JSON API

Die PBS MiData verfügt über ein einfaches JSON API, um Personen und Gruppen auszulesen. Das JSON Antwort Format folgt den Konventionen von http://jsonapi.org.

Für Chrome gibt es die Chrome App Postman, mit der die API gut getestet werden kann:
http://www.getpostman.com/

Die originale Dokumentation von Hitobito findest du in Englisch unter folgendem Link:
https://github.com/hitobito/hitobito/blob/master/doc/development/05_rest_api.md

Authentisierung

Die folgenden Methoden dienen zur Authentisierung und Verwaltung des authentication tokens. Als Parameter müssen immer person[email] und person[password] übergeben werden. In der Antwort ist der Wert des authentication_token enthalten, welches für die regulären Requests jeweils mitgegeben werden muss.

```
Token auslesen:             POST /users/sign_in.json
Token neu generieren:       POST /users/token.json
Token löschen:              DELETE /users/token.json
```

Endpunkte

Sobald das Authentication Token bekannt ist, können verschiedene Endpunkte abgefragt werden. Dazu bestehen zwei Möglichkeiten:

* Parameter: user_email und user_token werden als Pfadparameter angegeben, der Pfad muss mit .json enden (Bsp:/groups/1.json?user_email=zumkehr@puzzle.ch&user_token=abcdef).

* Headers: X-User-Email , X-User-Token und Accept (application/json) Header entsprechend setzen.

```
Hauptgruppe Details:        GET /groups
Gruppen Details:            GET /groups/1
Gruppen Personen:           GET /groups/1/people
Person Details:             GET /groups/1/people/1
```

Beispiel mit R

https://github.com/Michael-Schaer/rLang_Tests/blob/master/JSON_tests.R

Beispiel mit Python (2015)

Bitte die Zeilen 7 bis 9 anpassen. Die Gruppennummer findet ihr in der Datenbank am besten über die URL heraus, wenn ihr die Gruppen offen habt.

Für einen Test mit dem Script, müsst ihr noch das Python Paket "Requests" installieren.

http://de.python-requests.org/de/latest/user/quickstart.html#json-basierte-antwortdaten

```
__author__ = 'Martin Spielmann / Tux'

import requests

baseurl = "https://db.scout.ch"

usermail = "DEINEMAIL"
userpassword = "DEINPASSWORT"
groupid = 301

headers = {
    "Accept": "application/json"
}

try:
     # Authentication Token generieren und auslesen. Dazu müssen die User Credentials
     # mitgesendet werden im POST Request.
    res = requests.post(
         baseurl + "/users/sign_in.json",
         data={"person[email]": usermail,
         "person[password]": userpassword},
         headers=headers)

    authentication_token = res.json()['people'][0]['authentication_token']

    # Headers, welche mitgesendet werden um Token ergänzen
    # Der Token kann ab jetzt verwendet werden zur Authentifikation,
    # das Passwort ist nicht mehr nötig.
    # Der Token kann mit /users/sign_in.json (POST) neu generiert werden und mit
    # /users/token.json (GET) abgerufen und /users/token.json (DELETE) gelöscht werden.
    headers["X-User-Email"] = usermail
    headers["X-User-Token "] = authentication_token

    res = requests.get(baseurl + "/groups/" + str(groupid) + ".json", headers=headers)
    j = res.json()

    print("Untergruppen in der Gruppe '" + j['groups'][0]['name'] + "':")
    l =[g['name'] for g in j['linked']['groups'] if g['id'] in j['groups'][0]['links']['children']]
    print(l)
except:
    print("Fehler bei der Netzwerkkommunikation.")
    raise
```

Beispiel mit PHP (2015)

```
<?php
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <macki@dracheburg.ch> wrote this file.  As long as you retain this notice you
 * can do whatever you want with this stuff. If we meet some day, and you think
 * this stuff is worth it, you can buy me a beer in return. JoÃ«l Stampfli / Macki
 * ----------------------------------------------------------------------------



getscoutDB.php v0.1

Gets person details of a group from db.scout.ch

scoutDB->getGroups("<groupID1>,<groupID2>...<groupIDN>")

Returns array:
                $people
                                "id"
                                "nickname"
                                "email"
                                "first_name"
                                "last_name"
                                "address"
                                "zip_code"
                                "town"
                                "href"
                                "role1"
                                "role2"
                                ...
                                "phone1"
                                "phone2"
                                ...

macki@dracheburg.ch
*/


//Config

$user = "dummy@user.ch"; //User needs to be "Adressverwalter"
$password = "topSecret";
$urlBase = "https://db.scout.ch";
$groups2get = "1";        //Groupids

//hard config
$arrLabels = array("Spezialfunktion", "Fonction spÃ©ciale", "funzione speciale"); //Display labels
instead role_type for these roles



$labels = implode(",", $arrLabels);
$labels = utf8_encode($labels);
$arrLabels = explode(",", $labels);

//**************************************************************************//
class scoutDB {

        function scoutDB () {

        }

        function login ($user, $password) {

                global $urlBase;

                //Get token
                //API Url
                $url = $urlBase . "/users/sign_in.json?person[email]=". $user
."&person[password]=". $password;

                //Initiate cURL.
                $ch = curl_init();

                //The JSON data.
                $jsonData = array(
                        "person[email]" => $user,
```

```
                        "person[password]" => $password
                );

                //Encode the array into JSON.
                $jsonDataEncoded = json_encode($jsonData);

                //set cURL options
                curl_setopt($ch, CURLOPT_URL, $url);
                curl_setopt($ch, CURLOPT_POST, 1);
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
                curl_setopt($ch, CURLOPT_POSTFIELDS, $jsonDataEncoded);
                curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json'));

                //Execute the request
                $result = curl_exec($ch);

                //Close cURL
                curl_close($ch);

                //Decode JSON response
                $decoded = json_decode($result, TRUE);
                $authToken = $decoded["people"][0]["authentication_token"];

                return $authToken;
        }


        function qry($qry) {

                global $urlBase;
                global $user, $password, $authToken;

                $authToken = $this->login($user, $password);

                //API Url
                $url = $urlBase . "/groups" . $qry . ".json";

                //Initiate cURL.
                $ch = curl_init();

                //Set cURL options
                curl_setopt($ch, CURLOPT_URL, $url);
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
                curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json', 'X-
User-Email: ' . $user, 'X-User-Token: '.$authToken, 'Accept: application/json'));

                //Execute the request
                $result = curl_exec($ch);

                //Close cURL
                curl_close($ch);

                //Decode JSON response
                $decoded = json_decode($result, TRUE);

                return $decoded;
        }

        function getGroups($groupIds) {

                global $arrLabels;

                $groupId = array();
                $groupId = explode(",", $groupIds);

                //Get data
                $data = array();
                foreach ($groupId as $id) {
                        $qry = "/".$id."/people";
                        $data = $this->qry($qry);
                }

                //loop phone numbers into new array
                $numbers = array();
                foreach ($data["linked"]["phone_numbers"] as $number) {
                        $numbers[$number["id"]] = $number["number"];
                }

                //loop roles into new array
                $roles = array();
                foreach ($data["linked"]["roles"] as $role) {
```

```
                        if (in_array($role["role_type"], $arrLabels)) { //Replace role_type with
label
                                $roles[$role["id"]] = $role["label"];
                        }
                        else {
                                $roles[$role["id"]] = $role["role_type"];
                        }

                }

                //loop persons into new array
                for ( $i = 0; $i < count($data["people"]); $i++) {
                        //$id = $decoded["people"][$i]["id"];     //Could be used as array key..
                        $people[$i] = array(
                                "id" =>
        $data["people"][$i]["id"],
                                "nickname" =>                   $data["people"][$i]["nickname"],
                                "email" =>
        $data["people"][$i]["email"],
                                "first_name" =>         $data["people"][$i]["first_name"],
                                "last_name" =>          $data["people"][$i]["last_name"],
                                "address" =>                    $data["people"][$i]["address"],
                                "zip_code" =>                   $data["people"][$i]["zip_code"],
                                "town" =>
        $data["people"][$i]["town"],
                                "href" =>
        $data["people"][$i]["href"],

                        );

                        //append roles to $people
                        $roleids = $data["people"][$i]["links"]["roles"];
                        $j = 0;
                        foreach ($roleids as $id) {
                                if (array_key_exists($id, $roles)) {
                                        $roleindex = "role" . $j;
                                        $people[$i][$roleindex] = $roles[$id];
                                }
                                $j++;
                        }
                        //append phone numbers to $people
                        if (array_key_exists("phone_numbers", $data["people"][$i]["links"])) {
                                $numids = $data["people"][$i]["links"]["phone_numbers"];
                                $j = 0;
                                foreach ($numids as $id) {
                                        if (array_key_exists($id, $numbers)) {
                                                $numindex = "phone" . $j;
                                                $people[$i][$numindex] = $numbers[$id];
                                        }
                                        $j++;
                                }
                        }
                }


                return $people;

        }

}
//************************************************************************//



//Debug out
/*
//init scoutDB
$db = new scoutDB;

//JSON request on groupid
$people = $db->getGroups($groups2get);


echo "<html><body><table>";
for ($i = 0; $i < count($people); $i++) {
        echo "<tr>";
        foreach ($people[$i] as $value) {
                        echo "<td>" . utf8_decode($value) . "</td>";

        }
        echo "</tr>";
```

```
}
echo "</table></body></html>";
echo count($people);
//var_dump($decoded["people"]);
*/
?>
```

Originale Version (2015):
Olivier Brian / Zephir

Überarbeitete Version 2018:
Michael Schär