Summary

OurAPIServer is a tool for create a Restful service interface for our projects.

The main objective of the project is the creation of a standard tool with supports for many existing projects, including WordPress, Drupal, etc. Also does not limit to MySQL data source, it can be widely extended easily with the use of plugins.

Features

- Creating APIs very easy using a visual interface.
- Send Parameters in the URL using two ways (the PATH and the URL's variables)
- Make Requests in any possible format (GET, POST, PUT, DELETE).
- Keep Logs of each API's requests.
- Show Charts (Stats) for the Logs.
- Utilization of plugins to extend server functionalists.
- Full Cache Support.
- Support Bandwidth Throttling.

Setup

Requirements

{Incomplete}

Installation

Just copy the project to the folder 'www' of Apache2, and go to the project's URL using the Browser.

Configuration

The configuration of the projects is located on data/config.ini.

User Interface

The user interface is basically an API editor where you can perform the following actions:

- **Edit** (Edit a previously added API)
- **Stats** (Chart(stats) for all the API's requests)
- **Logs** (Logs for all the API's requests)
- **Delete** (Delete created APIs)
- **Clone** (Make an exact copy of the selected API)
- **Export** (Exports the API)

How it works

1. The app map all paths of created APIs.
2. Upon receiving a request, if it match with the URL mapped to the path, the app execute this API.
3. It checks whether the mapping coincides with the method defined (GET, POST …) in the API.
4. Check the Security Layer.
5. Executes the Action.
6. Execute the "After" Action.
7. Show the Output.

Making an API

To create an API must provide certain information to the system, as the name, the description, the route… In short you need to implement a REST service in our web interface.

{I} Basic

In this section we define the basic parameters of the API, which described below:

Name

Friendly name, we use this parameter to identify the API.

Description

Description of the API does (be explicit), need specify the goal and some helps about the operations.

Path

Path to access to the API, not need the complete URL only the relative path, example: if our path is “/hello” the URL to access is http://OURAPISERVERURL/hello.

Also we can define parameters in the URL, example: “/hello/@user” in this case @user is a parameter that can be used in the rest of the API (actions, after actions and security).

Other way to define parameters is using the URL variables (GET, POST…), example: “/hello?user=VALUE” in this case the parameter @user is VALUE.

Cache

Time (in seconds) that the system cache the result of the API. The system will be send the same output for this time. This will make your application run like a Formula 1 race car.

Throttling

Bandwidth Throttling (in kbps). Serving pages ASAP makes your application vulnerable to Denial-of-Service (DOS) attacks. You can specifies how much time it should take to process a request.

Method

Method to be used in the request. The system supports all methods although you can specify whether to connect via GET, POST, PUT, and DELETE.

Output

Format used to print the data. By default the system use JSON.

{II} Source

Source

This is the “core” of the system, in this place we select the “source” action to be executed. When we select an action in the bottom of the select button we see a brief explanation about the source.

The most simple “source” is PRINTER and the goal of this is “print” the text entered when the API is call, so if we write “Hello” in the action and for example in the path “/hello” , when we browse to http://OURAPISERVERURL/hello we see the “Hello” text.

Other common “source” is MySQL on this case we put in the action a QUERY of MySQL and defined in the parameters of connection.

Parameters

Each source need some parameters for works, for example: MySQL need host, port, database; SQLITE need path of database; LDAP need DN and host.

After Action

We see the after action selecting the {show advanced actions} link, this part is very usefully when we need fine adjust of our “output” because this “PHP” code is executed before the “action” and after the “output” by default the value of this field is “return $result;”, we need keep in mind that “$result” is an array of all items of the “action” query. This part is recommended for advanced users.

{III} Security

For each created API we can implement a “security” layer, for restring the access to the response. The security use the parameters assigned in the URL.

Type

In this place we define the “type” of security of our API, by default the type is NONE and mean that the API no have security.

A very basic example is the type MANUAL, for this type we need define four fields, the **parameters** for User and Password and the **values.** In context, if we path is /hello/@user/@password then in User Parameter we need put “user”, in

User the right value ex: “john”, in Password Parameter “password” and in the password ex: “123”.

Encryption

The encryption operation to apply to the password value.