FAIR in action!

Step 0 Introduction of the use case and the steps to be done
Expected duration: 10 min

Part 1 - Fill the FDP

Expected time: 45 min
Work type: individual

In this first part you will fill the hierarchy of the FDP with data representing your Registry. You will run the FDP Reference Implementation (RI) in your machine and fill the Resources in the registry.

1. Get the `docker-compose.yml` and the `application.yml` from the GitHub repository
2. Edit the `application.yml` file, replacing `<IP_ADDRESS>` with your IP address in the local network.
3. Run the command

   ```
   docker-compose up -d
   ```

   NB: the FDP runs on port 8050: be sure to have this port free. If not, edit the port in the fdp-client service in the docker-compose.yml and in application.yml the clientUrl address with the chosen port

4. Once the FDP is up, you can reach it via browser at `http://<YOUR_IP_ADDRESS>:8050`. Here you are presented with the FDP prefilled metadata: this must be edited.

   Login using albert.einstein@example.com / password default account

   NB: if at first you get an error message "Unable to get data", just wait and retry, the blazegraph db that stores the data takes more time to be up and running

5. The hierarchy of our FDP consists of four levels: the FDP itself, Catalogs, Datasets and Distributions. Fill every level of the hierarchy the FDP metadata adding one Catalog, one Dataset and one Distribution. As a general indication for filling the metadata, keep in mind that the FDP targets both humans and machines, so try to be exhaustive describing the Resources: think that a user, reading your description, should understand what the Resource is about. After filling the fields as indicated, click on the Publish button on the upper-right side of the created Resource
   a. FDP metadata: describes the FDP. It's the first point of contact for users to understand what they are
      i. Give a title to the FDP: include your name in the title you use
      ii. Give a description of the FDP

iii. Add the publisher metadata: it should be the name of an organization or a person name responsible of publishing the FAIR Data Point
iv. Add the version metadata: use the semver rules
v. Add the language: it has to be an IRI (i.e., a term from a controlled vocabulary). One of the most used formats is ISO639-1. It is possible to use the URL defined in https://id.loc.gov/.
    1. connect to the site
    2. in the home page go to the “Languages” section and select the standard ISO639-1.
    3. Select a language from the “Top Scheme Members”
    4. From the page get the URI(s)
vi. Add license from Creative Commons licenses (look at Appendix A)
vii. Set the start and end dates: for example, use the days of the Interhealth school
viii. Add the language of the UI same as for the point v.

b. Add one Catalog that describes the Registry of your group. Fill the metadata as done for the FDP. The Catalog
c. Add one Dataset that describes the tumor cases in your registry. In addition to the common metadata, for the Dataset you will need to add
    i. The theme of the dataset: in our case, it is the code of the disease of the cases in the dataset. There can be multiple themes. Search in bioportal the ICD codes of the diseases of your registry and insert the IRI of the found classes (i.e., the ID field). Try also to add the correspondent value in other terminologies, such as SNOMED or MEDDRA (hint: you can use the FHIR terminology service of the previous days or check if the record in bioportal indicates mappings
    ii. contact point: add a fake email address in the form of mailto IRI (e.g., mailto:blue@interhealth.org).
    iii. keyword: add some keywords that describe the dataset
d. Create a Distribution: the distribution describes how our datasest can be accessed and the format of the Dataset. There can be different way of accessing the Dataset (via an API, or downloading the dataset). In our case we have an endpoint to describe how to access to the openEHR REST API to perform AQL queries. Not all the metadata can be filled in this case. For example, since the distribution cannot be downloaded, the Download URL shoul not be set. The same for the byteSize: we don’t know the size of the dataset. You will instead enter:
    i. Access URL, which is the enpoint of the openEHR REST API service of your group.
    ii. Conforms To: The distribution exposes data with the openEHR REST API: the conformsTo property can be used to tell the user that the endpoint is compliant with the openEHR REST API by adding the URL of openEHR REST API
    iii. Media Type: since the API exposes data using JSON, set the MIME Type for JSON

Step 2 - Machine readability and FDP navigation

Expected time: 15 min
Work type: guided tutorial

In this step 2, you will see how the Resources created in the previous part is represented in the RDF machine readable format using the FDP Metadata Schema based on DCAT
One of the main point that can be stressed is how the FDP Resources provide information to allow navigation. Indeed, the FDP specifications provides only two mandatory levels to be filled: FDP metadata and Catalog. Different FDPs can have different resources described. To allow a client application to navigate a structure not known a priori, the Resources describe their children and how they are related using the LDP (Linked Data Platform) ontology.

Step 3 - Extend the Schemas to add new properties

Expected time: 1h
Work type: individual

All the Resources in the FDP can be extended with new attributes that can enhance the description of the entity they represent, and consequently increase their "findability".
In this part, you will extend the Metadata Schemas of the FDPs to add more properties (metadata) to the Resources. The RI allows extending the schemas in the `Metadata Schemas` section, by editing the `Form Definition`. This section uses a specific RDF ontology called SHACL, that allows to validate an RDF according to some constraints.

3.1 First guided example: add dcat:theme property to the Catalog

During the editing of the metadata, for the Datasets it was possible to specify a theme, and we used it to specify the IRI of the ICD-10 code of the disease. For example, since the Catalog contain Datasets for specific tumor (colon, prostate and breast), we want to specify for the Catalog a more generic ICD-10 code that comprises the specific themes of the datasets. For this purpose, we'd like also the Catalogs to have a theme property.

1. Go to `Metadata Schemas → Catalog`
2. Add a property definition for `dcat:theme` specifying:
    a. `sh:path` the IRI of the property (i.e., dcat:theme) (NB: dcat prefix is already specified in the prefix section)
    b. `sh:nodeKind`: this is the type of the value of the property. In our case we want the theme to be a term from a terminology, so we set `sh:IRI`
    c. we want the property to be mandatory so we specify `sh:minCount 1`
    d. we need to tell the RI to make the property to be editable and to be shown in the UI. For that we need to specify the properties `dash:editor` and `dash:viewer`, with values `dash:URIEditor` and `dash:LabelViewer`
3. Save and Release the schemas adding a description and a version.
4. Go to the Catalog record in the FDP click Edit and add a proper theme for the Catalog.

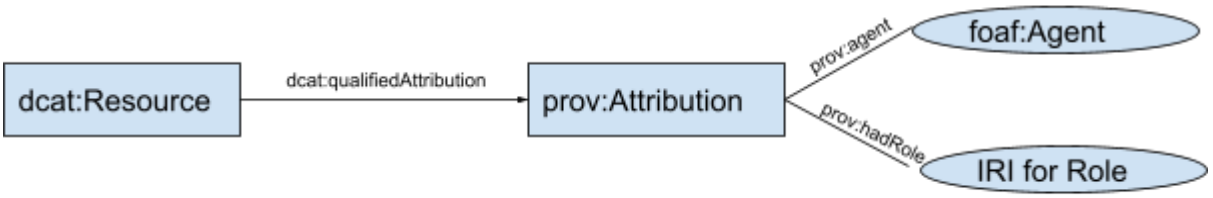
Other examples

Go on with metadata extension with the following properties:

- Country for Catalog: we want our Catalogs to have a country property
    - In DCAT, search a property that can be used to specify the country.
    - Decide whether the country is mandatory or not
    - Decide the type of the new property
    - When it's time to fill the value, use a term from http://id.loc.gov/ similarly to what we did for the language
- Type for Catalog:
    - In DCAT search for a property that can specify the type of Catalog and create the attribute in the form Catalog.
    - Search for appropriate values that can describe your Catalog in OLS (hint: it is a Registry and a very specific one)
- Type for Dataset:
    - Similarly to Catalog, add the type property and use the appropriate value
- Try to find in DCAT other useful properties that can be added

3.2 Missing direct property example

The DCAT vocabulary defines properties to specify that an Agent (Person or Organization) has a role in the Resource. The roles defined are very common: some examples are `dct:creator` and `dct:publisher`. There are some cases where it is necessary to indicate a use-case specific role. DCAT provides a generic property that allows to define a role: `prov:qualifiedAttribution`. As indicated in https://www.w3.org/TR/prov-o/#qualifiedAttribution the Range of this attribute is a node of class `prov:Attribution`. This class has two main properties: `dcat:agent` to indicate the Agent, and `prov:hadRole` to denote the role of the Agent. The figure shows the graph.



In this example, you will extend the Dataset's SHACL Form in such a way that it is possible to specify an arbitrary role in the Dataset.
The steps to do that are:

- Create a SHACL node definition of class `prov:Attribution` using `sh:NodeShape`
- Add to this class definitions for property `prov:agent` and `prov:hadRole`
- Add the `dcat:qualifiedAttribution` to the Dataset, specifying that the `sh:nodeKind` is the new defined node

Add some roles identified by an IRI to the Dataset (e.g., the data curator). To do that, search for roles in OLS.

Appendix A

Table of Creative Commons licenses

License	IRI	Description
Attribution CC BY	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by_4.0_.rdf	It lets others distribute, remix, adapt, and build upon your work, even commercially, as long as they credit you for the original creation
Attribution-ShareAlike CC BY-SA	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by-sa_4.0_.rdf	It lets others remix, adapt, and build upon your work even for commercial purposes, as long as they credit you and license their new creations under the identical terms
Attribution-NoDerivs CC BY-ND	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by-nd_4.0_.rdf	This license lets others reuse the work for any purpose, including commercially; however, it cannot be shared with others in adapted form, and credit must be provided to you.
Attribution-NonCommercial CC BY-NC	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by-nc_4.0_.rdf	This license lets others remix, adapt, and build upon your work non-commercially, and although their new works must also acknowledge you and be non-commercial, they don't have to license their derivative works on the same terms.
Attribution-NonCommercial-ShareAlike CC BY-NC-SA	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by-nc-sa_4.0_.rdf	This license lets others remix, adapt, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.
Attribution-NonCommercial-NoDerivs CC BY-NC-ND	https://raw.githubusercontent.com/creativecommons/cc.licenserdf/main/cc/licenserdf/licenses/creativecommons.org_licenses_by-nc-nd_4.0_.r	This license is the most restrictive of our six main licenses, only allowing others to download your works and share them with

	df	others as long as they credit you, but they can’t change them in any way or use them commercially.