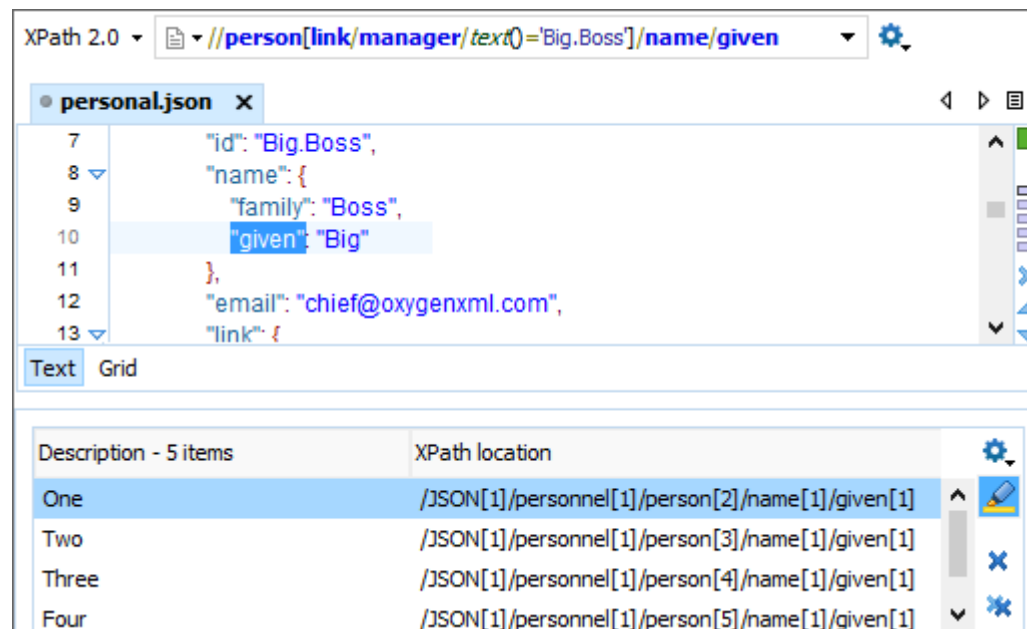
Querying and Transforming

- Query using XPath
- Transform JSON using XSLT and XQuery



XPath

- XPath Toolbar and XPath Builder View
- Content Completion
- XPath Over Multiple Files



The screenshot displays an IDE interface for XPath 2.0. At the top, the XPath toolbar shows the expression `//person[link/manager/text()='Big.Boss']/name/given`. Below the toolbar, the JSON file `personal.json` is open, with the value `"given": "Big"` highlighted on line 10. The XPath Builder view at the bottom shows a table with 5 items, listing the XPath location for each occurrence of the selected text.

Description - 5 items	XPath location
One	<code>/JSON[1]/personnel[1]/person[2]/name[1]/given[1]</code>
Two	<code>/JSON[1]/personnel[1]/person[3]/name[1]/given[1]</code>
Three	<code>/JSON[1]/personnel[1]/person[4]/name[1]/given[1]</code>
Four	<code>/JSON[1]/personnel[1]/person[5]/name[1]/given[1]</code>

Transform using XSLT/XQuery

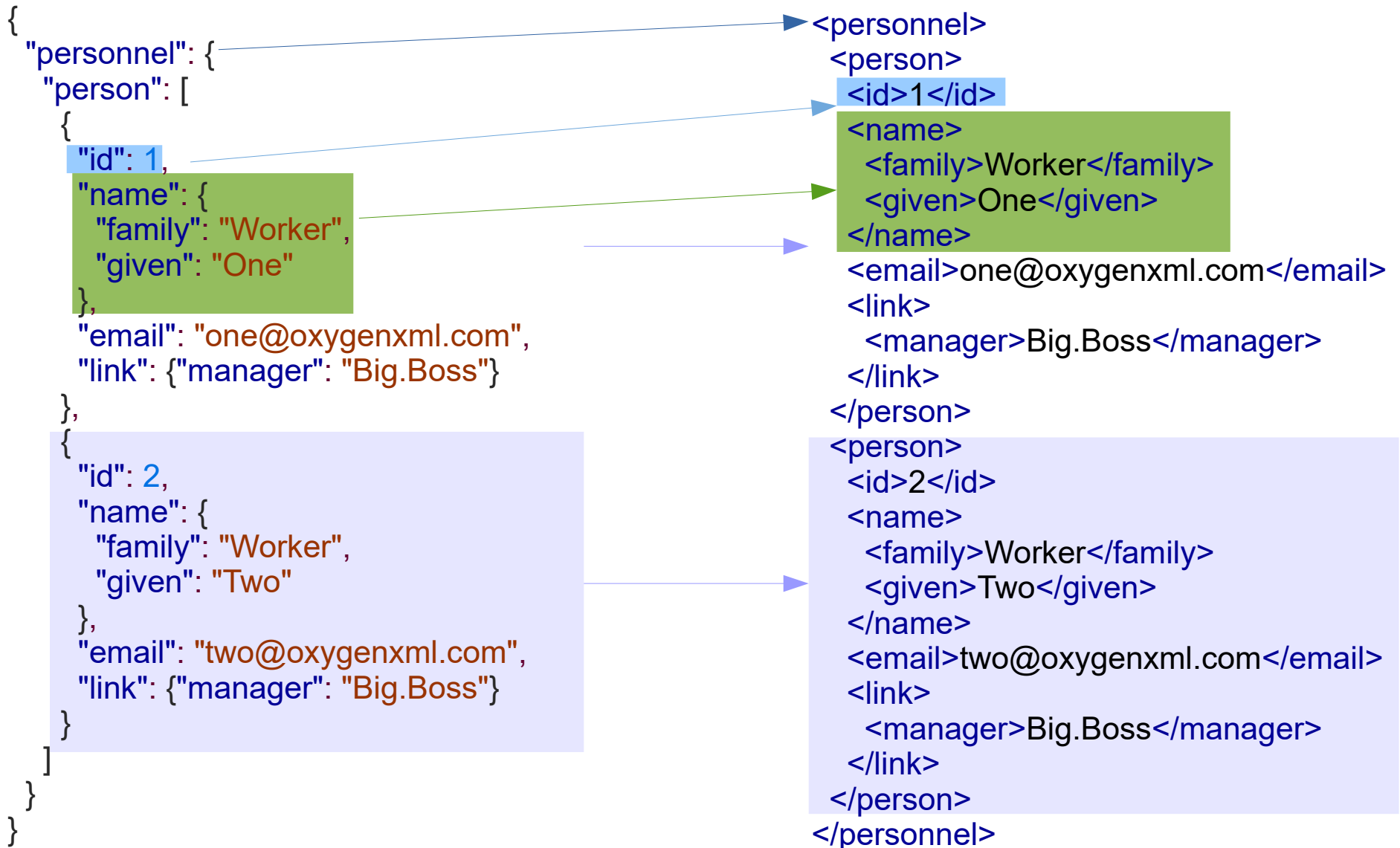
- Transform JSON Documents to Different Formats
- Create XSLT/XQuery Transformation Scenario for JSON
- XQuery Builder view
- Functions to process JSON document:
 - `json-doc($href as xs:string?) as item()?`
 - `json-to-xml($json-text as xs:string?) as document-node()?`

JSON Tools

- Convert between [JSON and XML](#)
- [XSD to JSON Schema converter](#)
- [Instance generator](#) from a JSON Schema
- [Generate JSON Schema](#) from JSON instance



JSON to XML



JSON to XML Conversion Details

- A <JSON> element is added as root if the converted JSON has multiple properties on the first level

```
{  
  "person": "one",  
  "id": "personnel-id"  
}
```

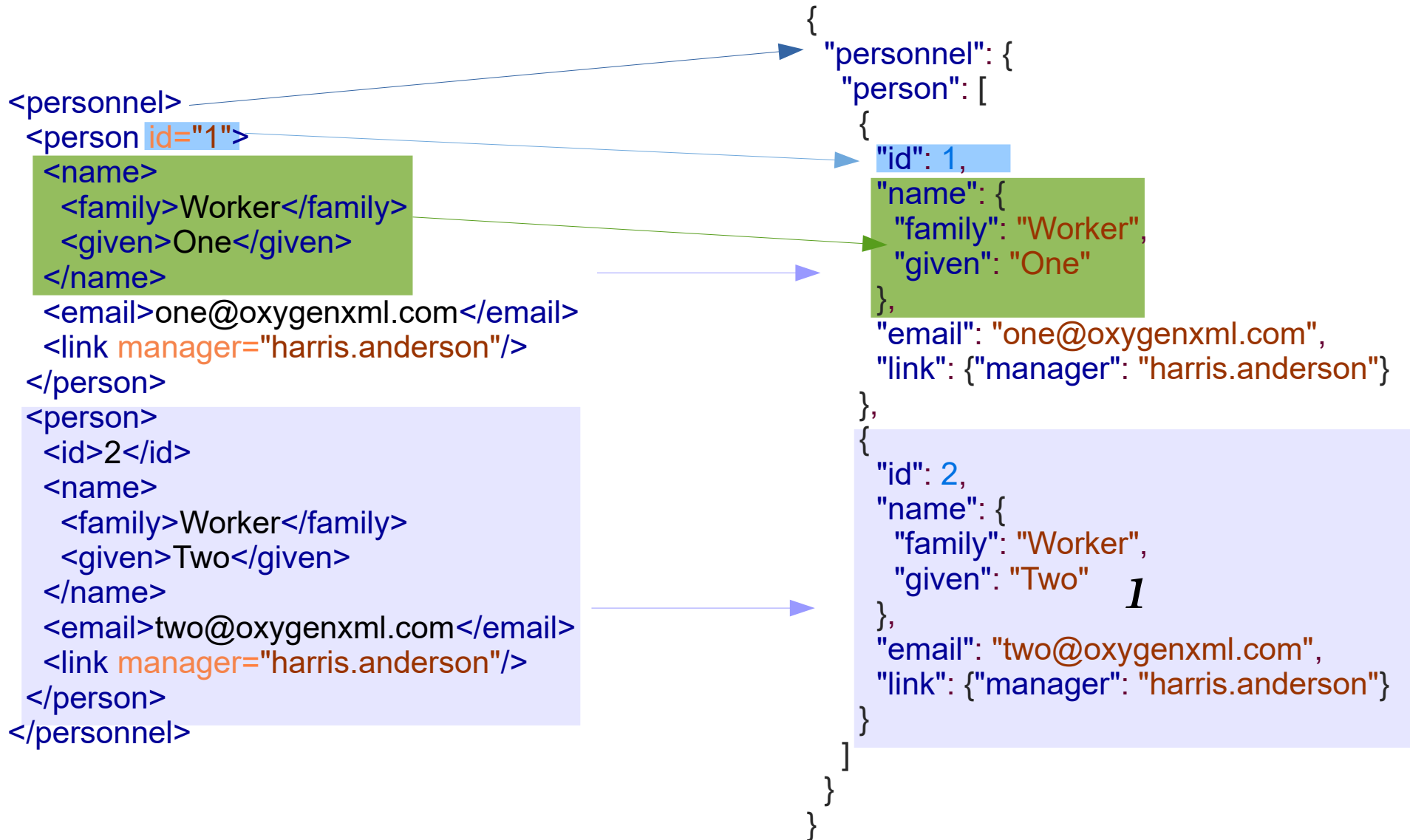
```
<JSON>  
  <person>one</person>  
  <id>personnel-id</id>  
</JSON>
```

- An <array> element is added as root if the converted JSON is an array

```
[  
  {"name": "Boss"},  
  {"name": "Worker"}  
]
```

```
<array>  
  <array>  
    <name>Boss</name>  
  </array>  
  <array>  
    <name>Worker</name>  
  </array>  
</array>
```

XML to JSON



XML to JSON Conversion Details

- XML attributes are converted to properties

```
<person id="1"/>
```

```
{"person": {"id": 1}}
```

- Multiple elements with the same name are converted into an array

```
<person>One</person>
```

```
"person": ["One", "Two"]
```

```
<person>Two</person>
```

- Text in mixed content will be converted in a **#text** property

```
<p>This is an <b>example</b>!</p>
```

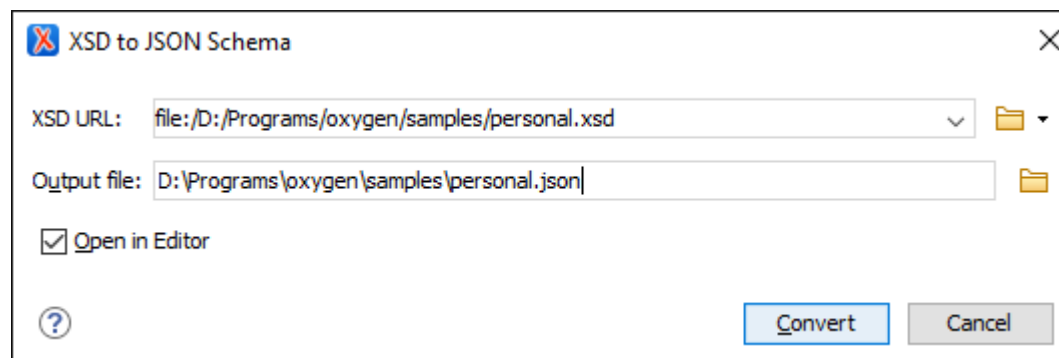
```
p": {"#text": "This is an",
```

```
"b": "example",
```

```
"#text1": "!"}
```

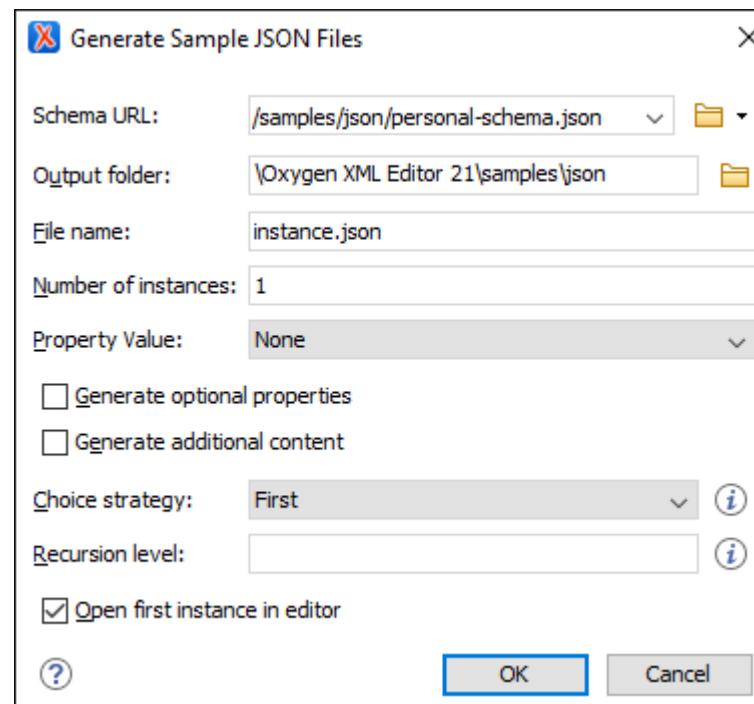
XSD to JSON Schema converter

- Tool for converting an XML Schema file (XSD) to a JSON Schema



Instance generator from a JSON Schema

- Tool for generating sample JSON files from a JSON Schema



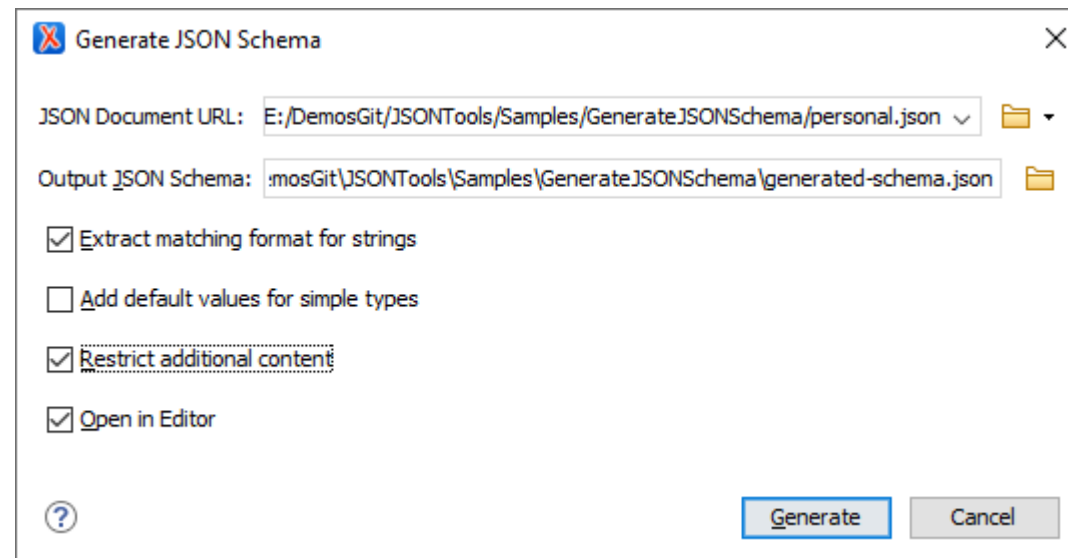
The screenshot shows a dialog box titled "Generate Sample JSON Files" with the following fields and options:

- Schema URL: /samples/json/personal-schema.json
- Output folder: \Oxygen XML Editor 21\samples\json
- File name: instance.json
- Number of instances: 1
- Property Value: None
- Generate optional properties
- Generate additional content
- Choice strategy: First
- Recursion level: (empty)
- Open first instance in editor

Buttons: ? (help), OK, Cancel

Generate JSON Schema from JSON instance

- Tool for generating JSON Schema



Conclusion

- JSON conforms to ECMA/ISO specification
- Validate JSON with JSON Schema and Schematron
- Editing based on JSON Schema
- Query and process JSON using XPath, XSLT/XQuery
- Useful JSON Tools



A long-exposure photograph of a multi-lane highway at night. The image shows vibrant light trails from cars, with white and yellow trails for headlights and red trails for taillights, curving along the road. The background is dark with some distant streetlights and trees.

Questions?

Octavian Nadolu
Software Architect at Syncro Soft

octavian.nadolu@oxygenxml.com
Twitter: [@OctavianNadolu](https://twitter.com/OctavianNadolu)
LinkedIn: [octaviannadolu](https://www.linkedin.com/in/octaviannadolu)