Molid Documentation

Angelo Veltens

May 15, 2020

Contents

A mock server, that can be used for testing Solid apps locally.

Use cases:

- Provide data on localhost during development
- Ensure your Solid app is really compatible to different kind of Pod structures
- Provision Test-Pods for automated integration testing
- ...

CHAPTER 1

Quick start

Start using `npx`, no installation needed

```
$ npx molid
```

```
Molid - mock Solid server listening on port 3333!
```

Molid provides some Solid data, that you can now access via HTTP, e.g.

```
$ curl http://localhost:3333/profile/card
```

All the mock data is stored within the `.molid` folder in your current directory. Adjust it to your needs and restart.

CHAPTER 2

Read The Docs

Read the whole documentation on https://molid.readthedocs.io

CHAPTER 3

Contents

3.1 Installation

3.1.1 Install as a global CLI command

```
$ npm install --global molid
```

then start Molid anywhere via

```
$ molid
```

Be aware that the data directory `.molid` will be created, where you execute the command.

3.1.2 Install for usage within your Solid app project

```
$ npm install --save-dev molid
```

Add a script to your package.json:

```
{
  "scripts": {
    "molid": "molid"
  }
}
```

Then start Molid via

```
$ npm run molid
```

3.2 The data directory

Molid uses the file system to initialize the data it provides at runtime. Unlike real Solid server implementations like Node Solid Server it will never *change* any data in that directory.

3.2.1 Default data dir

When you start Molid via CLI or an npm script, it will use the default data dir called *.molid*, relative to the place where you execute the command. The default data dir is also used when *calling start()* without a `dataDir`.

3.2.2 Initialization

When Molid starts it will create the data dir, if it does not yet exist. In that case, it will initialize the directory with a minimal Solid profile. If the data dir does already exist, Molid will not touch anything.

After initialization, Molid will continue to import the data from that directory into it's internal store.

3.2.3 Data Import

After initialization, Molid imports all data from the data dir to an internal store. The relative path of the files results in a matching url path, but the file endings starting with `$` will be stripped.

Some examples:

File	URL path
.molid/public/bookmarks	/public/bookmarks
.molid/notes.ttl	/notes.ttl
.molid/profile/card$.ttl	/profile/card

LDP Containers

Folders that contain at least one imported file, will lead to the creation of an LDP container. This includes parent folders. For example, if there is a file *.molid/frist/second/file.ttl*, the following containers will be created:

Folder	Container URL path
.molid/first/second/	/first/second/
.molid/first/	/first/
.molid/	/

Note: All container URLs end with a /. Requests without trailing slash will be answered with 404 Not Found.

Edit data

You can edit, add or remove files inside the data directory as you wish. All files have to contain valid RDF in turtle format. A folder has to contain at least one imported file (directly or in a sub-folder), otherwise it will be ignored.

Note: You have to restart Molid after you changed data.

3.3 Starting programmatically

You can start Molid programmtically from inside your application or test:

```
import { start } from 'molid';

async function your_code() {
    const molid = await start();
}
```

The code above will start Molid using a random available port and import data from *the default data directory*.

3.3.1 Start parameters

You can pass a configuration object to the `start` function:

```
import { start } from 'molid';

async function your_code() {
    const molid = await start({
        port: 3456,
        dataDir: path.join(__dirname, '.fake-data'),
    });
}
```

Parameter	Description
port	The port to start Molid on
dataDir	Absolute path to the data directory

3.3.2 Construct an absolute URI

When starting Molid, you probably want to fetch something from it, and therefore need the URI of a document. When Molid starts on a random port, you will not know the URI. But even if you defined the port yourself, the following function is still handy to construct absolute URIs referring to Molid resources:

```
import { start } from 'molid';

async function your_code() {
    const molid = await start();
    const webId = molid.uri('/person/card#me');
    const response = await fetch(webId);
    // ...
}
```

3.3.3 Stopping Molid

To stop a running Molid instance, call the `stop` method on the object returned from `start`

```
import { start } from 'molid';

async function your_code() {
    const molid = await start();
    // do your things ...
    molid.stop();
}
```

Note: You can start multiple Molid instances at once on different ports.

3.4 Usage in integration tests

3.4.1 General approach

By *starting Molid programmatically* it can easily be used inside integration tests. Just start Molid in a setup method and stop it during tear down.

Here is an example using Jest, but the approach can be applied to any test framework:

```
1  import { start } from 'molid';
2  import { loadProfile } from './your-code';
3
4  describe('using Molid for testing', () => {
5      let molid;
6      beforeAll(async () => {
7          molid = await start({
8              dataDir: path.join(__dirname, '.mock-profile-john'),
9          });
10     });
11
12     afterAll(async () => {
13         await molid.stop();
14     });
15
16     it('successfully fetches the profile document', async () => {
17         const webId = molid.uri('/profile/card#me');
18         const profile = await loadProfile(webId);
19         expect(profile.name).toEqual('John');
20     });
21 });
```

In lines 7-9 Molid is started before all tests, using a data directory next to the test. We placed some fake data in that directory and the expectations in our test are based on the data. In line 17 we construct a WebID using `molid.uri()` and pass it to a fictional `loadProfile` method. Line 13 finally stops the running Molid instance after all tests.

Warning: Be careful to always `await` both, the startup and stopping of Molid!

3.4.2 Jest support

Molid has built-in support to simplify Jest tests.

givenMolid(name, fn)

You can use `givenMolid(name, fn)` to create a Jest `describe` block, that will handle startup and shutdown of a Molid instance for you. The following test is an equivalent to the one described in the section *General approach*, but using `givenMolid`:

```
import { givenMolid } from 'molid/lib/molid-jest';
import { loadProfile } from './your-code';

describe('using the givenMolid function', () => {
    givenMolid('with a profile of a person called John', molid => {
        it('successfully fetches the profile document', async () => {
            const webId = molid.uri('/profile/card#me');
            const profile = await loadProfile(webId);
            expect(profile.name).toEqual('John');
        });
    });
});
```

Molid will start up before *all* of the tests of that group and shutdown afterwards. The running instance is passed to `fn`, so that you can easily generate URIs via `molid.uri()` inside your tests.

Molid will use a data directory next to the test file, that equals the `name` plus the extension `.molid`. So in the example above the data directory will be `with a profile of a person called John.molid`.

Note: Be aware that the directory name may be sanitized when you are using some fancy characters in the `name`.

3.5 Configuration

3.5.1 Adjust port

The port Molid is running on can be adjusted by setting the `PORT` environment variable to the desired value.

```
PORT=3030 npm run molid
```

3.6 Limitations

Molid is still in early development. Some features are missing intentionally, since Molid is a mock server and no full implemention of the Solid specification. Some other features might be useful to mock, but are missing yet.

Things Molid does not provide:

- Authentication / Authorization
- Link headers
- Metadata of container contents (sizes, modification dates, . . .)
- Updating data (POST, PUT, PATCH)
- Support for any file formats other than turtle
- for sure some more things not listed here

If one of the limitations are holding you back from using Molid, please file a feature request or contribute by submitting a merge request

3.7 Changelog

3.7.1 Version 0.3.0

New features

- non-empty folders get imported as LDP containers
- non-existing resources now return 404 instead of an empty document

3.7.2 Version 0.2.0

New features

- start and stop Molid programmatically
- use Molid for integration testing
- Jest support via givenMolid()

3.7.3 Version 0.1.2

New features

- Start via npx or via npm script
- Initialize with data from local .ttl files
- Serve data as text/turtle via http