Never-API

The Never-API is a custom-built RESTful API for retrieving the taxonomic data contained in the NCBI. This API was built to provide a more streamlined and efficient way to access the taxonomic data compared to the already existing NCBI Entrez API, as well as to have full control over the full stack. To keep the database of the Never-API in sync with the NCBI, it is updated and rebuilt with the latest data on a daily basis.

Since the Never-API is still in development, we implement a Set of low-level abstractions that allow for easy modification and extensions of our API-Client.

```ts
import type { GenomeLevel } from "../../types/Taxonomy";
import { Rank } from "../../types/Taxonomy";
```

Endpoints

To be able to painlessly extend our API-Client, we define an enum `Endpoint` that contains the available Endpoints of the Never-API. This way we have a single source of truth for all API endpoints, that can be easily referenced and modified.

```ts
export enum Endpoint {
  Accessions = "accessions",
  Children = "children",
  Levels = "levels",
  Lineage = "path",
  MRCA = "mrca",
  Names = "names",
  GenomeCount = "num_genomes",
  GenomeCountRecursive = "num_genomes_rec",
  Parent = "parent",
  Ranks = "ranks",
  Subtree = "subtree",
  Taxon = "taxi",
  TaxonID = "taxids",
  TaxonInfo = "taxa_info"
};
```

Entry

To be able to restrict the returned data from the Never-API to our specific set of expected values we import our custom enums `GenomeLevel` and `Rank`.

```ts
import type { GenomeLevel } from "../../types/Taxonomy";
import { Rank } from "../../types/Taxonomy";
```

We also define an interface for the genome counts in the way the Never-API returns them.

```ts
export interface NeverGenomeCount {
  level: GenomeLevel;
  count: number;
}
```

Using these types, we define a single interface for the entries returned by the Never-API. This allows us to have a consistent structure for the data we receive from the API, independent of the specific endpoint being called. In the end, this also simplifies the conversion of this `raw` data into our domain models by enabling us to use a single function/method for all data received by the API.

```ts
export interface Entry {
  name?: string;
  common_name?: string;
  taxid?: number;
  is_leaf?: boolean;
  parent?: number;
  rank?: Rank;
  accession?: string;
  level?: GenomeLevel;
  raw_genome_counts?: NeverGenomeCount[];
  rec_genome_counts?: NeverGenomeCount[];
};
```

Since the Never-API usually returns a set of results, we define a custom type called `Response` as an array of our `Entry` objects.

```ts
export type Response = Entry[]
```

Request

To be able to effortlessly and safely send requests to the Never-API, we create a general and extensible `Request` class that handles the construction and sending of API requests.To achieve this, we first define an enum `ParameterKey` that contains the available query parameters for the Never-API. This abstracts away possible changes to the API in the future and adds a safeguard against invalid parameters.

```ts
export enum ParameterKey {
  Term = "t",
  Exact = "e",
  Page = "p",
  PageSize = "n"
}
```

We implement the `Request` class using the Builder pattern.

Since the location of the Never-API does not change, we can hardcode the base URL, reducing the need for another parameter. The `Request` class contains methods for setting the endpoint and adding query parameters (Builder pattern).

```ts
export class Request {
  baseURL: string = "https://neighbors.evolbio.mpg.de/";
  endpoint?: Endpoint;
  parameters: URLSearchParams = new URLSearchParams();

  constructor(endpoint?: Endpoint) {
```

```
    this.endpoint = endpoint;
  }

  addParameter(key: ParameterKey, value: string | number | boolean): this {
    this.parameters.append(key, String(value));
    return this;
  }

  addParameters(params: URLSearchParams): this {
    for (const [key, value] of params) {
      this.parameters.append(key, value);
    }
    return this;
  }

  getParameters(): URLSearchParams {
    return this.parameters;
  }

  setEndpoint(endpoint: Endpoint): this {
    this.endpoint = endpoint;
    return this;
  }

  /**
   * The `Send` method constructs the full request URL using the base URL,
   endpoint, and query parameters and fetches the response from the Never-API.
   * We check if the response is ok (status code 2XX) before returning the JSON
   data.
   * @returns
```

The Send method constructs the full request URL using the base URL, endpoint, and query parameters and fetches the response from the Never-API. We check if the response is ok (status code 2XX) before returning the JSON data.

```ts
  async Send(): Promise<Response> {
    if (!this.endpoint) {
      throw new Error("Endpoint is not set!");
    }
    const requestUrl = new URL(this.endpoint + "/", this.baseURL);
    requestUrl.search = this.parameters.toString();

    const response = await fetch(requestUrl, { method: "GET" });

    if (!response.ok) {
      throw new Error("Never-API response was not ok: " + response.statusText);
    }
```

```
13      const json = await response.json();
14
15      // Workaround for the MRCA endpoint that does not return an array, but just
        an object that looks like this: { "taxid": 9606 } -> The endpoint will be
        changed later to return an array.
16      if (this.endpoint === Endpoint.MRCA && json.taxid || this.endpoint ===
        Endpoint.Parent && json.taxid) {
17        return [json];
18      }
19
20      if (!json) {
21        throw new Error(`Empty response to ${requestUrl} from Never-API`);
22      }
23
24      return json;
25    }
26  }
```

This general architecture allows for easy extension and modification of the API client in the future. By using a consistent structure for requests and responses, we can quickly adapt to changes in the Never-API without having to rewrite large portions of the code.

We also get the added benefit of keeping our code “DRY”, by reusing the `Request` class for all API calls.