

NaturalIntelligence / StubbyDB

Unwatch 1 Unstar 2 Fork 1

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings


Branch: master StubbyDB / README.md Find file Copy path

 **amitguptagwl** Changing donate button img url 5a8c185 3 hours ago

1 contributor

224 lines (156 sloc) 7.43 KB Raw Blame History


StubbyDB



npm install stubby-db
6 dependencies version 3.5.0
0 dependents updated 19 hours ago
312 downloads in the last month
download rank: top 20% of 288,500 packages



stubby-db downloads (3 months) 131



A nodejs based complete solution for maintaining the stubs for your project

To install

```
$npm install stubby-db -g
```

For basic help

```
$stubbydb --help
```

Important links : [Wiki](#), [NPM](#), [Demo](#) application, [issues](#)

Donate

History

Developed by a victim.

I was previously using an open source project to manage stubbed data. However with the time when the project is expanded, I started missing few features in that. I also faced many limitations. Hence I decided to created a simple, light weighted, highly customizable, and easy to handy tool to manage stub data in whichever way I want.

Description

Stubby DB is a npm module. In simple words you can stub the HTTP/HTTPS calls to provide stubbed data. You can make stubbed data service for SOAP or REST calls. Basic functionality is somewhat similar to stubby4j. But it has so many other features.

Features and Basic terminology

Folder structure

```
StubbyDB-Project
|__ dbsets
|__ dumps
|__ mappings
|__ stubs
```

Mappings

yaml based mappings are required to map a request with relevant response file. You can have multiple files instead of maintaining a big fat file. You can also mark files which should be used to map the requests. So the mappings which are in progress can be excluded.

```
- request:
  method: POST
  latency: 0
  headers:
  status: 200
  url: /stubs/simple
  post: Hello ([a-zA-Z]+) ([a-zA-Z]+)!! Your number is ((\[0-9]{2}) ([0-9]+))

  response:
    file: post.xml
```

Short Notations To further reduce the size of yaml mappings, there is a support for short notations;

```
- request: /stubs/healthcheck
  response: OK
```

Default configuration To further reduce the size of yaml file, You can use default configuration to be applied with each mapping.

```
- request:
  url: /stubs/simple
  post: Hello ([a-zA-Z]+) ([a-zA-Z]+)!! Your number is ((\[0-9]{2}) ([0-9]+))

  response:
    file: post.xml
```

Strategy

Many times, you don't have just one-to-one mapping between the request and response. You want to server different response every time, or you want to serve default response instead of saying 404 (stub data not found). There are many strategies, you can use suits to your requirement;

```
- request:
  url: /stubs/employee/([0-9]+)

  response:
    strategy: random
    files: ["<% url.1 %>.xml", "file2.xml", "file3.xml"]
```

Stubby DB is currently supporting following strategies;

1. first-found : In above example, whichever file is found first, serve that response.
2. random : In above example, serve any file from mentioned list.
3. round-robin : In above example, serve the files in sequential order. So on second same request, it'll serve response from file2.xml

Coming soon

- random + first-found
- round-robin + no-repeat
- random + no-repeat
- many others

Latency

When you want to serve the response with some delay. It may be useful to test negative scenarios or for performance test.

```
- request:
  url: /stubs/simple
  latency: 500 #In milliseconds

response:
  file: post.xml
```

Dynamic Response

Instead of creating multiple mappings for same request where just few query parameters or headers are being changed. You can use Regular expression to write generic mappings.

Capture some part from request and use it determine the file name, or use it in file contents, or in response headers.

```
- request:
  method: POST
  url: /stubs/post2
  post: Mobile=([0-9]+)
  #name=Amit&Mobile=781011111&Sal=100000.00&DOJ=25-APR-2012

response:
  #Mobile=781011111,781011111,<% post.2 %>
  body: <% post.0 %>,<% post.1 %>,<% post.2 %>
```

Data dumps

Stubby DB let you construct the response from multiple files. You can include contents of other file into your response files.

This is the content from sample response file.

```
Employee: <% post.1 %>
Projects:
  [[dumps/projects:project1,project2]]
```

Now you need not maintain a big response file. Split them and reuse in different responses. It'll also increase consistency. You can also decide at the run time which dumps have to be included.

DB sets

If you still feel that you have so many response files in your project and it is being difficult to maintain them, use this **skeleton based approach**.

Use a response file as skeleton and fill the data from a dbset.

Sample response file

```
Employee: <% url.1 %>
Name: ##Name##
Projects:
  [[dumps/projects:##Projects##]]
```

Actual Response

```
Employee: 001
Name: Some Name
Projects:

Title : Title 1
Description: This is project 1
:

Title : Title 2
Description: This is project 2
:
```

Sample dbset file: ./dbsets/employee

Num	Name	Projects
001	Some Name	project1, project2
002	Another Name	project2, project3, project4

Sample Mapping;

```
- request:
  url: /stubs/employee/([0-9]+)

dbset:
  db: employee
  key: <% url.1 %>
  #strategy: random #round-robin
  err:
    file: fault.xml

response:
  file: stubs/employeedetail
```

See sample [yaml](#) for more detail. DBset in itself many features.

Keep checking wiki for latest updates in APIs and more implementation detail.

DEBUG & Logging

Debugging with StubbyDB is very easy. StubbyDB creates debug.log, and exceptions.log for more detail about what is going on. You can see on screen logs with '-v' or '--verbose' option.

If query parameter 'debug=true' is provided with request URL. It gives additional detail with the response: Original request, Matched mapping, Raw and fine response etc. Response status depends on how your requests gets resolved.

If 'debug=true' is provided on root url, then it gives system level information, configuration etc.

Markers

StubbyDB has inbuilt support for markers. So if you write `{{TODAY+1y-2m+3d}}` in somewhere of response body or file, It'll be converted into 'yyyy-mm-dd' format date. If you say `{{URL:someurl}}`, it'll encode the URL. There are many other markers are in development. And in near future StubbyDB will be able to support markers written by you.

Configuration

StubbyDB provides you many way of configuring your project: commandline arguments, configuration file, directory structure.

SSL hanndshaking

Stubby DB supports HTTPS and 2 way SSL handshaking as well. Have a look on wiki page and demo application for more detail.

Compression

If accept-encoding header is set to deflate or gzip then stubby-db serve compressed response.

Note : Although I have many fantastic ideas for stubby-db, I am pausing it's development for some time due to other priorities. However I'll keep supporting for any bug, small features, and any request.

