Tide tool

unknown

Oct 19, 2021

CONTENTS

This tool interacts with the UK Environment Agency's tide data in `.csv` format to serve tide gauge readings at a series of tide gauges around the UK.

CHAPTER

ONE

REST API

1.1 Valid endpoints

The rest API should support four paths, `/station/json`, `/data/graph`, `/data/json` and `/data/html`, returning information on the station (in `json` format), the tide gauge data (as a graph, in `json` format and in html table format respectively) in response to an HTTP `GET` request. Each path should support a number of query parameter arguments to be given below

The `/data/json` path should also accept `POST` requests to upload more data to the application, with the data provided in `json` format.

See below for fuller details and some example calls.

1.2 The `/station/json` path

This should accept `GET` requests with required query parameters `stationReference` or `stationName` (but not both at the same time). When called on a valid station it should return a `json` string containing station information corresponding to:

```
{
    "stationName": "<the station name, as per .csv file>",
    "stationReference": "<the station reference, as per csv file>",
    "northing": "<OS northing value>",
    "easting": "<OS easting value>",
    "latitude": "<GPS latitude value>",
    "longitude": "<GPS longitude value>"
}
```

So that (e.g) a call to `/station/json?stationReference=E74339` returns

```
{
    "stationName": "Stornoway",
    "stationReference": "E74339",
    "northing": 932739,
    "easting": 142280,
    "latitude": 58.20781,
    "longitude": -6.38897
}
```

as does a `GET` request to `/station/json?stationName=Stornoway`. You can assume that the device on which your app is running is connected to the internet, if it needs to make external API calls.

1.3 The `/data/graph` path

This should accept `GET` requests with required query parameters `stationReference` or `stationName` (but not both at the same time), as well as optional `from` and `to` parameters. When called on a valid station, it should return either a response containing a `.png` file containing a graph of the tide data for the relevant platform, or an HTML page linking to such an image. The graph should be bounded by the `from` and `to` parameters, or contain all the station data if `from` and `to` are missing.

1.4 The `/data/json` path

When a `GET` request is made this endpoint should accept a query parameter `stationReference` or `stationName` (but not both at the same time) as well as optional parameters `from`, `to` and `statistic`.

When called as (e.g) `/data/json?stationName=Stornoway` the (ordered) tide value data for the Stornoway station should be return as a `json` string in the format

```
{
    "stationName": "Stornoway",
    "from": "2021-09-20T00:00:00Z",
    "to": "2021-09-26T06:00:00Z",
    "tideValues": {
                    "2021-09-20T00:00:00Z": -1.757,
                      ...
                    "2021-09-26T06:00:00Z": 0.087
                  }
}
```

with similar behaviour for the `stationReference parameter`

When the `from` and/or `to` parameters are present, then these limit the range of data which should be presented, e.g `/data/json?stationName=Stornoway?from=2021-09-23T01:30:00Z&to=2021-09-23T02:00:00Z` should return

```
{
    "stationName": "Stornoway",
    "from": "2021-09-23T01:30:00Z",
    "to": "2021-09-26T02:00:00Z"
    "tideValues": {
                    "2021-09-23T01:30:00Z": -1.611,
                    "2021-09-23T01:45:00Z": -1.722,
                    "2021-09-26T02:00:00Z": -1.78
                  }
}
```

The optional `statistic` parameter can be equal any one of `min`, `max` or `mean`. When present then the result of the relevant operation should be returned in the json (i.e. the station minimum, maxiumum or mean value). This parameter can appear with the `from`, `to`, or both, which limits the time period over which the statistic should be taken.

As an example the a `GET` request to `/data/json?stationName=Stornoway&statistic=mean` should return

```
{
    "stationName": "Stornoway",
    "stationReference": "E74339",
    "from": "2021-09-20T00:00:00Z",
```

(continues on next page)

(continued from previous page)

```
    "to": "2021-09-26T06:00:00Z",
    "mean": -0.6529161676646708
}
```

The path `/station/json` should also accept `POST` requests, with the optional query parameter `write`. The input data will be in the request body in `JSON` format, with the following example showing the schema:

```
[{
    "stationName": "Stornoway",
    "stationReference": "E74339",
    "dateTime": "2021-10-18T00:00:00Z",
    "tideValue": 1.234
 },{
    "stationName": "Newlyn",
    "dateTime": "2021-10-17T03:00:00Z",
    "tideValue": -0.032
 }
]
```

If the `write` query parameter is present and equal to `true`, then the updated data should be written to the `tideReadings.csv` file. If not present, or present and equal to `false`, then data should not be written to disk.

1.5 The `/data/html` path

When a `GET` request is made this endpoint should accept one query parameter from `stationReference`, `stationName` and `statistics` as well as optional parameters `from` and `to`. If present, the `statistics` parameter must be a comma separated list consisting of some or all of `min`, `max` and/or `mean`.

Examples of valid requests are:

`/data/html?stationName=Stornaway` `/data/html?stationReference=E72339&from=2021-09-26T05:45:0`
`/data/html?statistic=min,mean`

The response should be an HTML table containing the relevant data, limited appropriately by the `from` and `to` parameters. If either `stationName` or `stationReference` is present, then this should be an ordered table of tide values indexed by date & time. If `statistic` is present, then this should be a table of maximum, minimum and/or mean values, indexed by station name. Note that this is different behaviour than for the `/data/json` endpoint above.

1.6 The `process` module

Module containing a class to process tidal data.

class process.**Reader**(*filename*)

Class to process tidal data.

data [pandas.DataFrame] The underlying tide data.

Read in the rainfall data from a named `.csv` file using `pandas`.

The DataFrame data is stored in a class instance variable `data` indexed by entry.

Parameters **`filename`** (*str*) – The file to be read

Examples

```
>>> Reader("tidalReadings.csv").data.loc[0].stationName
'Bangor'
```

add_data(*date_time*, *station_name*, *tide_value*)

Add data to the reader DataFrame.

Parameters

- **date_time** (*str*) – Time of reading in ISO 8601 format
- **station_name** (*str*) – Station Name
- **time_value** (*float*) – Observed tide in m

Examples

```
>>> reader = Reader("tideReadings.csv")
>>> original_len = len(reader.data.index)
>>> reader.add_data("2021-09-20T02:00:00Z",
                    "Newlyn", 1.465)
>>> len(reader.data.index) = original_len + 1
True
```

max_tides(*time_from=None*, *time_to=None*)

Return the high tide data as an ordered pandas Series, indexed by station name data.

Parameters

- **time_from** (*str or None*) – Time from which to report (ISO 8601 format). If `None`, then earliest value used.
- **time_to** (*str or None*) – Time up to which to report (ISO 8601 format) If `None`, then latest value used.

Returns The relevant tide data indexed by stationName.

Return type pandas.Series

Examples

```
>>> reader = Reader("tideReadings.csv")
>>> tides = reader.max_tides()
>>> tides["Newlyn"]
2.376
```

mean_tides(*time_from=None*, *time_to=None*)

Return the mean tide data as an ordered pandas Series, indexed by station name data.

Parameters

- **time_from** (*str or None*) – Time from which to report (ISO 8601 format)
- **time_to** (*str or None*) – Time up to which to report (ISO 8601 format)

Returns The relevant tide data indexed by stationName.

Return type pandas.Series

Examples

```
>>> reader = Reader("tideReadings.csv")
>>> tides = reader.mean_tides()
>>> tides["Newlyn"]
0.19242285714285723
```

min_tides(*time_from=None*, *time_to=None*)

Return the low tide data as an ordered pandas Series, indexed by station name data.

Parameters

- **time_from** (*str or None*) – Time from which to report (ISO 8601 format) If `None`, then earliest value used.
- **time_to** (*str or None*) – Time up to which to report (ISO 8601 format) If `None`, then latest value used.

Returns The relevant tide data indexed by stationName.

Return type pandas.Series

Examples

```
>>> reader = Reader("tideReadings.csv")
>>> tides = reader.min_tides()
>>> tides["Newlyn"]
-2.231
```

station_graph(*station_name*, *time_from=None*, *time_to=None*)

Return a matplotlib graph of the tide data at a named station, indexed by the dateTime data.

Parameters

- **station_name** (*str*) – Station Name
- **time_from** (*str or None*) – Time from which to report (ISO 8601 format)
- **time_to** (*str or None*) – Time up to which to report (ISO 8601 format)

Returns Labelled graph of station tide data.

Return type matplotlib.figure.Figure

station_tides(*station_name*, *time_from=None*, *time_to=None*)

Return the tide data at a named station as an ordered pandas Series, indexed by the dateTime data.

Parameters

- **station_name** (*str or list of strs*) – Station Name(s) to return
- **time_from** (*str or None*) – Time from which to report (ISO 8601 format)
- **time_to** (*str or None*) – Time up to which to report (ISO 8601 format)

Returns The relevant tide data indexed by dateTime and the stationName(s)

Return type pandas.DataFrame

Examples

```
>>> reader = Reader("tideReadings.csv")
>>> tides = reader.station_tides(["Newlyn", "Bangor"])
>>> tides.loc["2021-09-20T02:00:00Z", "Newlyn"]
0.937
```

write_data (*filename*)
Write data to disk in .csv format.

Parameters **filename** (*str*) – filename to write to.

PYTHON MODULE INDEX