Repository API

Note: you can try using a live Hydrilla instance at https://hydrilla.koszko.org/api_v2*; try pasting this into the URL bar:* https://hydrilla.koszko.org/api_v2/query?url=https://opencores.org/projects

Fetching resource info

"resource" is now the equivalent of what we used to call "bag" in Haketilo 0.1.

Let's assume we want to retrieve the definition of resource helloapple. We can issue a GET to https://hydrilla.example.com/resource/helloapple.json. It returns:


```
{
    "$schema": "https://hydrilla.koszko.org/schemas/api_resource_description-1.schema.json",
    "source_name": "hello",
    "source_copyright": [
        {
            "file": "report.spdx",
            "sha256": "249599a36ad9f6f33b8bf16ca6ace2413d95d8cdb481173968fa4e97fd0a5635"
        },
        {
            "file": "LICENSES/CC0-1.0.txt",
            "sha256": "a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499"
        }
    ],
    "type": "resource",
    "identifier": "helloapple",
    "long_name": "Hello Apple",
    "uuid": "a6754dcb-58d8-4b7a-a245-24fd7ad4cd68",
    "version": [
        2021,
        11,
        10
    ],
    "revision": 1,
    "description": "greets an apple",
    "dependencies": [
        {
            "identifier": "hello-message"
        }
    ],
    "scripts": [
        {
            "file": "hello.js",
            "sha256": "18dd7d7ac9f74a5ba871791ac6aa4d55530a336ae1b373538b6dd5320b330414"
        },
        {
            "file": "bye.js",
            "sha256": "cee8d88cf5e5346522fd9ee5e1f994b335fb68d2f460b27b2a2a05f0bcd2b55b"
        }
    ]
}
```


or 404 Not Found in case resource helloapple does not exists. If multiple versions are available, newest version's JSON description gets returned. A different one can be retrieved by including version in the GET request. Requested path should then take the form /resource/<resource_name>/<version> (e.g. GET https://hydrilla.example.com/resource/helloapple/2021.11.9).

File can be retrieved by issuing a GET request to https://hydrilla.example.com/file/sha256/<file's_sha256_sum>. For example, requests to https://hydrilla.example.com/file/sha256/a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499 and

hydrilla.example.com/file/sha256/18dd7d7ac9f74a5ba871791ac6aa4d55530a336ae1b373538b6dd5320b330414 yield contents of LICENSES/CC0-1.0.txt and hello.js, respectively.)

A JSON schema describing resource definition format is available here.

Fetching mapping info

"mapping" is now a set of what we used to call "pattern" in Haketilo 0.1.

Let's assume we want to retrieve the definition of mapping helloapple (mapping is allowed to have the same identifier as a resource). We can issue a GET to https://hydrilla.example.com/mapping/helloapple.json. It returns:


```
{
    "$schema": "https://hydrilla.koszko.org/schemas/api_mapping_description-2.schema.json",
    "source_name": "hello",
    "source_copyright": [
        {
            "file": "report.spdx",
            "sha256": "249599a36ad9f6f33b8bf16ca6ace2413d95d8cdb481173968fa4e97fd0a5635"
        },
        {
            "file": "LICENSES/CC0-1.0.txt",
            "sha256": "a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499"
        }
    ],
    "type": "mapping",
    "identifier": "helloapple",
    "long_name": "Hello Apple",
    "uuid": "54d23bba-472e-42f5-9194-eaa24c0e3ee7",
    "version": [
        2021,
        11,
        10
    ],
    "description": "causes apple to get greeted on Hydrillabugs issue tracker",
    "payloads": {
        "https://hydrillabugs.koszko.org/***": {
            "identifier": "helloapple"
        },
        "https://hachettebugs.koszko.org/***": {
            "identifier": "helloapple"
        }
    }
}
```


or 404 Not Found in case helloapple mapping does not exist. If multiple versions are available, newest version's JSON description gets returned. A different one can be retrieved by including version in the GET request. Requested path should then take the form /mapping/<mapping_name>/<version> (e.g. GET https://hydrilla.example.com/mapping/helloapple/2021.11.8).

File can be retrieved by issuing a GET request to https://hydrilla.example.com/file/sha256/<file's_sha256_sum>. For example, request to https://hydrilla.example.com/file/sha256/a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499 yields contents of LICENSES/CC0-1.0.txt.

A JSON schema describing mapping definition format is available here.

Querying mappings that match given URL

Assume we want to query mappings with patterns matching https://example.org/a/b. We can issue a GET to https://hydrilla.example.com/query?url=https://example.org/a/b. It returns:


```
{
    "$schema": "https://hydrilla.koszko.org/schemas/api_query_result-2.schema.json",
    "mappings": [
```

```
        {
            "identifier": "example-org-minimal",
            "long_name": "Example.org Minimal",
            "version": [
                2022,
                5,
                10
            ]
        },
        {
            "identifier": "example-org-experimental",
            "long_name": "example.org fix bundle",
            "version": [
                0,
                4
            ]
        }
    ]
}
```


The value of "mappings" can be an empty array ([]) in case no mappings match given URL. In case many versions of some mapping match the URL, only the most current of those versions that do is listed. In case the URL is of wrong format, 400 Bad Request is returned.

A JSON schema describing query result format is available here.

Fetching source info

By "source" we mean description of a source package used to build Hydrilla packages.

Assume we want to get description of source package hello. We can issue a GET to https://hydrilla.example.com/source/hello.json. It returns:


```
{
    "$schema": "https://hydrilla.koszko.org/schemas/api_source_description-2.schema.json",
    "source_name": "hello",
    "source_copyright": [
        {
            "file": "report.spdx",
            "sha256": "249599a36ad9f6f33b8bf16ca6ace2413d95d8cdb481173968fa4e97fd0a5635"
        },
        {
            "file": "LICENSES/CC0-1.0.txt",
            "sha256": "a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499"
        }
    ],
    "source_archives": {
        "zip": {
            "sha256": "f61bfc5594a2603452a8c8e5f7a3f36f255bc39d5ffea02c9fdefc0de6461c74"
        }
    },
    "upstream_url": "https://git.koszko.org/hydrilla-source-package-example",
    "definitions": [
        {
            "type": "resource",
            "identifier": "helloapple",
            "long_name": "Hello Apple",
            "version": [
                2021,
                11,
                10
            ]
        },
```

```
        {
            "type": "resource",
            "identifier": "hello-message",
            "long_name": "Hello Message",
            "version": [
                2021,
                11,
                10
            ]
        },
        {
            "type": "mapping",
            "identifier": "helloapple",
            "long_name": "Hello Apple",
            "version": [
                2021,
                11,
                10
            ]
        }
    ]
}
```

Or 404 Not Found in case no such source package exists. Here zip was listed among available source archive extensions. It means we can download the zip archive of this source package by issuing GET to https://hydrilla.example.com/source/hello.zip.

With schema version beginning with major number 1, zip archive is always guaranteed to be available.

A JSON schema describing source package description format is available here.