Une image contenant texte, signe

Description générée automatiquement

**How works WRAPI Example Middleware**

**Author**

Quentin WENZINGER – Writer

**DOCUMENT TRACKING**

|  |  |  |  |
| --- | --- | --- | --- |
| Mises à jour | | | |
| *Version* | *Date* | *Authors* | *Purpose of the update* |
| 1.0 | 04/01/2022 | Quentin WENZINGER | Initial version |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Mailing list | | | |
| Radioplayer France |  | Radioplayer WorldWide |  |
| Yann Legarson | *I* | Michael Hill | *I* |
| Victor Perrot | *I* | Leo Andrews | *I* |
|  |  | Raunak Lodha | *I* |
|  |  | Laurence Harrison | *I* |
|  |  | Lawrence Galkoff | *I* |

***A*** *= Application,* ***O*** *= Observations,* ***I*** *= Information,* ***V****= Validation*

**SUMMARY**

[1. Project Depenencies 4](#_Toc92113384)

[2. Error handling 4](#_Toc92113385)

[3. Log management 4](#_Toc92113386)

[4. Cache management 4](#_Toc92113387)

[5. Managers 5](#_Toc92113388)

[6. Routes 5](#_Toc92113389)

[7. Réponses 5](#_Toc92113390)

[8. Stations data management 5](#_Toc92113391)

[a. OnAir 5](#_Toc92113392)

[b. OnDemand 5](#_Toc92113393)

[c. Schedule 6](#_Toc92113394)

[9. OnDemand data management 6](#_Toc92113395)

[10. Categories data management 6](#_Toc92113396)

[11. Recommendations data management 6](#_Toc92113397)

# Project Depenencies

The project has the following dependencies:

* body-parser: to manage POST requests contents
* cors: to manage CORS query
* dotenv: to manage variables environment parameters
* express: to manage REST requests
* fs: to manage local files
* https: to manage HTTPS communication
* radioplayer-wrapi-sdk: to manage WRAPI communication

# Error handling

There are several levels of error handling within the application:

* Missing file detection (.env, https key or certification, WRAPI key)
* Missing configuration detection (in .env)
* Detection of malformed query (missing or bad shaped parameter)
* Loading in progress detection
* Internal error detection
* WRAPI error detection

# Log management

Logs are generated with it creation date

Pre-launch errors send a message explaining the problem in the console.

Operational errors send a message in the console and in an external "middleware\_error.txt" file to keep track of the event.

# Cache management

A cache management configuration purpose to the application user is in the .env file.

These recommendations follow the "How to" "Caching" section of the "Developers Radioplayer” website, frequency updates:

* 24 hours for Stations informations
* 12 hours for Program Schedule
* 12 hours for Ondemand
* 90 secondes for On Air

And we added :

* 24 hours for Categories

In case of incorrect input, missing data or CUSTOM\_CACHING=false in the .env file then the system will apply by default the cacheExpireAt result of the request concerned.

# Managers

A DataManager singleton will manage data through multiple other managers. One by route (Categories, Ondemand, Recommendation, Stations).

All managers are in ‘./managers/’ folder.

# Routes

Middleware is cut by the roads by which it can be solicited.

These are:

* Categories (/categories)
* Ondemand (/ondemand)
* Recommendations (/recommendations)
* Stations (/stations)

You can find them in ‘./routes/’folder.

Except recommendations endpoint, other endpoint are GET request.

# Request paramters

Every endpoint can support request parameters you can found on <https://developers.radioplayer.org/api-reference> .

# Réponses

Middleware responses are formed with the same structure as WRAPI.

These are composed of a "data" part where the data is stored and a "meta" part where descriptions of the query and the data are stored.

The Middleware's responses for non-cached data are those of the WRAPI itself.

# Stations data management

When application starts, WRAPI stations are loaded and stored in the cache.

This data is updated periodically through its associated cache management environment variable (STATIONS\_CACHING or cacheExpireAt of the query).

Once done, the application gives access to its endpoints starting by /stations.

Stations are sorted by their rpuid to optimize search.

When a stations/onair, stations/schedule or stations/ondemand query is performed for the first time the result is kept until its caching time is reached.

## OnAir

Data sorted by category (current or next).

Each category has its own caching time.

## OnDemand

Data sorted by category (pure or seriesId).

This category represent endpoint it come from, /stations/ondemand/ or /stations/ondemand/:seriesid.

Each category is store in a two-dimensional tab that represent page and size requested if data paginated.

Each combination has its own caching time.

## Schedule

Data stored in a two-dimensional tab that represent page and size requested if data paginated.

Each element has its own caching time.

If query has ‘from’ or ‘to’ filled then the request is directly sent to WRAPI.

# OnDemand data management

/ondemand/ et /ondemand/ :odIds data are paginated but we can’t ask for a specific page or size.

We can’t build a cache system and requests are directly sent to WRAPI.

# Categories data management

Data sorted by country (ISO 3166 numeric country code) then by type (‘live’, ‘ondemand’).

Each type has its own caching time.

# Recommendations data management

There are too many parameters to build a consistent cache system, requests are directly sent to WRAPI.

Warning : this endpoint is a POST.