Aalto University
School of Science

UNIVERSITY OF HELSINKI

SeCo

HELDIG
Helsinki Centre for Digital Humanities

Creating a semantic portal easily using Sampo-UI

Annastiina Ahola, Heikki Rantala

Sampo-UI framework

- **Framework for developing user interfaces for semantic portals**
  - multiple application perspectives to the data
  - two-step usage cycle: filter & analyze
  - has been used both in the Sampo series of portals as well as external projects
- **Available on GitHub under the MIT license: https://github.com/SemanticComputing/sampo-ui**

Philosophy

- **Faceted searching done with SPARQL**
  - Sampo-UI doesn’t require your triple store to implement a certain kind of text indexing to work
    - *Text indexing can be utilized for efficient text-based search*
- **Configuration primarily done through JSON configuration and SPARQL query files without having to touch JavaScript**

Sampo-UI “standard” UI model

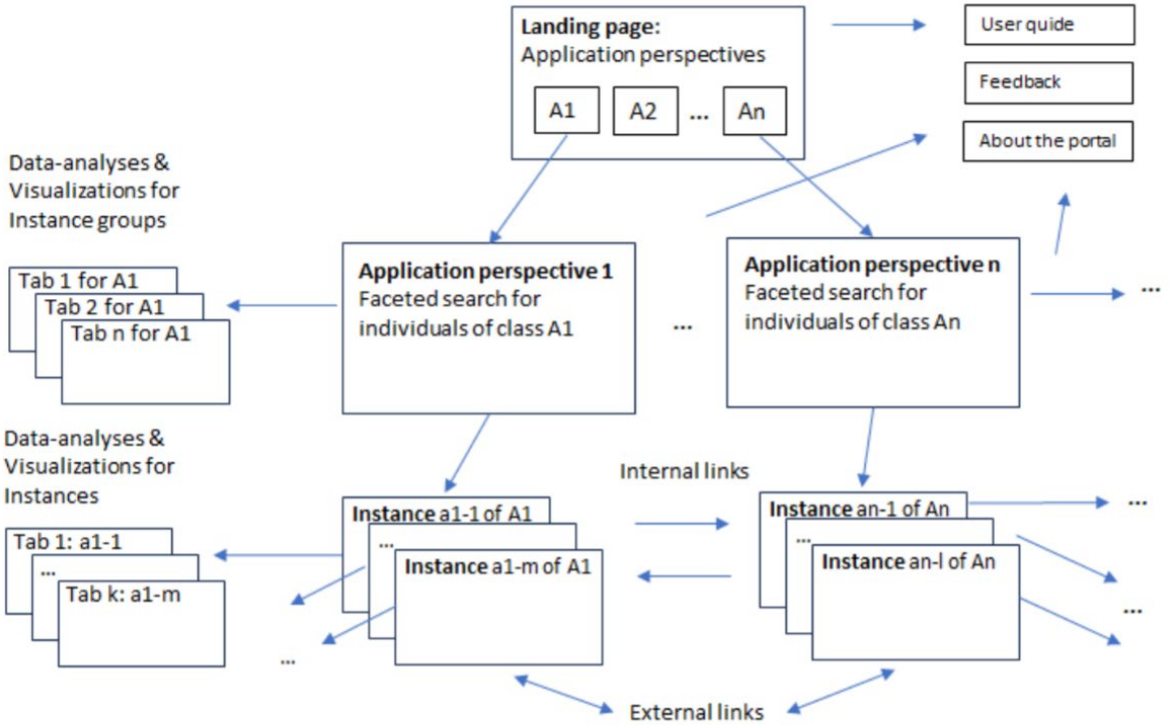
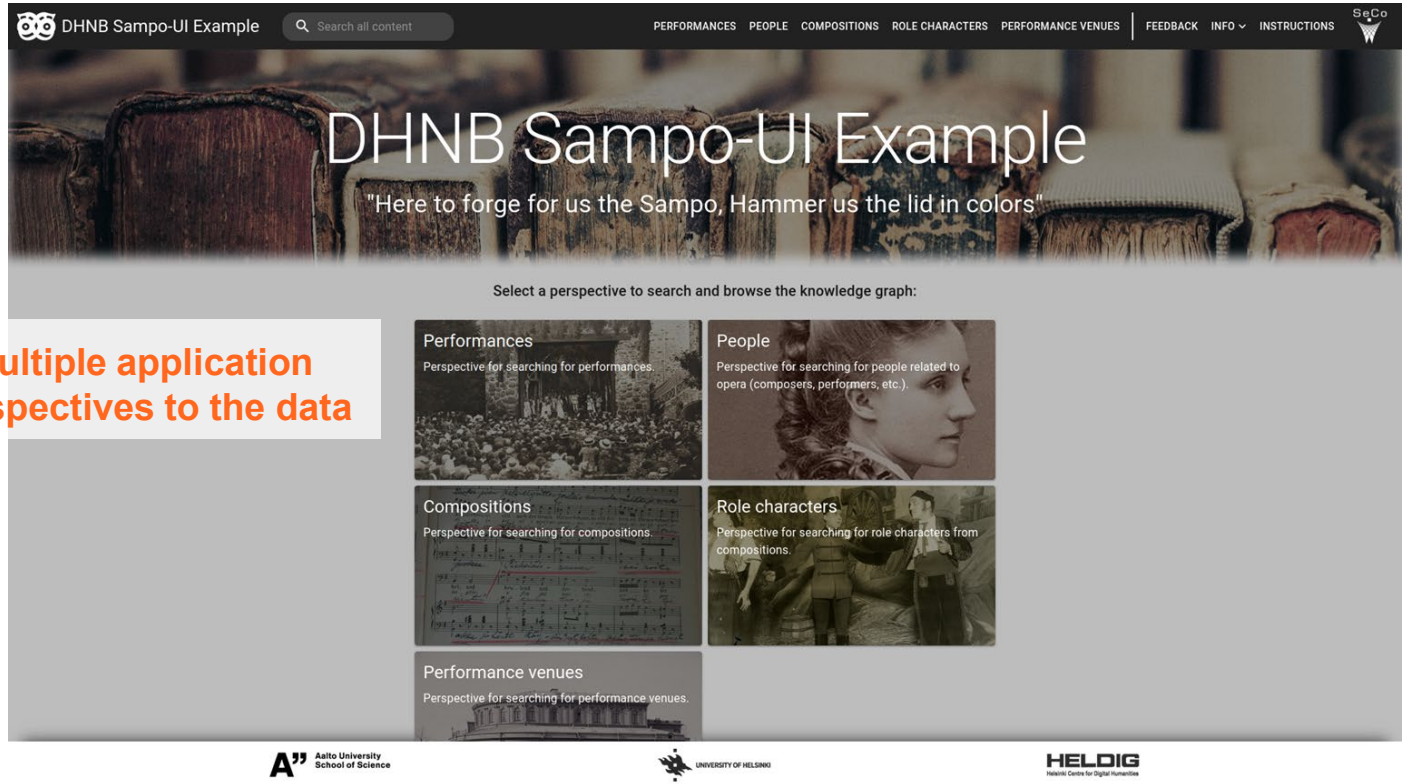


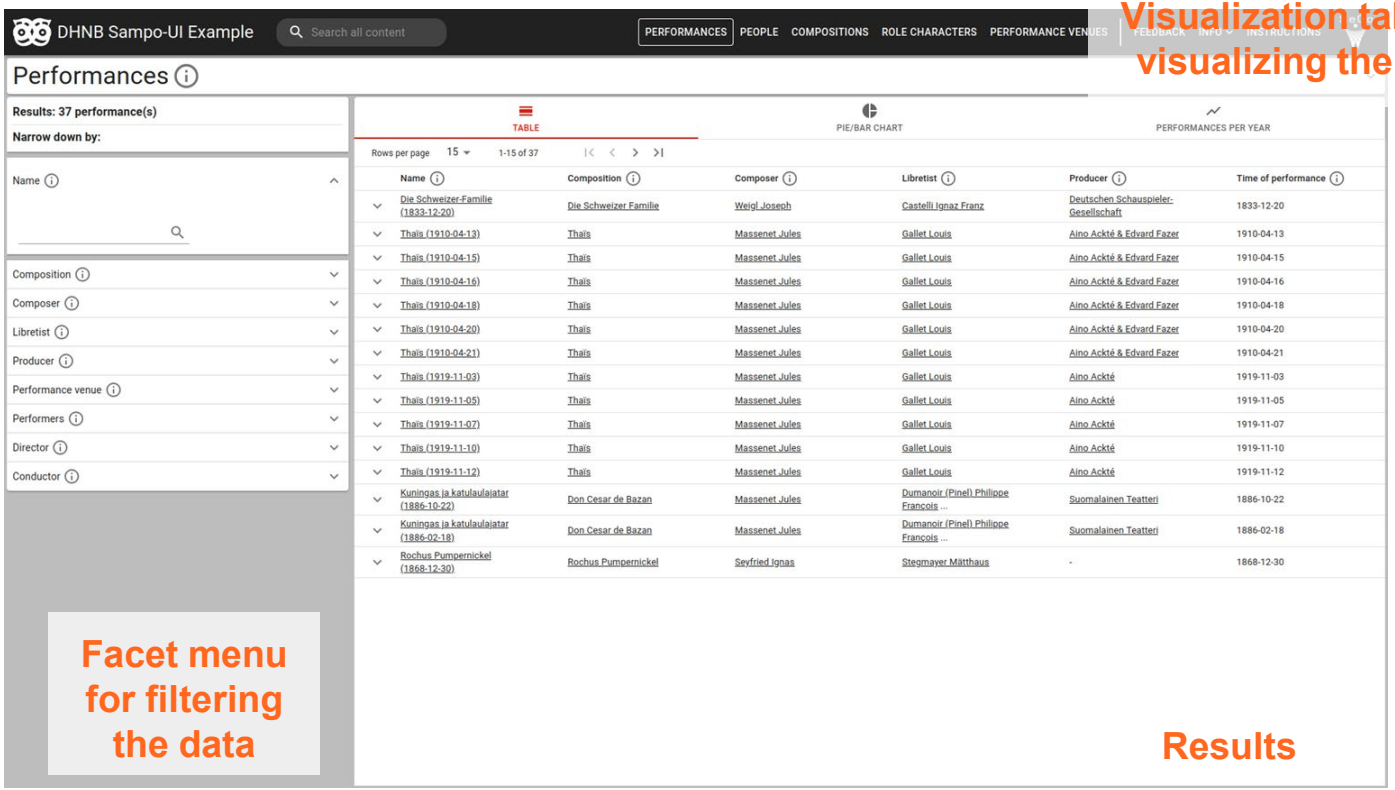
Figure 1: Navigational page structure of a portal based on Sampo-UI.

Sampo-UI framework: Structure

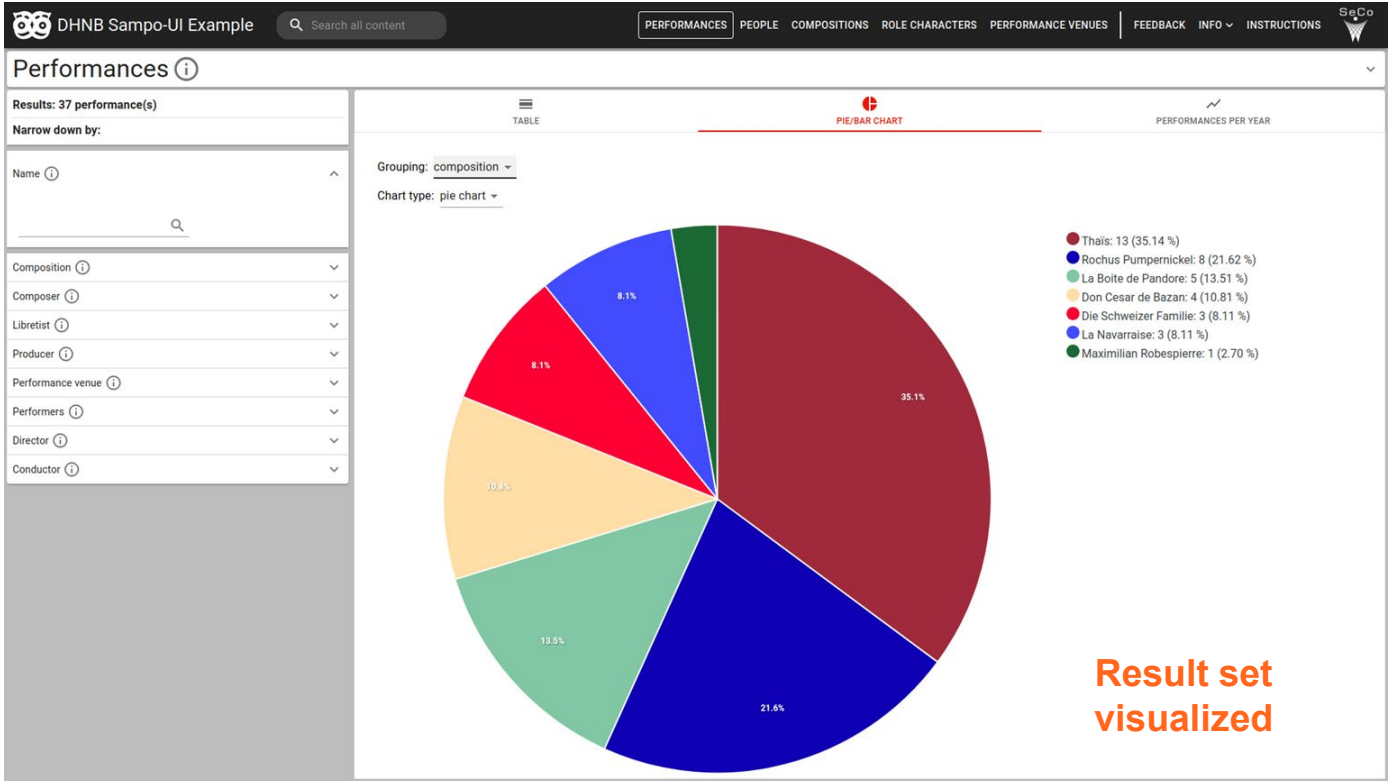
Multiple application perspectives to the data



Sampo-UI framework: Structure



Sampo-UI framework: Structure



Result set visualized

Written & video tutorial

- **Tutorial showing the steps of setting up Sampo-UI using DBpedia's endpoint/data**
  - In video format: https://vimeo.com/817028609
  - Written tutorial: https://seco.cs.aalto.fi/tools/sampo-ui/Sampo-UI-tutorial.pdf
  - Code from the tutorial available on GitHub: https://github.com/SemanticComputing/dbpedia-sampo-ui-demo

Tutorial Sampo-UI setup

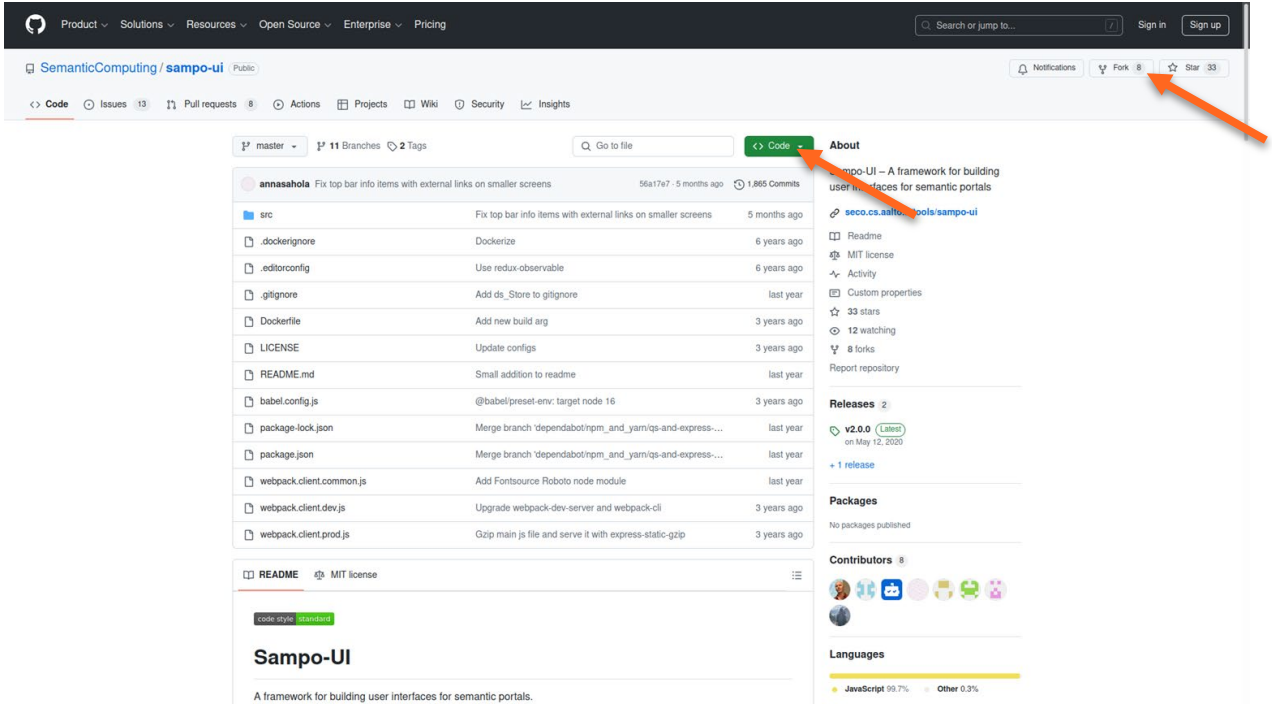
- **Sampo-UI setup for running it with the data from “Creating LOD from databases” (subset of OperaSampo data)**
  - 5 search perspectives: performances, people, compositions, role characters and performance venues
  - 1 perspective with only instance pages: producers
- **Data and scripts as well as a portal setup available on GitHub: https://github.com/SemanticComputing/sampo-lod-tutorial**
  - Sampo-UI setup in the ‘set-up-sampo-ui’ folder: https://github.com/SemanticComputing/sampo-lod-tutorial/tree/main/set-up-sampo-ui

Requirements

- **Node.js (https://nodejs.org/en/) version 16.13.0**
  - Recommended: Node Version Manager (nvm) for managing different versions of Node.js installed on your computer
- **Nodemon (https://nodemon.io/)**
- **Optional: Docker**
  - You can also run Sampo-UI inside a Docker container on your computer, but this is not required for development

Getting started

- **Fork or download source code from GitHub**



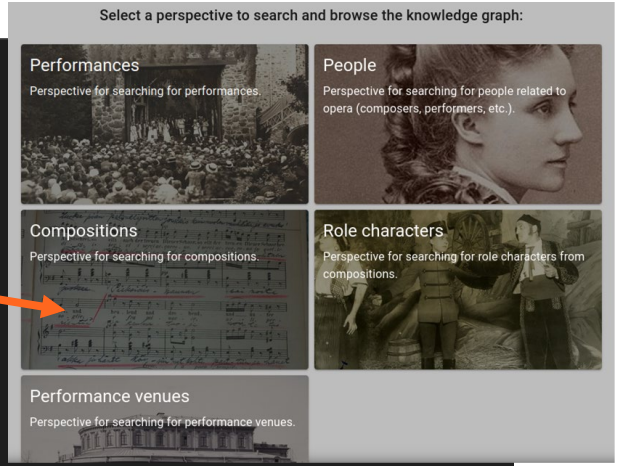
Configuration

- **Configuration done through JSON configuration files**
  - a file for portal-wide configuration
  - one file for each configured perspective
- **Queries with SPARQL**
  - one file for each perspective with the perspective-specific queries for getting all wanted properties
  - many general queries already exist (e.g., getting facet values), you just need to pass the correct predicates in configuration

Portal configuration

```
{
    "portalID": "sampo",
    "rootUrl": "",
    "perspectives": {
        "searchPerspectives": [
            "performances",
            "people",
            "compositions",
            "roles",
            "places"
        ],
        "onlyInstancePages": [
            "producers"
        ]
    },
    "localeConfig": {
        "defaultLocale": "en",
        "readTranslationsFromGoogleSheets": false,
        "availableLocales": [
            {
                "id": "en",
                "label": "English",
                "filename": "localeEN.json"
            }
        ]
    },
    "sitemapConfig": {
        "baseUrl": "https://sampo-ui.demo.seco.cs.aalto.fi",
        "langPrimary": "en",
        "langSecondary": "fi",
        "outputDir": "./src/server/sitemap_generator",
        "sitemapUrl": "https://sampo-ui.demo.seco.cs.aalto.fi/sitemap",
        "sitemapInstancePageQuery": "sitemapInstancePageQuery"
    },
```

Configure available perspectives by ID



Configure available locales and files for getting the required translations

Perspective configuration: Basics

```
"id": "performances",
"endpoint": {
    "url": "http://localhost:3048/ds/sparql",
    "useAuth": false,
    "prefixesFile": "SparqlQueriesPrefixes.js"
},
"sparqlQueriesFile": "SparqlQueriesPerformances.js",
"baseURI": "http://ldf.fi/operasampo/",
"URITemplate": "<BASE_URI><LOCAL_ID>",
"facetClass": "scop:Performance",
"frontPageImage": "main_page/performances-card.jpg",
"searchMode": "faceted-search",
"defaultActiveFacets": [
    "prefLabel"
],
"defaultTab": "table",
"defaultInstancePageTab": "table",
"resultClasses": {
    "performances": {
        "paginatedResultsConfig": {
            "tabID": 0,
            "component": "ResultTable",
            "tabPath": "table",
            "tabIcon": "CalendarViewDay",
            "propertiesQueryBlock": "performanceProperties",
            "pagesize": 15,
            "sortBy": null,
            "sortDirection": null
        },
```

Perspective ID (referenced in portal configuration)

Configuration for the endpoint used in the perspective and what prefixes should be used in queries

Name of file containing the SPARQL queries for the perspective

Class of the instances included in the perspective

Variable name for result table query in query file

Perspective configuration: Queries

```
const perspectiveID = 'performances'

export const performanceProperties = `
    {
      ?id skos:prefLabel ?prefLabel__id .
      BIND(?prefLabel__id AS ?prefLabel__prefLabel)
      BIND(CONCAT("/${perspectiveID}/page/", REPLACE(STR(?id), "^.*\\\\/(.+)", "$1")) AS ?prefLabel__dataProviderUrl)
      BIND(?id as ?uri__id)
      BIND(?id as ?uri__dataProviderUrl)
      BIND(?id as ?uri__prefLabel)
    }
    UNION
    {
      ?id scop:composition ?composition__id .
      ?composition__id skos:prefLabel ?composition__prefLabel .
      FILTER(LANG(?composition__prefLabel) = 'fi')
      BIND(CONCAT("/compositions/page/", REPLACE(STR(?composition__id), "^.*\\\\/(.+)", "$1")) AS ?composition__dataProviderUrl)
    }
    UNION
    {
      ?id scop:composition/scop:composedBy ?composer__id .
      ?composer__id skos:prefLabel ?composer__prefLabel .
      BIND(CONCAT("/people/page/", REPLACE(STR(?composer__id), "^.*\\\\/(.+)", "$1")) AS ?composer__dataProviderUrl)
    }
    UNION
    {
      ?id scop:composition/scop:libretist ?libretist__id .
      ?libretist__id skos:prefLabel ?libretist__prefLabel .
      BIND(CONCAT("/people/page/", REPLACE(STR(?libretist__id), "^.*\\\\/(.+)", "$1")) AS ?libretist__dataProviderUrl)
    }
    UNION
```

Queries basic label information and forms the link for the instance page

Queries the performance's original composition and its label in Finnish

Queries the composer of the original composition and that person's label

Queries the libretist of the original composition and that person's label

Perspective configuration: Properties

```
properties": [
    {
        "id": "uri",
        "valueType": "object",
        "makeLink": true,
        "externalLink": true,
        "sortValues": true,
        "numberedList": false,
        "onlyOnInstancePage": true
    },
    {
        "id": "prefLabel",
        "valueType": "object",
        "makeLink": true,
        "externalLink": false,
        "sortValues": true,
        "numberedList": false,
        "minWidth": 200
    },
    {
        "id": "composition",
        "valueType": "object",
        "makeLink": true,
        "externalLink": false,
```

Included properties (columns in the table) and their configuration are configured in the “properties” list

ID should match the name used in the SPARQL queries

Object type properties should at least have an ID (e.g., composition__id) and a label (composition__prefLabel) from the queries

Name	Composition	Composer	Libretist	Producer
Die Schweizer-Familie (1833-12-20)	Die Schweizer Familie	Weigl Joseph	Castelli Ignaz Franz	Deutschen Schauspieler-Gesellschaft
Thaïs (1910-04-13)	Thaïs	Massenet Jules	Gallet Louis	Aino Ackté & Edvard Fazer
Thaïs (1910-04-15)	Thaïs	Massenet Jules	Gallet Louis	Aino Ackté & Edvard Fazer
Thaïs (1910-04-16)	Thaïs	Massenet Jules	Gallet Louis	Aino Ackté & Edvard Fazer

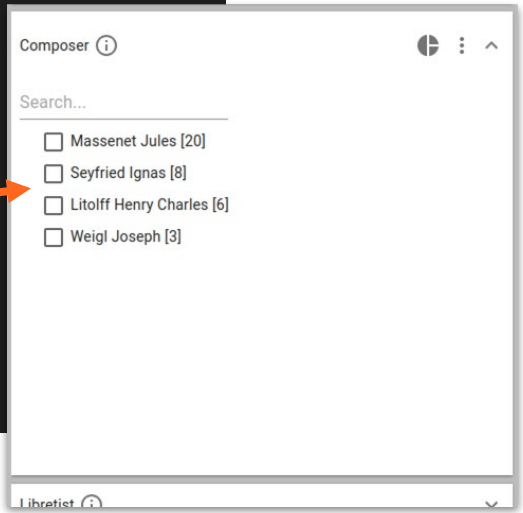
Perspective configuration: Facets

```
"facets": {
    "prefLabel": {
        "containerClass": "one",
        "facetType": "text",
        "filterType": "textFilter",
        "sortByPredicate": "skos:prefLabel",
        "textQueryProperty": "skos:prefLabel"
    },
    "composition": {
        "containerClass": "ten",
        "facetType": "list",
        "facetLabelFilter": "FILTER(LANG(?prefLabel_) = 'fi')",
        "filterType": "uriFilter",
        "predicate": "scop:composition",
        "searchField": true,
        "sortButton": true,
        "sortBy": "instanceCount",
        "sortByPredicate": "scop:composition/skos:prefLabel",
        "sortDirection": "desc",
        "pieChartButton": true
    },
    "composer": {
        "containerClass": "ten",
        "facetType": "list",
        "filterType": "uriFilter",
        "predicate": "scop:composition/scop:composedBy",
        "searchField": true,
        "sortButton": true,
        "sortBy": "instanceCount",
        "sortByPredicate": "scop:composition/scop:composedBy/skos:prefLabel",
        "sortDirection": "desc",
        "pieChartButton": true
    },
```

Type for facet (e.g., checkbox, integer range, …)

Configuration for getting the labels for values

Predicate for getting label values from instances



Perspective configuration: Visualizations

```
"resultClasses": {
    "performancesByProperty": {
        "resultClasses": {
            "performancesByPerformancePlace": {
                ]
            }
        }
    },
    "performanceLineChart": {
        "tabID": 2,
        "component": "ApexCharts",
        "tabPath": "performances_per_year",
        "tabIcon": "ShowChart",
        "sparqlQuery": "performancesByYearQuery",
        "facetClass": "performances",
        "filterTarget": "performance",
        "resultMapper": "mapLineChart",
        "resultMapperConfig": {
            "fillEmptyValues": true
        },
        "createChartData": "createSingleLineChartData",
        "title": "Vuosittaiset esitykset",
        "xaxisTitle": "Vuosi",
        "xaxisType": "category",
        "xaxisTickAmount": 30,
        "yaxisTitle": "Lukumäärä",
        "seriesTitle": "Lukumäärä",
        "stroke": {
            "width": 2
        }
    }
```

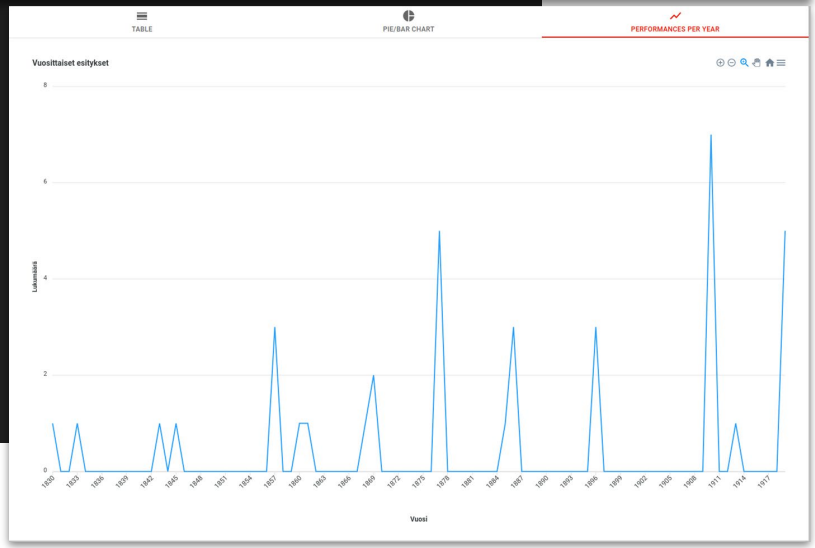
Component used for visualizing

Mapper for mapping the data from SPARQL bindings

Function taking the mapped data and outputting the data with the necessary configuration and format for the visualization component (e.g., adding chart options)

```
SELECT ?category (COUNT (DISTINCT ?performance) as
?count) WHERE {
    <FILTER>
    ?performance a scop:Performance ;
                        scop:performanceDate ?date .
    BIND(YEAR(xsd:date(?date)) as ?category)
}
GROUP BY ?category
ORDER BY ?category
```

SPARQL query for the visualization data



Other tools and frameworks for creating web applications for LOD

- Linked Data Reactor (LD-R)
- Metaphactory
- (VRTI-KG Explorer)
- …

Conclusions

- **After learning the basic process, setting up a new portal using Sampo-UI is easy and can be done in a few hours**
- **Sampo-UI includes many ready-to-use components that only require editing JSON and writing SPARQL to configure**
  - With basic use there is no need to touch the underlying JavaScript code or have deep understand of how it works
  - Examples of different kinds of facets and visualizations come with the framework and can be used as a base
    - *For more advanced use, Sampo-UI can be extended with new components, mappers, chart data creation functions to allow for more customization*

Links

- **Tool page: https://seco.cs.aalto.fi/tools/sampo-ui/**
- **GitHub: https://github.com/SemanticComputing/sampo-ui**
- **Demo: https://sampo-ui.demo.seco.cs.aalto.fi/**
- **Tutorial:**
  - Video: https://vimeo.com/817028609
  - Written tutorial: https://seco.cs.aalto.fi/tools/sampo-ui/Sampo-UI-tutorial.pdf