Data models

Zacharias Zacharodimos
CERN

Outline

- Data model
- JSONSchema
- Mapping
- Loaders
- Serializers
- Entrypoints

Data model

What is a data model?

Data model

- Data format

Data model

- Data format
- External to internal representation

Data model

- Data format
- External to internal representation
- Internal to external representation

Data model

- Data format
- External to internal representation
- Internal to external representation
- Data Access management

JSONSchema

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

https://json-schema.org

JSONSchema

- Describe your JSON data format

https://json-schema.org

JSONSchema

- Describe your JSON data format
- Provide a human and machine readable documentation

https://json-schema.org

JSONSchema

- Describe your JSON data format
- Provide a human and machine readable documentation
- Ensure validation of your data

https://json-schema.org

Mapping

“Mapping is the process of defining how a document and its fields are stored and indexed”

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping

- Describe your data format
  - Describe which fields are numbers, dates or geolocations
  - Format of date fields (e.g `yyyy-MM-dd`)

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping

- Describe your data format
  - Describe which fields are numbers, dates or geolocations
  - Format of date fields (e.g `yyyy-MM-dd`)
- Describe which fields should be indexed so they can be searchable

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping

- Describe your data format
  - Describe which fields are numbers, dates or geolocations
  - Format of date fields (e.g `yyyy-MM-dd`)
- Describe which fields should be indexed so they can be searchable
- Describe how dynamically added fields should be handled

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping

- Describe your data format
  - Describe which fields are numbers, dates or geolocations
  - Format of date fields (e.g `yyyy-MM-dd`)
- Describe which fields should be indexed so they can be searchable
- Describe how dynamically added fields should be handled
- Describe how fields should be analyzed when are indexed

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Loaders

“You can think of loaders as the definition of your input formats for records”

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-loaders

Loaders

- Loaders are responsible to transform your request payload into internal JSON format

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-loaders

Loaders

- Loaders are responsible to transform your request payload into internal JSON format
- Request data validation

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-loaders

Serializers

“You can think of serializers as the definition of your output formats for records”

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-serializers

Serializers

- Serializers are responsible to transform your internal JSON format to an external representation
    - Different output formats(e.g JSON, XML, MARCXML, etc.)

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-serializers

Serializers

- Serializers are responsible to transform your internal JSON format to an external representation
    - Different output formats(e.g JSON, XML, MARCXML, etc.)
- Control which fields should be returned when retrieving a record

https://invenio.readthedocs.io/en/latest/tutorials/understanding-data-models.html#define-serializers

Entrypoints

“What is this `setup.py`?”

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

- Invenio is developed as a Flask application

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

- Invenio is developed as a Flask application
- All its different modules have been developed as Flask extensions that wrap the final application and add up functionality.

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

But how are these extensions are being discovered and registered?

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

But how are these extensions are being discovered and registered?

- Using python `setuptools` package
  - ***Setuptools*** *is a collection of enhancements in Python that help developers to build and distribute python packages, especially if your package has dependencies on other*

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

But how are these extensions are being discovered and registered?

- Using python `setuptools` package
  - ***Setuptools*** *is a collection of enhancements in Python that help developers to build and distribute python packages, especially if your package has dependencies on other*
- Using the `entry_points` keyword argument in your `setup.py` you can specify a dictionary mapping of `entry_point_group` name to strings or list of strings
  - E.g `setup( # ... entry_points={'group.subgroup': 'entry_point_name = some_module:SomeClass'})`

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

But how are these extensions are being discovered and registered?

- Using python `setuptools` package
  - ***Setuptools*** *is a collection of enhancements in Python that help developers to build and distribute python packages, especially if your package has dependencies on other*
- Using the `entry_points` keyword argument in your `setup.py` you can specify a dictionary mapping of `entry_point_group` name to strings or list of strings
  - E.g `setup( # ... entry_points={'group.subgroup': 'entry_point_name = some_module:SomeClass'})`
- The `entry_points` are used to dynamically discover plugins and services provided by a project

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

Example

- Blogging tool that allows translation plugins
- `Blogtool.parsers` entry point group name
- Distributions register their parsers under this entrypoint group
- Blogging tool discovers on runtime all the available parsers

https://setuptools.readthedocs.io/en/latest/setuptools.html

Entrypoints

In the same fashion every Invenio module provide its entrypoint groups so other modules or instances can use them to register their services.

For example:

Your data model module

```
'invenio_jsonschemas.schemas': [
    'records = my_site.records.jsonschemas',
]
```

Invenio module, i.e. invenio-jsonschemas/ext.py

```
...
entry_point_group = 'invenio_jsonschemas.schemas'
# Load the json-schemas from extension points.
if entry_point_group:
   for base_entry in pkg_resources.iter_entry_points(
            entry_point_group):
       directory = os.path.dirname(base_entry.load().__file__)
       state.register_schemas_dir(directory)
...
```

https://setuptools.readthedocs.io/en/latest/setuptools.html


Lets craft some **data models**

Questions?