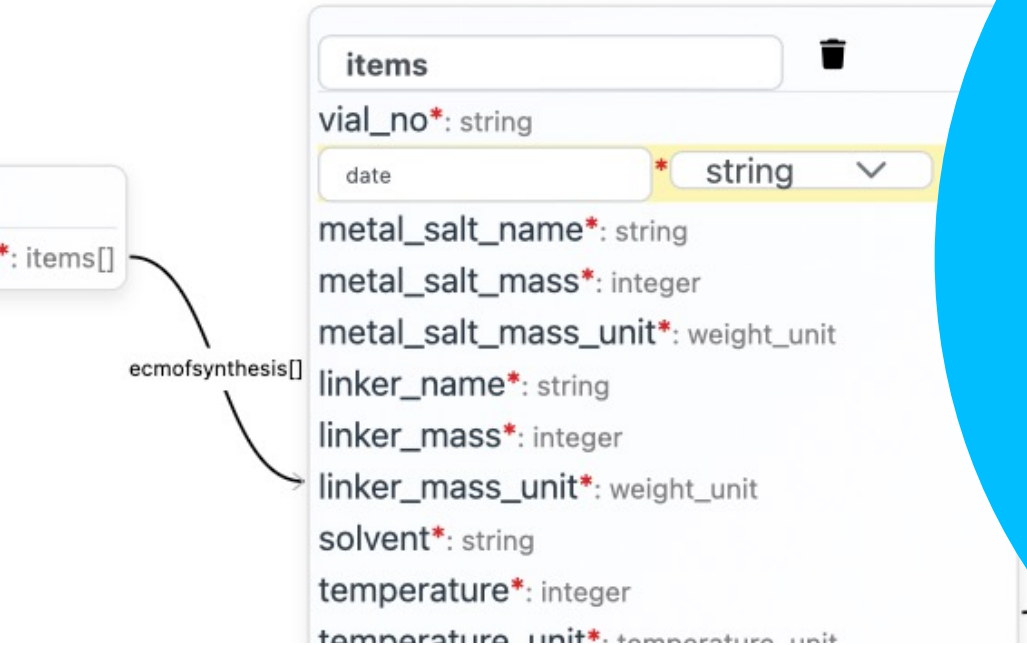
University of Stuttgart
Institute for Parallel and Distributed Systems



Data Model Creation with MetaConfigurator

Felix Neubauer

The Team

Initial Student Project Team

Felix Neubauer

Continues the Project in PhD

Paul Bredl

Keyuri Patel

Minye Xu

Supervision

Jürgen Pleiss

Benjamin Uekermann

Partners

preCICE

SimTech

Cluster of Excellence SimTech,
Stuttgart Center for Simulation Science

IBTB

University of Stuttgart, Institute for
Biochemistry and Technical Biochemistry

Max Planck Institute for Solid State Research

“Have you ever struggled with data interoperability?”

Content

- Data Models (what is it and why does it matter?)
- The Problem
- The Solution
- Live Demo
- Conclusion

What is a Data Model?

A data model defines the structure and constraints of data

```
{
  "name": "Alex"
}
```

```
{
  "name": "Alex",
  "age": -10
}
```

```
{
  "name": "Alex",
  "age": 10
}
```

```
{
  "$schema": "http://json-schema.org/
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer",
      "minimum": 0
    }
  },
  "required": [
    "name",
    "age"
  ]
}
```

Why Data Models Matter

WITHOUT A DATA MODEL

- Inconsistent structures
- Data duplication
- Hard to validate
- Manual effort needed

→ No FAIR[1]ness

WITH A DATA MODEL

- Defined structure
- Interoperability
- Automatic validation
- Easier analysis and automation

→ Enables FAIR[1]ness

[1]: Findable, Accessible, Interoperable and Re-Usable [1]

[1]: WILKINSON, Mark D., et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 2016, 3. Jg., Nr. 1, S. 1-9.

Challenges in Creating Data Models

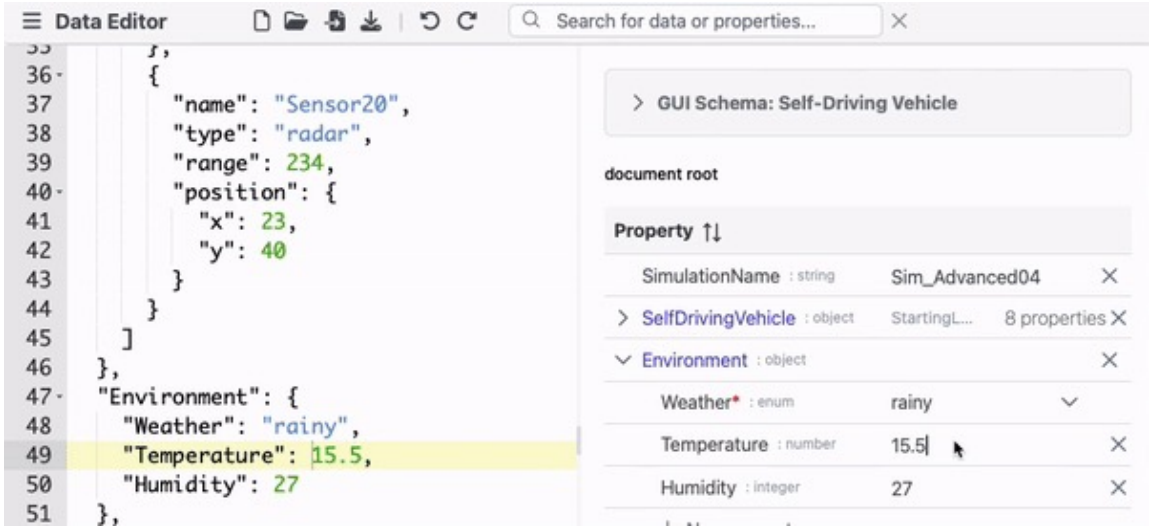
High technical barrier (e.g., JSON Schema syntax)

Lack of easy-to-use tools

Fragmentation between different teams and organizations

Introducing MetaConfigurator

MetaConfigurator [2] is an open-source tool for creating and editing data models (schemas) and data for JSON/YAML.



[2]: NEUBAUER, Felix, et al. MetaConfigurator: A User-Friendly Tool for Editing Structured Data Files. *Datenbank-Spektrum*, 2024, S. 1-9.

LIVE DEMO

www.metaconfigurator.org

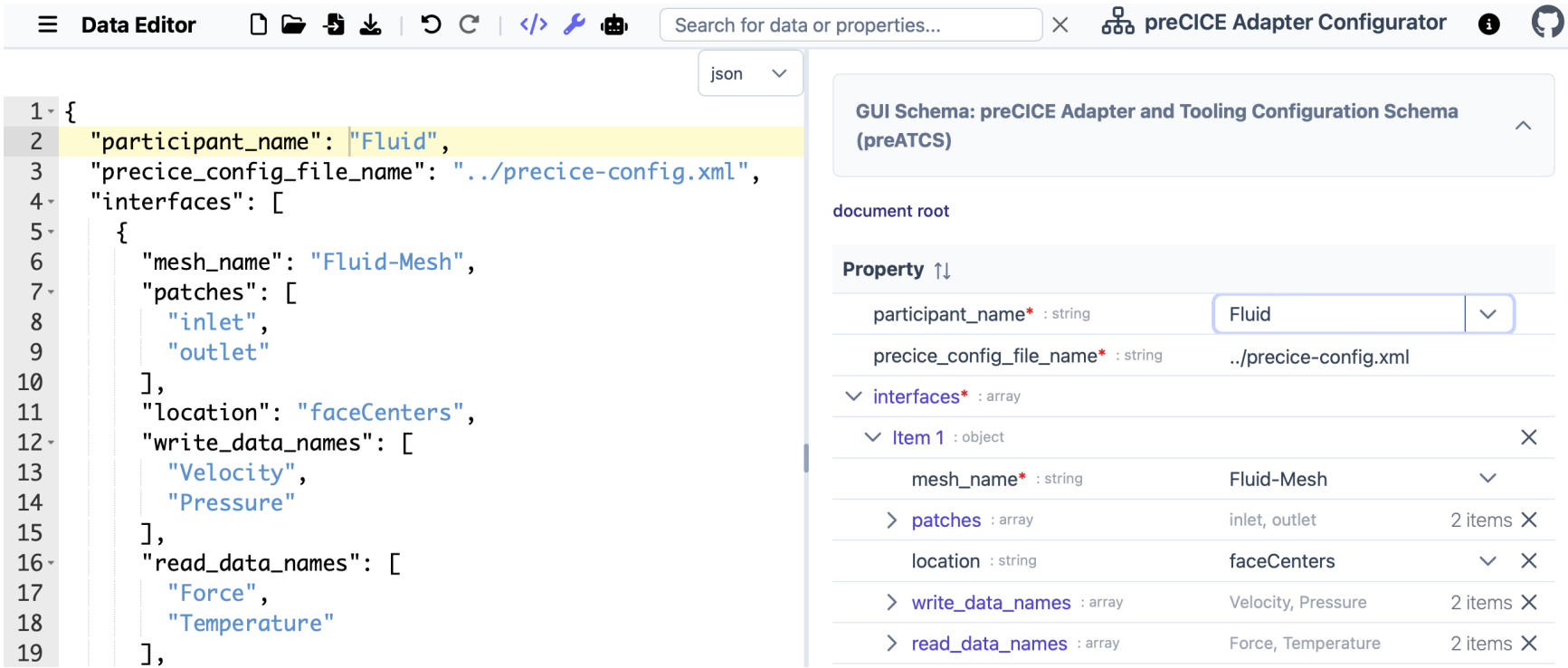
- With a dataset[2] from Chemistry: Metal Organic Framework (MOF) synthesis
- They put chemicals together in certain conditions to produce new chemical
- Goal: reach certain characteristics in produced chemical
- Big dataset in Excel

[2]: Dataset provided by Esengül Çiftçi, Max Planck Institute for Solid State Research

What to do with our new Data Model?

- Formal specification of our desired data structure
- Can be communicated or published
- Automatic validation
- Code generation
- Script that processes the data
  - Enriching the compound entries with chemical information from PubChem
  - Learning algorithm to optimize the synthesis

Example 2: preCICE Adapter Configurator



Future Directions

- Generating JSON Schema from an Ontology
- Generation of source code from schemas (e.g. Python API)
- AI integration
- Large scale user study
- More documentation, tutorials

- Feedback & Early Adopters are very welcome!

Takeaways

- Data models enable interoperability, validation and integration of data across domains and teams → making data more FAIR
- MetaConfigurator supports in creation and maintenance of data models

Try it out yourself!

https://github.com/MetaConfigurator/meta-configurator/

University of Stuttgart
Institute for Parallel and Distributed Systems

Thank you!

Felix Neubauer

e-mail felix.neubauer@ipvs.uni-stuttgart.de
www.logende.org

University of Stuttgart
Institute for Parallel and Distributed Systems
Universitätsstraße 38, 70569 Stuttgart,

https://github.com/MetaConfigurator/meta-configurator/

Backup
Data Formats

```json
{
  "SelfDrivingVehicle": {
    "StartingLocation": {⟺},
    "Destination": {⟺},
    "PlanningAlgorithm": "Dijkstra",
    "VehicleType": "Level 3",
    "PassengerCapacity": 5,
    "MaxSpeed": 180,
    "is4-Wheel-Drive": false,
    "Sensors": [⟺]
  },
  "Environment": {
    "Weather": "rainy",
    "Temperature": 15.5,
    "Humidity": 27
  },
  "Vehicles": {⟺},
  "PedestrianGroups": [⟺],
  "Waypoints": {⟺},
  "SimulationSettings": {
    "Duration": 260,
    "TimeUnit": "seconds"
  }
}
```

JSON

```yaml
SelfDrivingVehicle:
  StartingLocation:⟺
  Destination:⟺
  PlanningAlgorithm: Dijkstra
  VehicleType: Level 3
  PassengerCapacity: 5
  MaxSpeed: 180
  is4-Wheel-Drive: false
  Sensors:⟺
Environment:
  Weather: rainy
  Temperature: 15.5
  Humidity: 27
Vehicles:⟺
PedestrianGroups:⟺
Waypoints:⟺
SimulationSettings:
  Duration: 260
  TimeUnit: seconds
```

YAML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <SimulationName>Sim_Advanced04</SimulationNam
  <SelfDrivingVehicle>
    <StartingLocation>
    <Destination>
    <PlanningAlgorithm>Dijkstra</PlanningAlgori
    <VehicleType>Level 3</VehicleType>
    <PassengerCapacity>5</PassengerCapacity>
    <MaxSpeed>180</MaxSpeed>
    <is4-Wheel-Drive>false</is4-Wheel-Drive>
    <Sensors>
  </SelfDrivingVehicle>
  <Environment>
    <Weather>rainy</Weather>
    <Temperature>15.5</Temperature>
    <Humidity>27</Humidity>
  </Environment>
  <Vehicles>
  <PedestrianGroups>
  <Waypoints>
  <SimulationSettings>
    <Duration>260</Duration>
    <TimeUnit>seconds</TimeUnit>
  </SimulationSettings>
</root>
```

XML

Backup
Data Models (= Schemas)

- A schema defines…
  - the structure to which data needs to adhere to
  - constraints, conditions, etc.
  - metadata (e.g., attribute descriptions)
- It enables…
  - data validation
  - communication of data models
  - common tooling

```
"Environment": {
  "type": "object",
  "properties": {
    "Weather": {
      "type": "string",
      "description": "Current weather conditions",
      "enum": [
        "sunny",
        "rainy",
        "cloudy"
      ]
    },
    "Temperature": {
      "type": "number",
      "description": "Current temperature in Celsius."
    },
    "Humidity": {
      "type": "integer",
      "description": "Relative humidity as a percentage.",
      "minimum": 0,
      "maximum": 100
    }
  },
```

Backup
Schema Language Comparison

TABLE I
EVALUATION OF DIFFERENT SCHEMA LANGUAGES

Schema language	# Search results	IDE support	# Node packages	Expressiveness
JSON schema	245,000	8 / 10	4,536	9 /9
XSD	151,000	8 / 10	116	8 / 9
DTD	69,700	9 / 10	34	6 / 9
CUE	10,500	4 / 10	97	8 / 9
Avro	20,000	8 / 10	211	5 / 9
JSON Type Definition (JTD)	109	0 / 10	17	5 / 9
TypeSchema	8,450	0 / 10	5	8 / 9
Protobuf	44,800	9 / 10	1,210	4 / 9
GraphQl schema	31,000	7 / 10	1,509	6 / 9

TABLE II
COMPARISON OF EXPRESSIVENESS OF DIFFERENT SCHEMA LANGUAGES

Schema language	Simple types	Complex types	Descriptions	Example values	Default values	Optional values	Constraints	Conditions	References	Result
JSON schema	✓	✓	✓	✓	✓	✓	✓	✓	✓	9 / 9
XSD	✓	✓	✓	x	✓	✓	✓	✓	✓	8 / 9
DTD	✓	✓	x	x	✓	✓	x	✓	✓	6 / 9
CUE	✓	✓	✓	✓	x	✓	✓	✓	✓	8 / 9
Avro	✓	✓	x	x	✓	✓	x	x	✓	5 / 9
JTD	✓	✓	x	x	x	✓	x	✓	✓	5 / 9
TypeSchema	✓	✓	✓	x	✓	✓	✓	✓	✓	8 / 9
protobuf	✓	✓	x	x	x	✓	x	x	✓	4 / 9
GraphQL schema	✓	✓	✓	x	✓	✓	x	x	✓	6 / 9

MetaConfigurator: A User-Friendly Tool for Editing Structured Data Files

Felix Neubauer[1] · Paul Bredl[1] · Minye Xu[1] · Keyuriben Patel[1] · Jürgen Pleiss[2] · Benjamin Uekermann[1]

Received: 31 January 2024 / Accepted: 26 April 2024 / Published online: 30 May 2024

Abstract
Textual formats to structure data, such as JSON, XML, and YAML, are widely used for structuring data in various domains, from configuration files to research data. However, manually editing data in these formats can be complex and time-consuming. Graphical user interfaces (GUIs) can significantly reduce manual efforts and assist the user in editing the files, but developing a file-format-specific GUI requires substantial development and maintenance efforts. To address this challenge, we introduce *MetaConfigurator*: an open-source web application that generates its GUI depending on a given schema. Our approach differs from other schema-to-UI approaches in three key ways: 1) It offers a unified view that combines the benefits of both GUIs and text editors, 2) it enables schema editing within the same tool, and 3) it supports advanced schema features, including conditions and constraints. In this paper, we discuss the design and implementation of *MetaConfigurator*, backed by insights from a small-scale qualitative user study. The results indicate the effectiveness of our approach in retrieving information from data and schemas and in editing them.

Keywords Rdm · Json · Yaml · Schema · Editor · Configuration · Data · Research data management · Tool · Gui

Backup
Related Work

- JSON Schema related research: schema inference [3, 4], validation [5], witness generation [6], schema matching [7], etc.

[3]: Čontoš, Pavel; Svoboda, Martin: JSON schema inference approaches. In: International Conference on Conceptual Modeling. Springer, pp. 173–183, 2020.

[4]: Baazizi, Mohamed-Amine; Colazzo, Dario; Ghelli, Giorgio; Sartiani, Carlo: Parametric schema inference for massive JSON datasets. The VLDB Journal, 28:497–521, 2019.

[5]: Attouche, Lyes; Baazizi, Mohamed-Amine; Colazzo, Dario; Ghelli, Giorgio; Sartiani, Carlo; Scherzinger, Stefanie: Validation of modern JSON schema: Formalization and complexity. Proceedings of the ACM on Programming Languages, 8(POPL):1451–1481, 2024.

[6]: Attouche, Lyes; Baazizi, Mohamed-Amine; Colazzo, Dario; Ghelli, Giorgio; Sartiani, Carlo; Scherzinger, Stefanie: Witness generation for JSON Schema. arXiv preprint arXiv:2202.12849, 2022.

[7]: Waghray, Kunal: JSON schema matching: Empirical observations. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. pp. 2887–2889, 2020.

Backup
Related Work

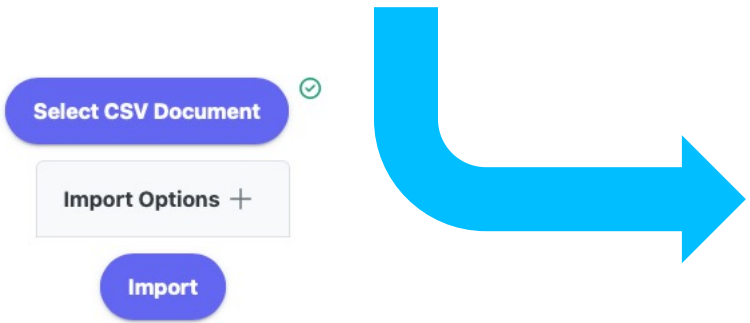
- Schema Editors: there exist different tools, such as JSON Editor Online[2], Adamant [8], or paid software, however less user-supporting features, such as tooltips or auto-completion or less JSON schema features supported

[3]: https://jsoneditoronline.org, accessed 2025/01/20.

[8]: Siffa, Ihda Chaerony; Schäfer, Jan; Becker, Markus M: Adamant: a JSON schema-based metadata editor for research data management workflows. F1000Research, 11, 2022.

Backup
Feature: CSV Import

A	B	C	D	E	F	G	H	I
Vial no	Date	Ligand type	Ligand mass	Ligand mass unit	Metal salt type	Metal salt mass	Metal salt mass unit	Solvent
ESN-1	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF
ESN-2	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF
ESN-3	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF
ESN-4	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml water
ESN-5	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml water
ESN-6	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml water
ESN-7	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml Hac
ESN-8	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml Hac
ESN-9	03/01/22	Benzenedicarboxylic acid	66	mg	FeCl3	65	mg	4 ml DMF+0,1 ml Hac
ESN-10	05/02/22	Fumaric acid	46	mg	FeCl3	65	mg	4 ml DMF
ESN-11	05/02/22	Fumaric acid	46	mg	FeCl3	65	mg	4 ml DMF+0,1 ml HAc
ESN-12	05/02/22	Fumaric acid	46	mg	FeCl3	65	mg	4 ml DMF
ESN-13	05/02/22	Biphenyl-4,4'-dicarboxylic acid	10	mg	FeCl3	6	mg	3 ml DMF
ESN-14	05/02/22	[1,1':4',1''-Terphenyl]-4,4''-dicarboxylic acid	10	mg	FeCl3	5	mg	4 ml DMF+0,1 ml HAc
ESN-15	05/16/22	Benzenedicarboxylic acid	16	mg	VCl3	50	mg	5 ml EtOH+1 ml HCl(1 M)
ESN-16	05/16/22	Benzenedicarboxylic acid	16	mg	Sc(NO3)3 H2O	25	mg	2 ml DMF+2 ml DEF
ESN-17	05/16/22	Fumaric acid	12	mg	Sc(NO3)3 H2O	25	mg	2 ml DMF+2 ml DEF
ESN-18	05/16/22	Fumaric acid	12	mg	VCl3	33	mg	5 ml EtOH+1 ml HCl(1 M)
ESN-19	05/19/22	Benzenedicarboxylic acid	50	mg	VCl3	47	mg	3 ml DMF+0,1 ml HAc
ESN-20	05/19/22	Benzenedicarboxylic acid	50	mg	VCl3	47	mg	3 ml DMF+0,1 ml water




```
{
  "mof_synthesis": [
    {
      "vial_no": "ESN-1",
      "date": "03.01.22",
      "ligand_type": "Benzenedicarboxylic acid",
      "ligand_mass": 66,
      "ligand_mass_unit": "mg",
      "metal_salt_type": "FeCl3",
      "metal_salt_mass": 65,
      "metal_salt_mass_unit": "mg",
      "solvent": "4 ml DMF",
      "temperature": 120,
      "temperature_unit": "deg C",
      "time": 1,
      "time_unit": "day",
      "reaction_in": "oven",
      "mof": "MIL-88B (Fe)",
      "results": "orange powder"
    },
    {
      "vial_no": "ESN-2",
      "date": "03.01.22",
      "ligand_type": "Benzenedicarboxylic acid",
      "ligand_mass": 66,
      "ligand_mass_unit": "mg",
      "metal_salt_type": "FeCl3",
      "metal_salt_mass": 65,
      "metal_salt_mass_unit": "mg",
      "solvent": "4 ml DMF",
      "temperature": 120,
      "temperature_unit": "deg C",
      "time": 2,
      "time_unit": "hour",
      "reaction_in": "heat blok",
      "mof": "MIL-88B (Fe)",
      "results": "orange powder"
    },
```

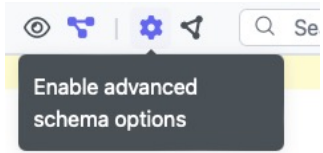

[...]

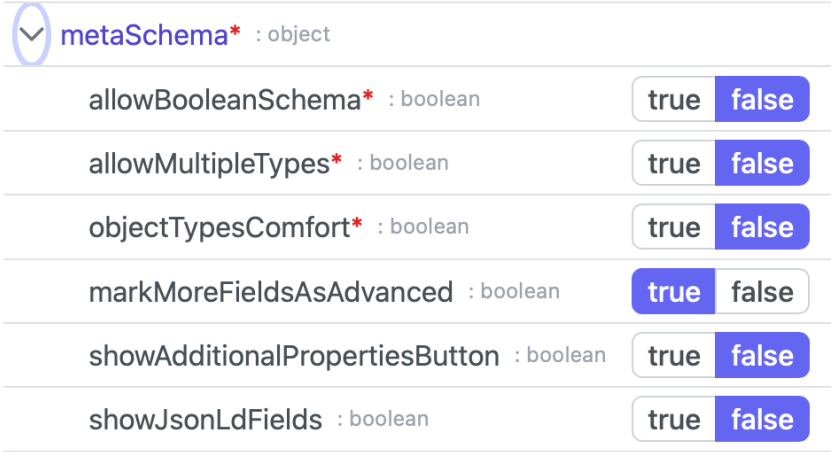
Backup

Feature: Meta Schema Builder

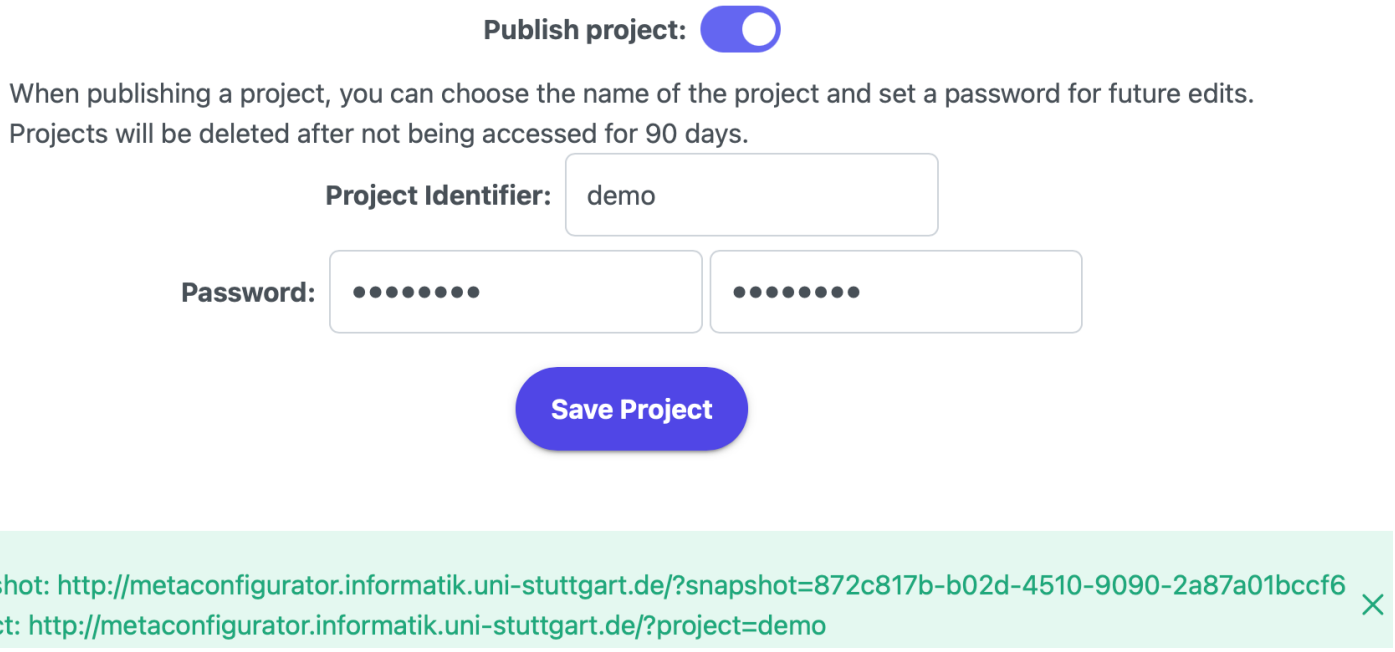
- Dynamically builds meta schema based on user settings
- Meta schema is responsible for complexity of schema editor

→ Advanced mode and beginner mode for schema editor

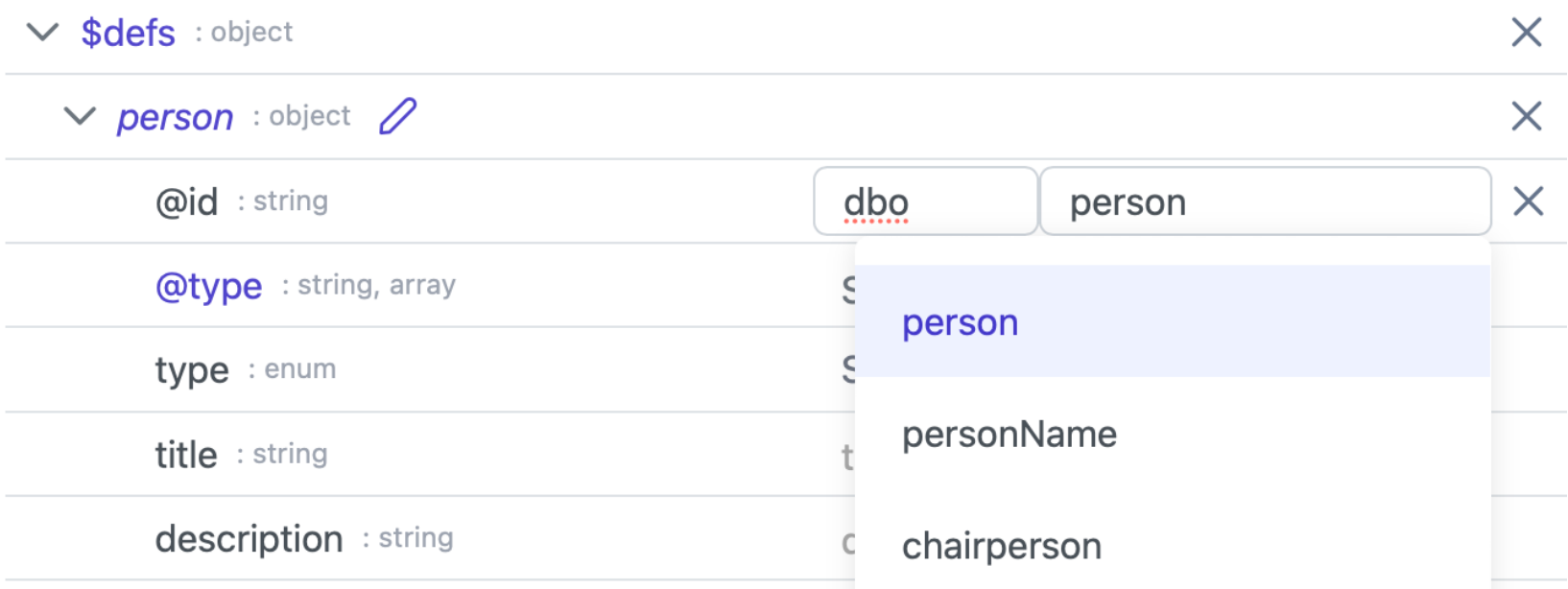




Backup
Feature: Snapshot Sharing



Backup
Feature: Ontology Integration



Backup

Application Case #1: MOF Synthesis (in Chemistry)

- Scenario from Biochemistry department[1]
- About Metal-organic framework (MOF) synthesis data

1. Export Excel table[2] as CSV
2. Import of CSV into MetaConfigurator (+ automatic schema inference)
3. Changes in schema through intuitive schema editor diagram
4. External Python script[3] for appending metadata to the synthesis data

→ Now data is in standardized format, interoperable. Schema can be shared with others, automatic validation possible. Easier maintenance of data through tools, such as MetaConfigurator.

1
2 In cooperation with Jürgen Pleiss
3 The Excel document with synthesis data was provided by Esengül Ciftci
The Python script queries data from the PubChem database, based on code from Torsten Gieß

Backup

Application Case #2: preCICE Adapter Configuration Schema

- preCICE is a coupling library for partitioned multi-physics simulations[1]
- It has more than 10 different configuration files for different adapters and tools
- Some are solver-specific but many more or less the same
- Introduction of common Adapter Schema using MetaConfigurator

→ MetaConfigurator offers up-to-date web form showing the fields of the schema in discussion & allows for editing the schema (without prior experience needed)

Backup

User Study – Research Questions

- **RQ1:** Which aspects of the tool can be improved?
- **RQ2:** Are users able to perform particular types of tasks using the tool?
- **RQ3:** Would people use the tool in practice?
- GOAL:
  - To improve our tool with help of feedback
  - To gather information about user habits

Backup
User Study - Methodology

- Small-scale: 5 Participants
- Qualitative
- Participants needed to solve example tasks:
  - Retrieving information
  - Editing data
  - Editing the schema
- Participants were asked to share their opinion and improvement suggestions

Backup
User Study - Results

- **RQ1:** Which aspects of the tool can be improved?
  - 23 aspects identified: 20 addressed, 3 remain as future work
- **RQ2:** Are users able to perform particular types of tasks using the tool?
  - 90% of the tasks correctly without hints
  - 100% of the tasks solved with giving small hints
- **RQ3:** Would people use the tool in practice?
  - All participants responded they would potentially use MetaConfigutator in practice
  - Some recommended our tool to other people