Gmail by Google

Ryan Hess <hess.ryanm@gmail.com>

Documentation on Login API

Alexandru Mos <alexandrumos@gmail.com> Sat, Oct 5, 2013 at 8:25 AM
To: Sam Anderson <sam@theemptyspace.com>
Cc: Nick Delfino <ndelfino@umass.edu>, Ryan Hess <hess.ryanm@gmail.com>

Hello,

I'll explain here how the API will work. I don't know yet where the API will be hosted, I'll use a dummy URL: http://test.com/api/v1

Each request to the API will end to a **section** and a **method**. This is how the URL will look like: http://test.com/api/v1/**section**/**method**

Obs. In case the server can't be configured trough Apache rewrite engine to skip index.php from the URL, this is how the URL will look like: http://test.com/api/v1/index.php/**section**/**method**

The idea is that a request to authenticate the user is needed first. This request will return an token which will be used later to authorize requests to the API. An token, after creation is valid for 10 minutes. When a request using that token is made, the TTL period will be increased with another 10 minutes. This time interval is configurable, I don't know for sure how will be into the productive version of the API.

Each request to the API will end up to an URL containing section and method. All the send info will be transmitted into POST variables (including the token). The only request which can be done without token is the one to authenticate the user.

Here's an example on how a user can be authenticated:

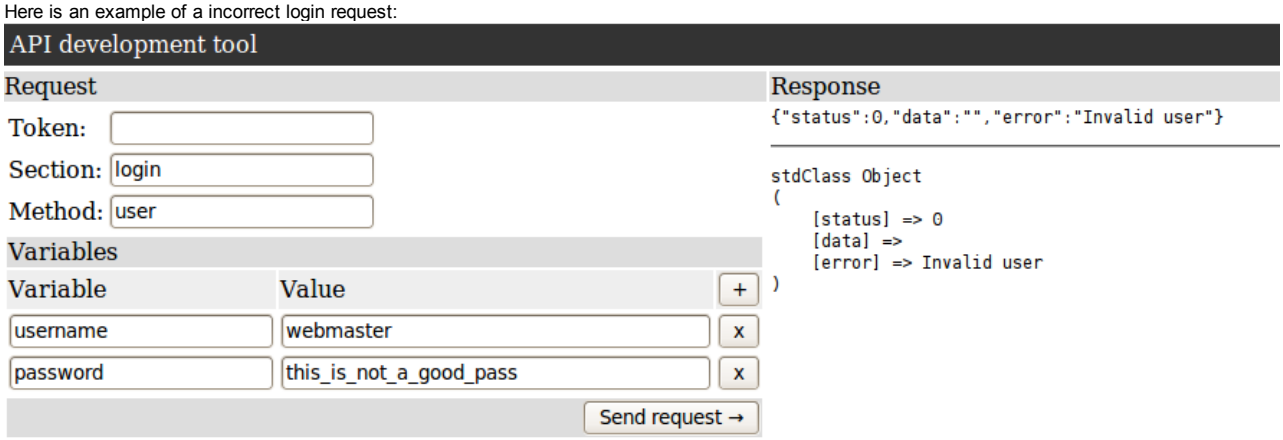
URL: http://test.com/api/v1/login/user
POST fields:
username: user's username
password: MD5 hash of the password (to avoid sending plain passwords over public networks)

Request, if successful will return a JSON encoded object. I attach a screenshot from the tool I made to test the API:

Each response is an object containing this fields:
status = 1 => This is an successful request, the additional **data** field is available and stores the data returned by the request (can be string, int, array). In this case it holds an object containing two variables: **token** and **userid**. Token will be user to authorize requests later and userid is just to know the user's ID, if needed.

status = 0 => An unsuccessful request. However, a request which is made correctly (for example to retrieve a list of *things*, and the table containing those *things* is empty) but the result is empty will a successful request. status = 0 is only for errors.

Here is an example of a incorrect login request:

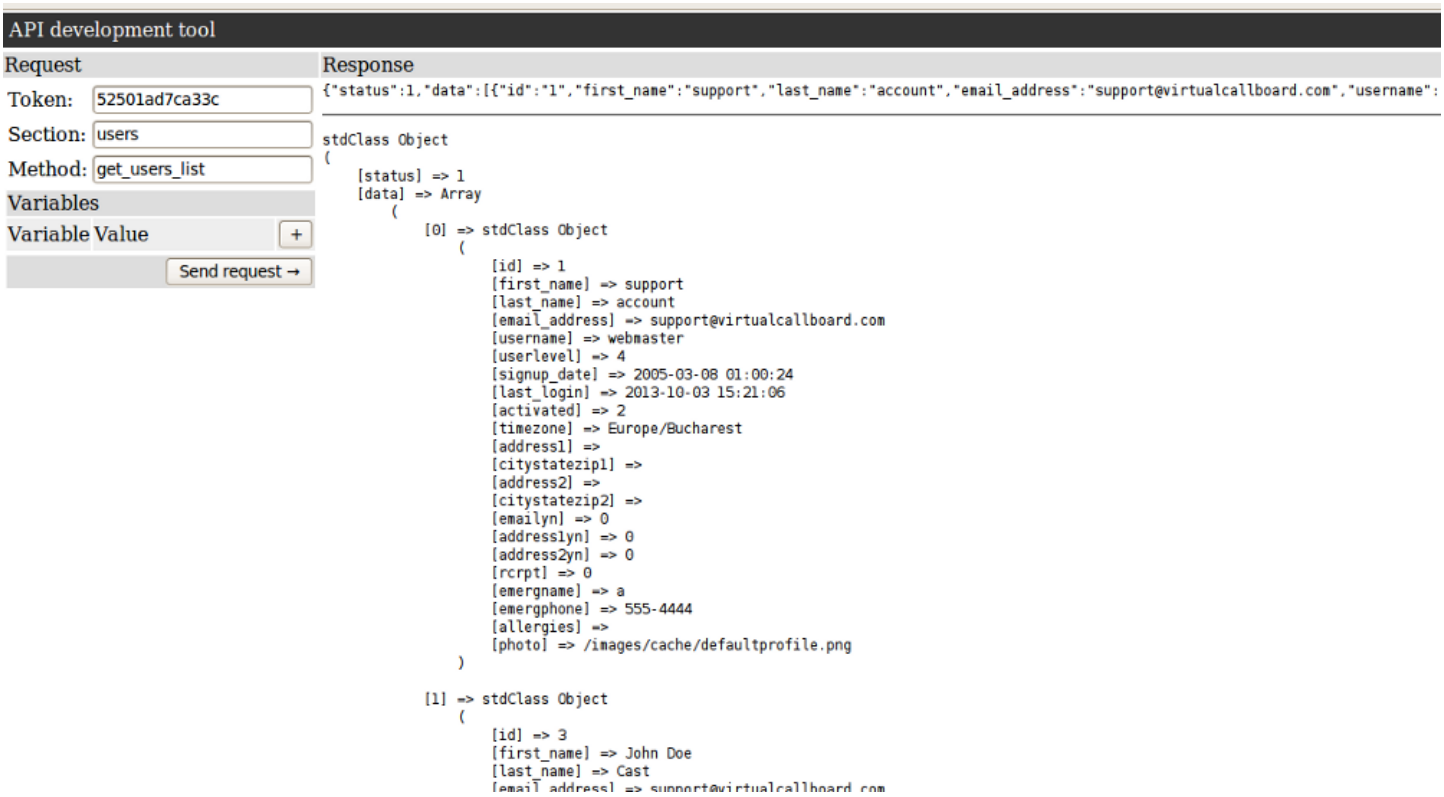


For requests which have **status** = 0, there is always an empty **data** value and an **error** value which returns a string telling which is the problem.

This is how a request will be made. In this case I make a request to obtain the user's list:

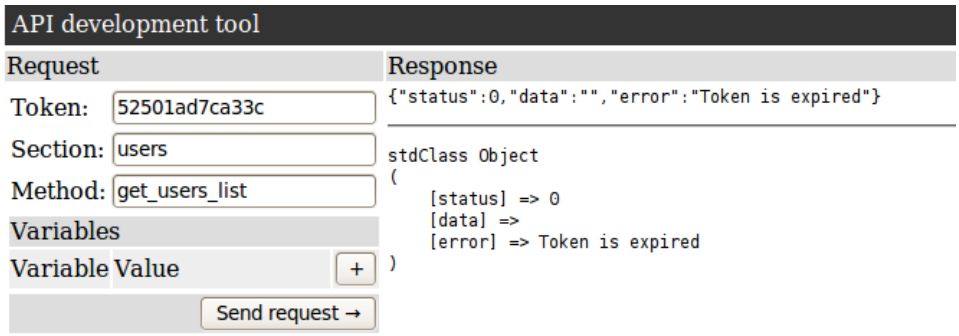
section: users
method: get_users_list

This request doesn't need any parameter except the **token** which is obtain at login. If the token is correct, a successful response will look like this:

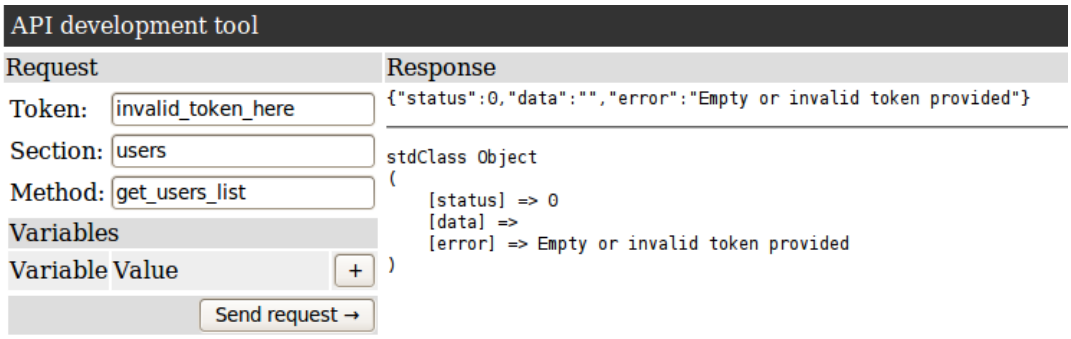


Data value now keeps an array of objects, each object being an row from the user's table.

However, if meanwhile the token expires, here's how a "token expired" response looks like:

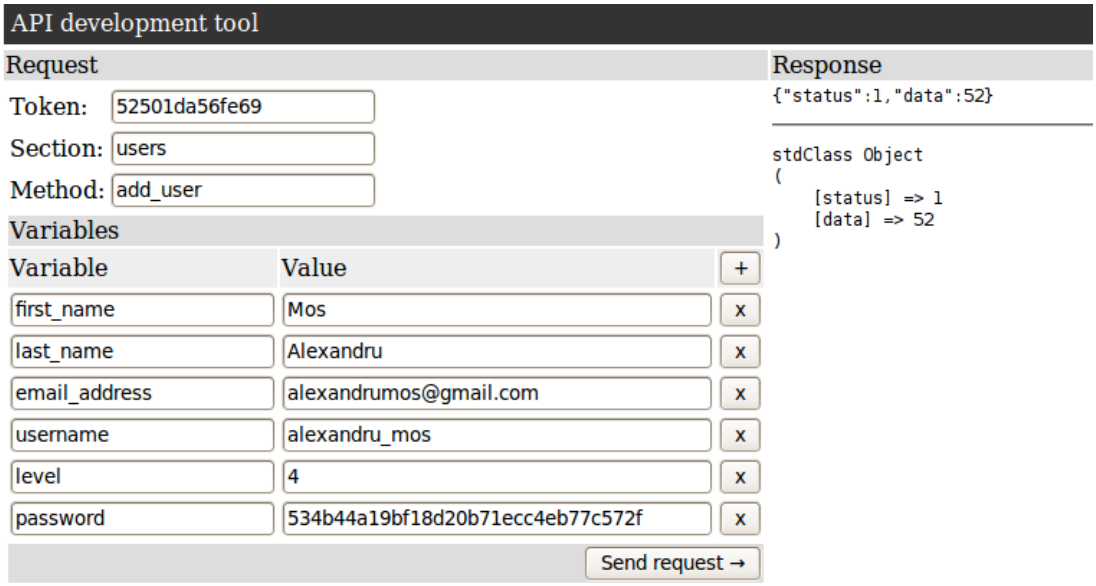


Requests with inexistent tokens will respond like this:



I'll explain how a request which contains a few fields will look like. For example, let's create a user. To create an users, the request needs to be done at
section = users
method = add_user
and contain the following POST variables:
first_name
last_name
email_address
username
level
password

This is how the response looks like:



Usually for newly created stuff the **data** value will have the new thing's ID. In this case, the newly created user have the ID = 52.

Obs. In the API development app I put the token input near the section/method inputs for convenience, but it's actually treated like any other variable send trough POST. Who knows PHP can view here how a request is built using cURL library:

```php
<?php

    if (isset($_POST['submit']))
    {
        $token = isset($_POST['token'])? trim(strtolower($_POST['token'])) : '';
        $section = trim(strtolower($_POST['section']));
        $method = trim(strtolower($_POST['method']));

        $query = array();

        if (is_array($_POST['var']) && is_array($_POST['val']))
        {
            for ($x = 0; $x <= count($_POST['var']) - 1; $x++)
            {
                $var = trim(strtolower($_POST['var'][$x]));
                $val = $_POST['val'][$x];

                if (strlen($var) > 0)
                {
                    $query[$var] = $val;
```

```
            }
        }

    }

    if (strlen($token) > 0 && !isset($query['token']))
    {
        $query['token'] = $token;
    }

    $url = 'http://localhost/api/v1/index.php/'.$section.'/'.$method;
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $query);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $rsp = curl_exec($ch); // this is the response, as string

    curl_close($ch);
}
```

Hope my quick documentation helps. After I finish to implement all the sections and methods I'l start do the documentation for the available calls. I will include the input variables and what outputs every request. The methods I already created might change but I think the login system will remain unchanged.

Tomorrow I work for a couple of hours, but starting from Monday I'll work so that I finish the coding part before Tuesday afternoon. After that the documentation which shouldn't take me longer than a few hours.

I attached the API development tool to the email (it's called index.php) and when the API will become accessible only the line 31 needs to be changed to set the $url variable to the API's good URL and it will work.

Best regards,
Alexandru Mos

[Quoted text hidden]

index.php
8K