Exercise 4: Operating on Linked Data

The git repository can be found here: https://github.com/Obiratus/exercise-4-SD

Info: The markup file in the repo is more readable.

Task 1

Task 1.1

WebID: https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me

Task 1.2

All queries can be found here: task1_2_queries

Change to the query directory:

```
$ cd task1_1_queries
```

Just for Simon: Test to check running queries from a file

```
$ comunica-sparql https://fragments.dbpedia.org/2016-04/en -f 0_test.sparql
```

1. Query WebIds With file 1_allPeopleWebIds.sparql:

```
$ comunica-sparql https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me -f 1_allPeopleWebI
```

Directly:

```
$ comunica-sparql https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me "SELECT ?knownPers
```

Description Query retrieves the WebIds of individuals mentioned in Simon's profile as "known".

```
SELECT ?knownPersonId
WHERE {
    <https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me> <http://xmlns.com/foaf/0.1/kno
}
```

- Subject: Simon's public WebID
- Predicate: Relationshyp type –> foaf:know
- Object: Variable, filled with WebIdDs of peoples Simon knows. ###### HTTP requests A single HTTP GET request is executed to fetch Simon's profile document from: https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card

2. Query Names With file 2_allPeopleNames.sparql:

```
$ comunica-sparql-link-traversal -f 2_allPeopleNames.sparql -l debug
```

Directly:

```
$ comunica-sparql-link-traversal "PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT DISTINCT ?name WHERE { <
```

Run it without traversal With file, for development and debug:

```
$ comunica-sparql https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me -f 2_allPeopleName
```

Directly:

```
$ comunica-sparql https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me "PREFIX foaf: <htt
```

Description This query retrieves the names (`foaf:name`) of all individuals, starting with Simon's direct `foaf:knows` relationships and recursively traversing all `foaf:knows` connections through the `foaf:knows*` property path (friends of friends, and so on). The `DISTINCT` keyword ensures unique names in the output.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name WHERE {
    <https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me> foaf:knows* ?person.
    ?person foaf:name ?name.
}
```

First tripple: - Subject: Simon's public WebID - Predicate: Relationshyp type –> foaf:know. The * operator indicates recursive traversal—retrieving people Simon knows directly (one foaf:knows link) and indirectly (multiple foaf:knows links, like friends-of-friends). - Object: Variable, filled with WebIdDs of peoples Simon knows directly or indirectly (friend-of-friends).

Second tripple: - Subject: This is a variable representing one of the individuals retrieved in the first triple - Predicate: This is the property used to describe a name. It specifies that the person `(?person)` has a `foaf:name` property - Object: Variable, filled with names of the persons

What happens with Traversal (using `comunica-sparql-link-traversal`):

1. The query initially requests Simon's profile (request 1)
2. From the `foaf:knows` predicate in Simon's profile, it identifies WebIds (`knownPersonId`) of connected individuals and requests each remote profile document (request 2-4):
3. For "friends of friends" (or higher-level connections), further HTTP requests are made recursively to fetch additional profiles. All profiles are queried for the `foaf:name` predicate (request 5+). ###### HTTP requests - with traversal
4. Simon's Profile:
   https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card
5. Jeremy's Profile:
   https://wiser-solid-xi.interactions.ics.unisg.ch/jeremy/profile/card
6. Christoph Buehler's Profile:
   https://wiser-solid-xi.interactions.ics.unisg.ch/cbue/profile/card
7. Sandro Pod's Profile:
   https://wiser-solid-xi.interactions.ics.unisg.ch/sandro-pod/profile/card
8. Jeremy's First Alter Ego:
   https://wiser-solid-xi.interactions.ics.unisg.ch/jeremy2/profile/card
9. Dominik Steinmann's Profile:
   https://wiser-solid-xi.interactions.ics.unisg.ch/DSteinmann/profile/card
10. Jeremy's Second Alter Ego:
    https://wiser-solid-xi.interactions.ics.unisg.ch/jeremy3/profile/card
11. Esra Pod's Profile:
    https://wiser-solid-xi.interactions.ics.unisg.ch/EsraPod/profile/card

What happens without Traversal (using `comunica-sparql`):

- The query only fetches Simon's profile document, without following `foaf:knows` links to other profiles.
- As a result, the names of recursively connected individuals are not retrieved.

HTTP requests - without traversal

This requests are made in the given order: 1. Simon's Profile with Fragment Identifier:
https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card#me

2. Simon's Profile (Base Document):
   https://wiser-solid-xi.interactions.ics.unisg.ch/simon/profile/card —

Task 2

See Pod.java