API

The nirvana has an RESTful-ish JSON-based API. This means you can access information about uploaded packages from your client application or even upload new packages.

Requests

All API urls are "subdirectories" of `http://NIRVANA/api/`, so if you're using the official nirvana instance, it is `http://nirvana.ooc-lang.org/api/`. All API urls given in this documentation will be relative to this "root" url.

If you want to issue a request for information, just perform a HTTP GET request on this url as described in this documentation. For some urls, you'll be able to add an optional `type` parameter that specifies the desired result format; this will be explicitly denoted in the url description. If you've passed an invalid type, an error response (see below) will be given. The error text will be "Unknown type: *type*".

All requests return a JSON object. The exact format depends on the url and the specified format.

In any case, this JSON object will contain a special value `"__result"` whose value is either `"ok"` (if the request was successful and the desired data is returned) or `"error"` (if the request failed, e.g. due to invalid input values). If `"__result"` is `"error"`, the response will contain one additional field `"__text"` that contains a error description as a string named. Example:

```
{
    "__result": "error",
    "__text": "ZOMBIES!!!"
}
```

Oh, and never ever forget the trailing slash.

URLs

/categories/

Format:

```
/categories/
```

Get a list of all registered package categories.

You can pass the `type` parameter here whose value can be `contents` (default) or `details`.

If `type` is `contents`, return a JSON object mapping category slugs to category names. If `type` is `details`, return a JSON object with one value `categories` that is mapped to a JSON array containing all available category slugs.

Example:

```
% curl 'http://nirvana.ooc-lang.org/api/categories/'
{"lang-libs": "Libraries useful for programming languages development",
 "nonsense": "Temporary packages for testing.", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/api/categories/?type=contents'
{"lang-libs": "Libraries useful for programming languages development",
 "nonsense": "Temporary packages for testing.", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/api/categories/?type=details'
{"__result": "ok", "categories": ["nonsense", "lang-libs"]}
```

/category/

Format:

```
/category/:slug/
```

Use this URL to demand information about one specific category given by its slug.

If `type` is `contents` (default), return a JSON object mapping package slugs to package names.

If `type` is `details`, return a JSON object containing the following values:

- `slug`
- `name`
- `package`: an array of package slugs connected to this category.

Example:

```
% curl 'http://nirvana.ooc-lang.org/api/category/nonsense/'
{"helloworld": "Hello World!", "empty": "Empty", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/api/category/nonsense/?type=details'
{"__result": "ok", "packages": ["helloworld", "empty"],
 "slug": "nonsense", "name": "Temporary packages for testing."}
```

/packages/

Packages

Format:

```
/packages/:package_slug/
```

Use this url to get information about one specific package.

If `type` is `contents` (default), return a JSON object mapping version slugs to version names.

If `type` is `details`, return a JSON object containing the following values:

- `slug`
- `name`
- `author`
- `homepage`
- `latest_version`: the slug of the version marked as "latest".
- `versions`: an array of version slugs.

Example:

```
% curl 'http://nirvana.ooc-lang.org/api/packages/helloworld/'
{"0.1": "Wurst", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/api/packages/helloworld/?type=details'
{"category": "nonsense", "name": "Hello World!", "author": "fred", "versions": ["0.1"],
 "homepage": "", "slug": "helloworld", "latest_version": "0.1", "__result": "ok"}
```

Versions

Format:

```
/packages/:package_slug/:version_slug/
```

Use this url to get information about a specific package's specific version.

If `type` is `contents` (default), return a JSON object mapping variant slugs to variant names.

If `type` is `details`, return a JSON object containing the following values:

- `slug`
- `name`
- `package`: package's slug.
- `latest`: true if this version is marked as being the latest version.
- `variants`: an array of variant slugs.

Example:

```
% curl 'http://nirvana.ooc-lang.org/api/packages/helloworld/0.1/'
{"src": "Source", "linux-i686": "", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/api/packages/helloworld/0.1/?type=details'
{"name": "Wurst", "package": "helloworld", "__result": "ok",
 "variants": ["linux-i686", "src"], "slug": "0.1", "latest": true}
```

Variants

Format:

```
/packages/:package_slug/:version_slug/:variant_slug/
```

Use this url to get information about a variant of a version of a package.

No `type` parameter here.

Return a JSON object containing the following values:

- `slug`
- `name`
- `version`: version's slug.
- `usefile`
- `checksums`
- `checksums_signature`

The three latter values contain the urls of the usefile, checksums and checksums signature files. These urls are relative to the nirvana **root** url (**not** the api url).

Examples:

```
% curl 'http://nirvana.ooc-lang.org/api/packages/helloworld/0.1/src/'
{"version": "0.1",
 "checksums_signature": "/packages/helloworld/0.1/src/helloworld.checksums.sig",
 "name": "Source", "usefile": "/packages/helloworld/0.1/src/helloworld.use",
 "checksums": "/packages/helloworld/0.1/src/helloworld.checksums",
 "slug": "src", "__result": "ok"}
% curl 'http://nirvana.ooc-lang.org/packages/helloworld/0.1/src/helloworld.use'
Name: Hello Woooorld!
Version: 0.1
Variant: src
Origin: meatshop://
Build: ooc helloworld.ooc
Binaries: helloworld
```

/submit/

Format:

```
/submit/
```

Use this request to submit a new usefile to nirvana.

This request has to be issued as a HTTP POST request and requires some form-encoded data:

- `usefile`: the contents of the usefile.
- `user`: the uploader's username.
- `token`: the uploader's api token, used for authentication.
- `slug`: the usefile's package slug.
- `checksums`: the contents of the checksums file (default: empty).
- `name`: the variant's name (default: empty).

The package and version are required to exist.

The usefile is required to contain at least the `Name`, `Version`, `Variant` and `Origin` fields.

If the variant was successfully added to the nirvana, a JSON object with a single value (apart from the obligatory `__result` value, of course) is returned:

`path`: the variant path relative to the nirvana root.

/authorized/

Format:

```
/authorized/
```

Use this request to test if a certain user's api token is correct and if this user is authorized to manage a specific variant.

This request has to be issued as a HTTP POST request and requires some form-encoded data:

- `user`
- `token`
- `package`: package's slug
- `version`: version's slug
- `variant`: variant's slug

Return a JSON object containing this value:

`authorized`: a boolean value determining if the user is authorized to manage this variant.

/search/

Format:

```
/search/?pattern=...
```

Use this request to search for packages. Pass a simple (non-regex) pattern as a parameter, nirvana will search its database and return all packages that contain parts of your pattern in their names, slugs and author's user names.

A JSON object mapping package slugs to package names is returned.

Example:

```
% curl 'http://nirvana.ooc-lang.org/api/search/?pattern=hel'
{"__result": "ok", "helloworld": "Hello World!"}
% curl 'http://nirvana.ooc-lang.org/api/search/?pattern=ndd'
{"greg": "greg recursive descent parser generator", "__result": "ok"}
```

Questions?

If you have any questions, feel free to ask them in our irc channel #ooc-lang on freenode.