

# Postman-App - HTTPful

## Inhalt:

1. Theorie
2. Erstes Beispiel für ein Webservice mit PHP (Localhost - xampp).
3. Beispiel mit Zugriff auf eine externe API bei [www.github.com](http://www.github.com)
4. Beispiel mit einer externe API bei <https://jsonplaceholder.typicode.com/>

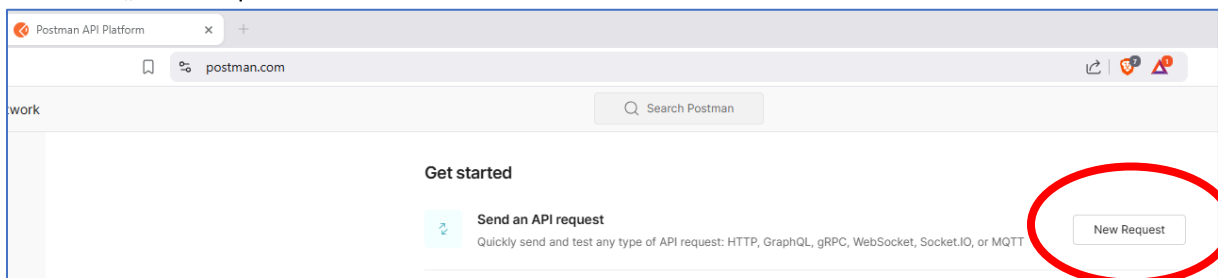
## 1.)Installation

**Zum Testen von APIs.** Sendet Requests und holt Daten im JSON-Format.

Ursprünglich als Erweiterung zum Chrome-Browser entstanden, ist es nun auch separat als Chrome-App verfügbar.

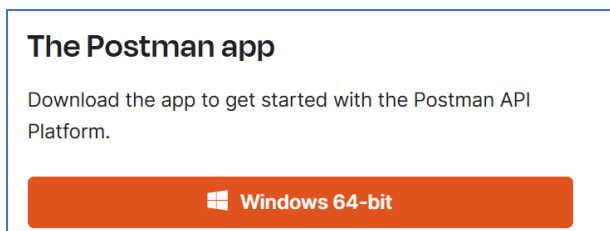
**Möglichkeit a)** online mit einer Registrierung (E-Mail, Username und Passwort) – inkl. E-Mail - Verify <https://www.postman.com/> oder [www.postman-echo.com](http://www.postman-echo.com)

Klicke auf „New Request“



**Möglichkeit b)** Installation:

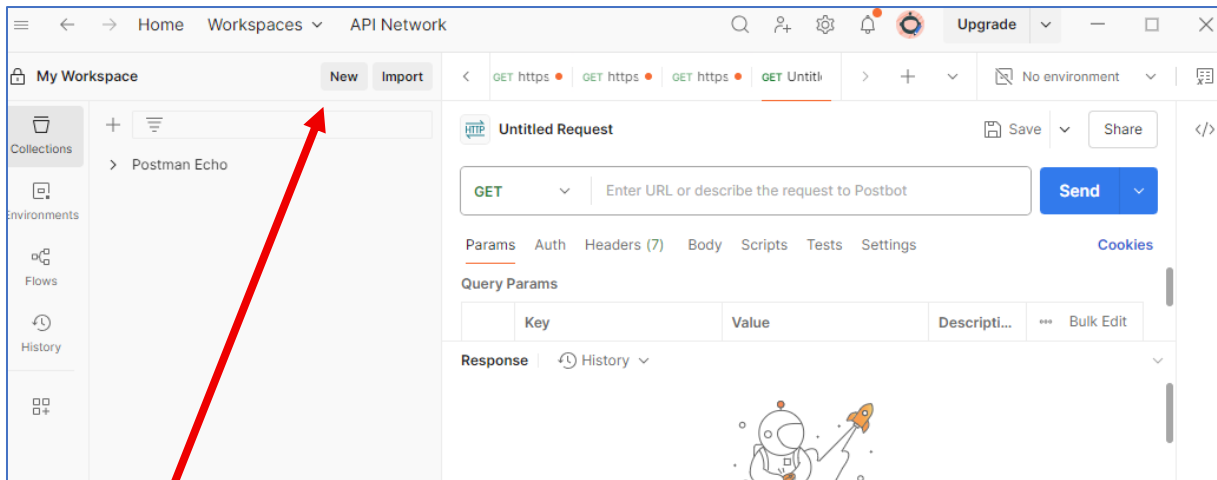
Download von <https://www.postman.com/downloads/>



Dann das Setup ausführen.

HTTP GET lässt sich im Browser testen, indem wir den URL aufrufen, aber für alle anderen Verben benötigen wir ein Tool zur Unterstützung. **Hier bietet sich Postman an, der uns eine grafische Oberfläche zur Verfügung stellt**, mit der sich HTTP-Anfragen in allen Formen, Farben und Varianten modellieren lassen. Postman steht als Chrome-App für alle Plattformen zur Verfügung.

Mit Postman hat man eine Möglichkeit, wie man eine API ohne Frontend und ohne Browser auf Herz und Nieren testen kann.



Klicke auf New und dann HTTP.

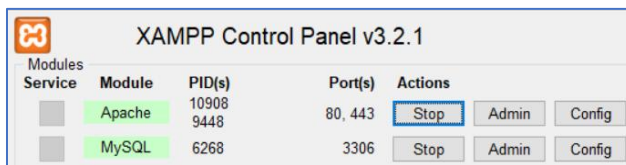
Weiter Info: <https://www.getpostman.com/docs/>

## 2)Erstes Beispiel für ein kleines Webservice mit PHP (Localhost - XAMPP).

Erstelle in xampp/htdocs einen Ordner „postman.test“ und darin ein PHP-Script mit einfacher echo-Ausgabe. Das soll später mit Postman abgerufen werden.

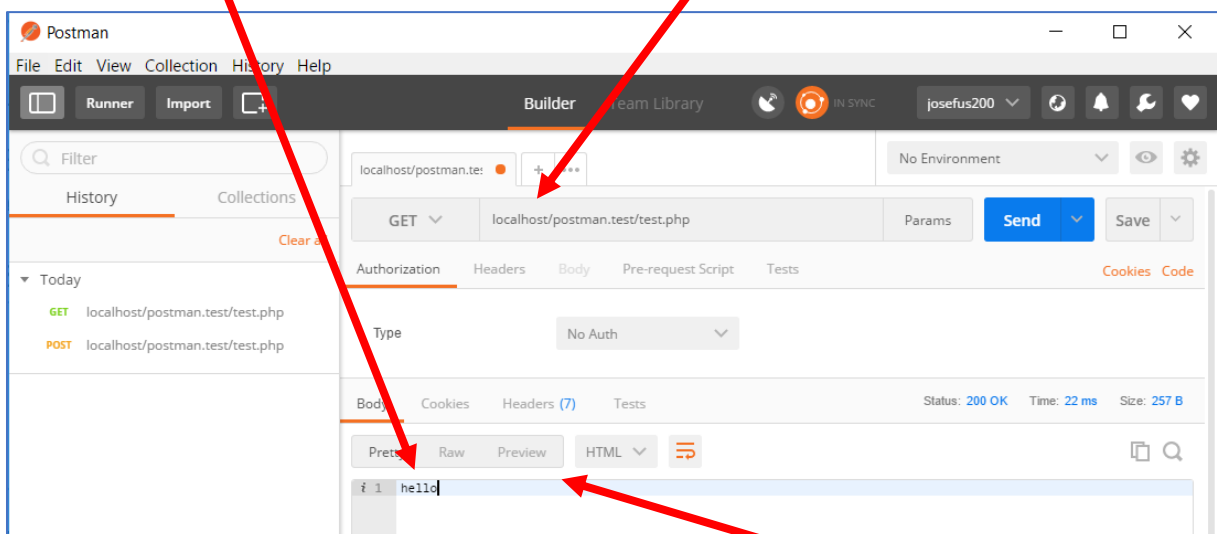
```
1 <?php
2
3 echo "hallo";
4 ?>
```

Starte auch den XAMPP-Server.



Dann ist die URL einzugeben. Beachte den Pfad zum htdocs Ordner und zum php-File. Dann „Send“.

Das Ergebnis sieht man unten:



Wir haben hier mit der „get“-Methode gearbeitet. Mit Klick auf „Preview“ sieht es noch besser aus.

## 2)Erstellen von JSON – Inhalt für eine JSON-Response

Die JSON-Response kann der Kunde nutzen. Er kann eine Abfrage durchführen.

Beispiel: eine Android-App will „results“ von der Datenbank, wird zuerst eine Anfrage („request“) an das API gesendet, welche dann eine Antwort („response“) sendet. Diese ist JSON-Format und kann gelesen werden.

Lösche das „echo“ und schreibe zuerst ein Array: \$arr mit einem Status-Code und einer Message. Dann Daten aus der Datenbank, die man anbietet.

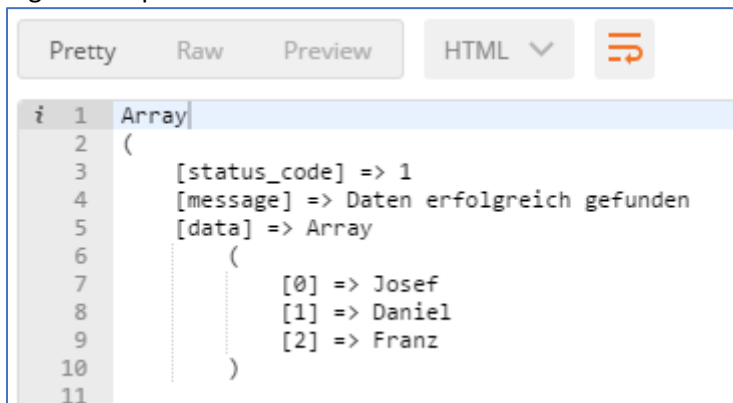
```
1 <?php
2 //DB Ergebnisse
3 $results = ['Josef', 'Daniel', 'Franz'];
4
5 $arr['status_code'] = 1;
6 $arr['message'] = "Daten erfolgreich gefunden";
7 $arr['data'] = $results;
8
9 print_r($arr);
10 ?>
```

Speichern: test2.php

Code:

```
<?php
//DB Ergebnisse
$results = ['Josef','Daniel','Franz'];
$arr['status_code'] = 1;
$arr['message'] = "Daten erfolgreich gefunden";
$arr['data'] = $results;
print_r($arr);
?>
```

Ergebnis in postman:



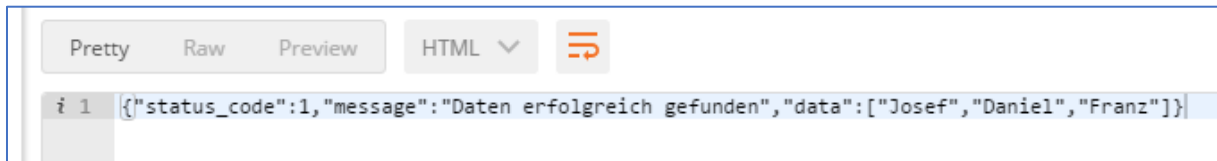
```
Pretty Raw Preview HTML
i 1 Array
2 (
3     [status_code] => 1
4     [message] => Daten erfolgreich gefunden
5     [data] => Array
6         (
7             [0] => Josef
8             [1] => Daniel
9             [2] => Franz
10        )
11    )
```

Aber: Das Ergebnis ist nicht nutzbar, da der Client dieses Format nicht lesen kann. Daher muss es lesbar gemacht werden:

Ändere den Code, um die Daten in JSON-Format zu ändern für die Ausgabe:  
Zuerst statt der „print()“-Funktion benutze die „echo json\_encode()“-Funktion.

```
7 $arr['data'] = $results;
8
9 echo json_encode($arr);|
10 ?>
```

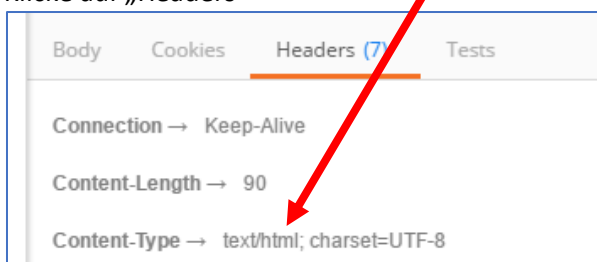
Resultat in lesbaren JSON-Format:



In JSON betrachten: klicke auf die Option „JSON“:



Hier kann man auch den „Content-Type“ ansehen, der HTML ist: Klicke auf „Headers“

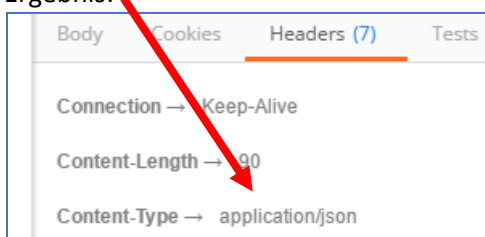


Der sichere Weg, um Antworten zu senden ist aber, den Type in JSON zu ändern. Dazu schreibe folgenden Code dazu:

header('Content-Type: application/json');

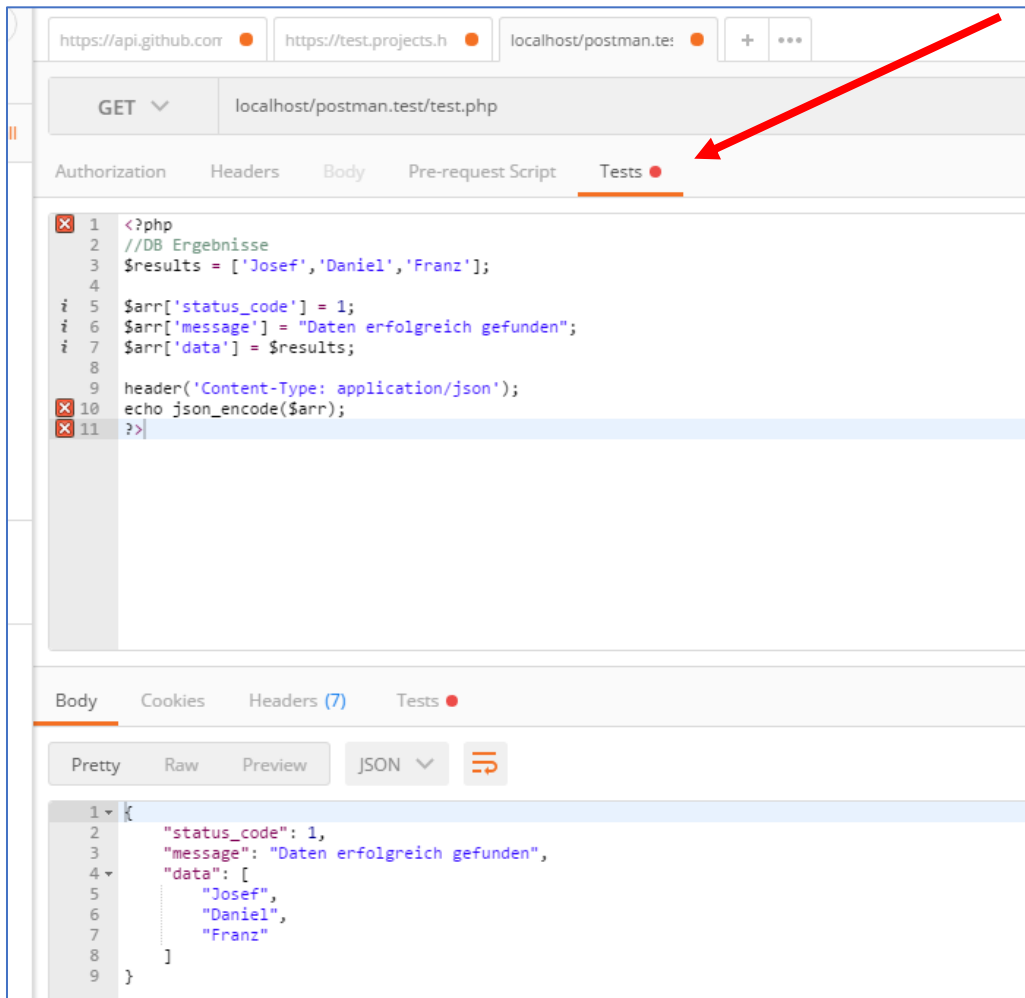
```
7 $arr['data'] = $results;
8
9 header('Content-Type: application/json');
10 echo json_encode($arr);
11 ?>
```

Ergebnis:



**INFO:**

Man benötigt keinen Editor, da man in Postman selbst den Code schreiben kann. Klicke auf „Tests“:



### 3) Beispiel mit Zugriff auf eine externe API bei [www.github.com](https://www.github.com)

Benötigt immer eine sichere Verbindung <https://>

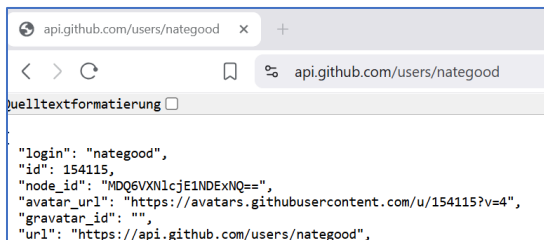
Github.com stellt ein API zur Verfügung, mit dem auf Daten zugegriffen werden kann. Das nutzen wir, um mit Requests darauf zuzugreifen und Abfragen zu bekommen.

Quelle und Info: <https://developer.github.com/>  
<https://docs.github.com/de/rest?apiVersion=2022-11-28>

Gib folgende Adresse ein:

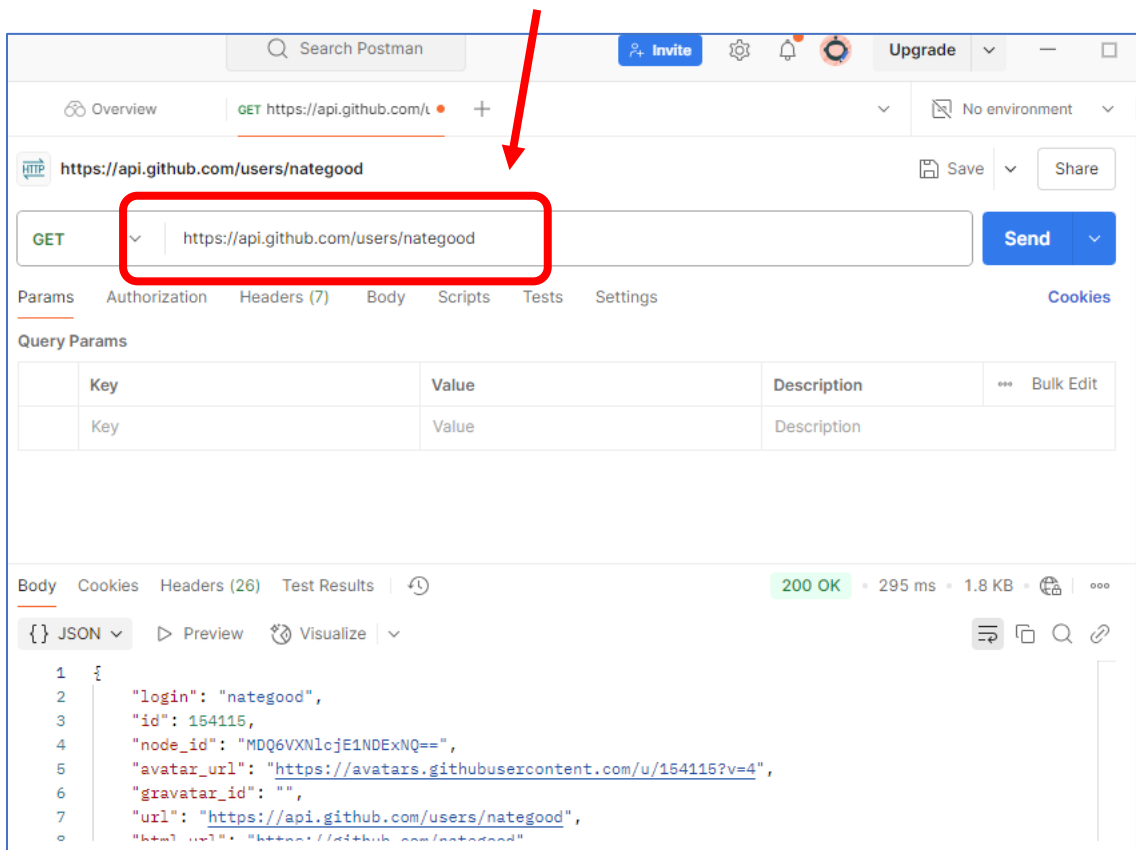
<https://api.github.com/users/nategood> und klicke auf „Send“.

Die GET-Abfrage kann man auch im Browser abfragen, ABER mit postman sind auch alle anderen Möglichkeiten vorhanden (POST, DELETE)



```
"login": "nategood",
"id": 154115,
"node_id": "MDQ6VXNlcjE1NDExNQ==",
"avatar_url": "https://avatars.githubusercontent.com/u/154115?v=4",
"gravatar_id": "",
"url": "https://api.github.com/users/nategood",
```

Ist „JSON“ eingestellt erhält man eine schöne Darstellung für diesen User:

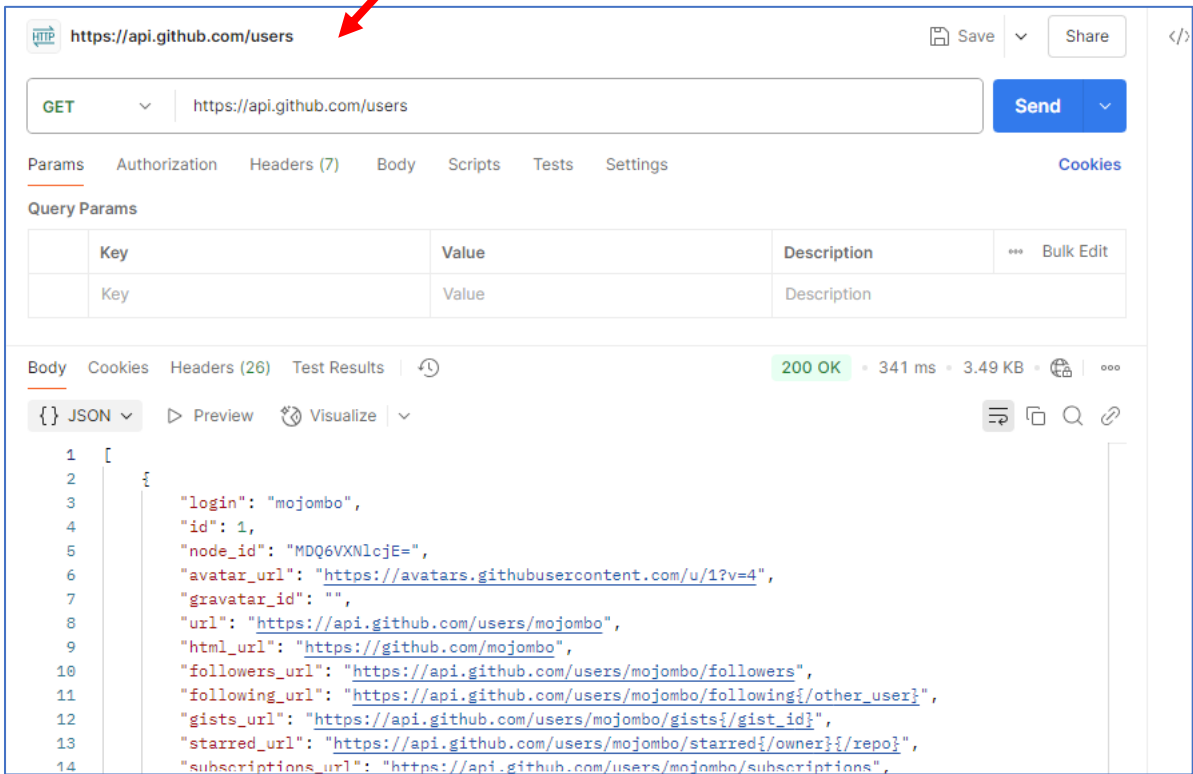


The screenshot shows the Postman interface. A red arrow points to the URL field in the request editor, which contains `https://api.github.com/users/nategood`. Below the request editor, the 'Query Params' section is empty. The 'Body' section shows the response in JSON format:

```
{
  "login": "nategood",
  "id": 154115,
  "node_id": "MDQ6VXNlcjE1NDExNQ==",
  "avatar_url": "https://avatars.githubusercontent.com/u/154115?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/nategood",
  "html_url": "https://github.com/nategood"
}
```

Will man alle User anzeigen lassen, lasse den User „nategood“ weg:

GET <https://api.github.com/users>



https://api.github.com/users

GET <https://api.github.com/users> Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

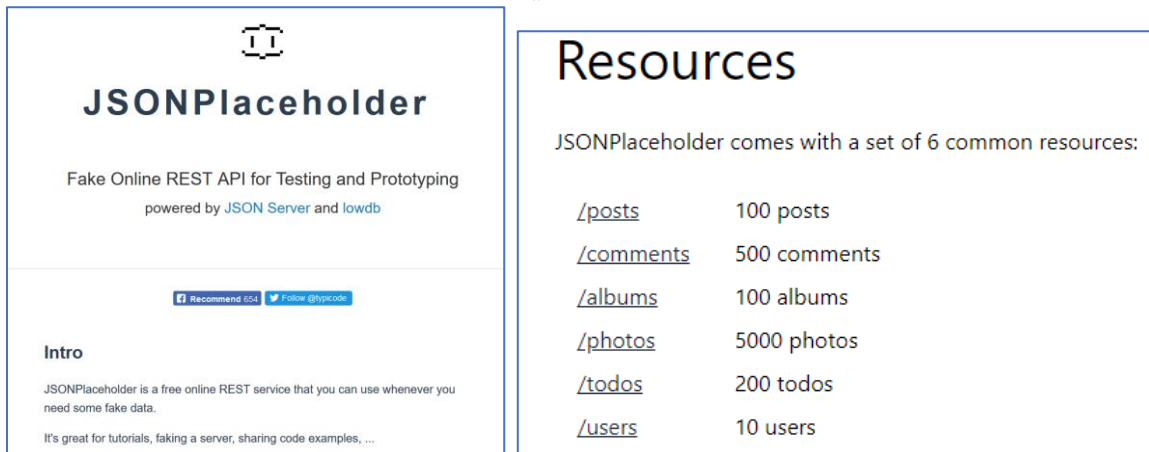
Body Cookies Headers (26) Test Results 200 OK • 341 ms • 3.49 KB

{ } JSON Preview Visualize

```
1 [
2   {
3     "login": "mojombo",
4     "id": 1,
5     "node_id": "MDQ6VXNlcjE=",
6     "avatar_url": "https://avatars.githubusercontent.com/u/1?v=4",
7     "gravatar_id": "",
8     "url": "https://api.github.com/users/mojombo",
9     "html_url": "https://github.com/mojombo",
10    "followers_url": "https://api.github.com/users/mojombo/followers",
11    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
12    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
13    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}/{/repo}",
14    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
```

#### 4) Beispiel mit einer externen API bei <https://jsonplaceholder.typicode.com/>

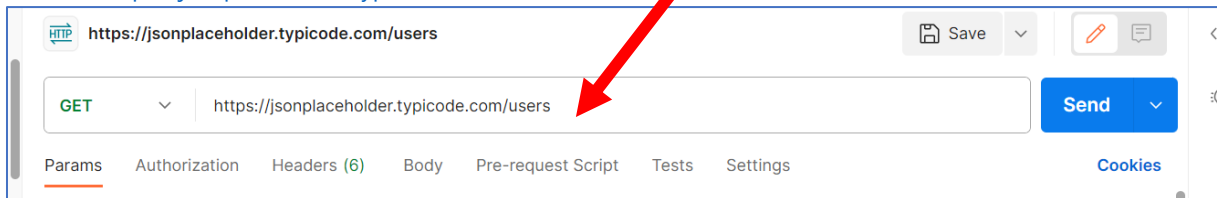
Öffne die Seite und scrolle nach unten bis zu „Resources“ für 6 Quellen:



The screenshot shows the JSONPlaceholder website. On the left, there is a header with the logo and the text "JSONPlaceholder Fake Online REST API for Testing and Prototyping powered by JSON Server and lowdb". Below this is an "Intro" section. On the right, there is a "Resources" section with the text "JSONPlaceholder comes with a set of 6 common resources:" followed by a list of resources:

<a href="#">/posts</a>	100 posts
<a href="#">/comments</a>	500 comments
<a href="#">/albums</a>	100 albums
<a href="#">/photos</a>	5000 photos
<a href="#">/todos</a>	200 todos
<a href="#">/users</a>	10 users

Gib ein: <https://jsonplaceholder.typicode.com/users>



The screenshot shows a REST client interface. The URL bar contains "https://jsonplaceholder.typicode.com/users". The method is set to "GET". A red arrow points to the URL. Below the URL bar, there are tabs for "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", and "Settings". A "Send" button is visible on the right.

Ergebnis:



The screenshot shows the "Body" tab of the REST client interface. The response is displayed in JSON format:

```
1 [
2   {
3     "id": 1,
4     "name": "Leanne Graham",
5     "username": "Bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas Light",
9       "suite": "Apt. 556",
10      "city": "Gwenborough",
11      "zipcode": "92998-3874",
12      "geo": {
13        "lat": "-37.3159",
14        "lng": "81.1496"
15      }
16    },
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
```

Wähle auch



The screenshot shows the REST client interface with the URL bar containing "https://jsonplaceholder.typicode.com/photos". The method is set to "GET".