**Postman The Complete Guide - REST API Testing**

**Section 1: Introduction and the first steps in Postman**

A picture containing graphical user interface

Description automatically generated

Snipets > Set a global variable

GET

*let* response = pm.response.json();

pm.globals.set("uuid", response.uuid);

POST

{

    "name": "John",

    "email": "johm@axample.com",

    "id": "*{{uuid}}*"

}

Snipets > Status code: Code is 200

pm.test("Status code is 200", *function* () {

    pm.response.to.have.status(200);

});

**Section 2: Creating API requests with Postman**

Если переменная с адресом сохранена как переменная коллекции (Не глобальная) то запрос будет фэйлится пока не будет сохранен в этой коллекции

{{baseUrl}}/tools

**Test – Set Collection Variables**

*let* response = pm.response.json();

pm.collectionVariables.set("apiToken", response.accessToken);

There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer from the server.

[Headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#headers)

[HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) from a request follow the same basic structure of an HTTP header: a case-insensitive string followed by a colon (':') and a value whose structure depends upon the header. The whole header, including the value, consist of one single line, which can be quite long.

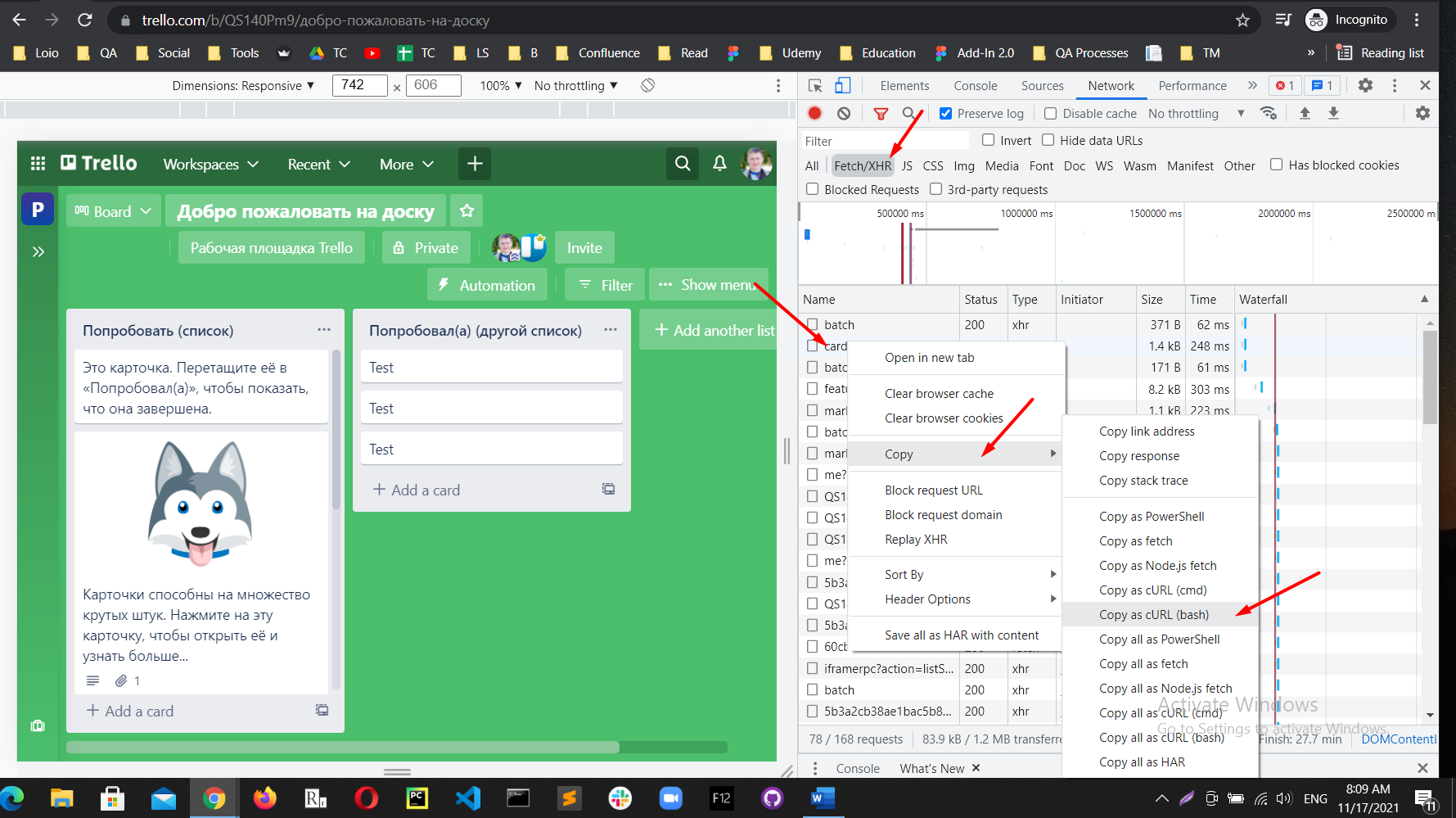
Many different headers can appear in requests. They can be divided in several groups:

* [General headers](https://developer.mozilla.org/en-US/docs/Glossary/General_header), like [Via](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Via), apply to the message as a whole.
* [Request headers](https://developer.mozilla.org/en-US/docs/Glossary/Request_header), like [User-Agent](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent) or [Accept](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept), modify the request by specifying it further (like [Accept-Language](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Language)), by giving context (like [Referer](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer)), or by conditionally restricting it (like [If-None](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None)).
* [Representation headers](https://developer.mozilla.org/en-US/docs/Glossary/Representation_header) like [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) that describe the original format of the message data and any encoding applied (only present if the message has a body).

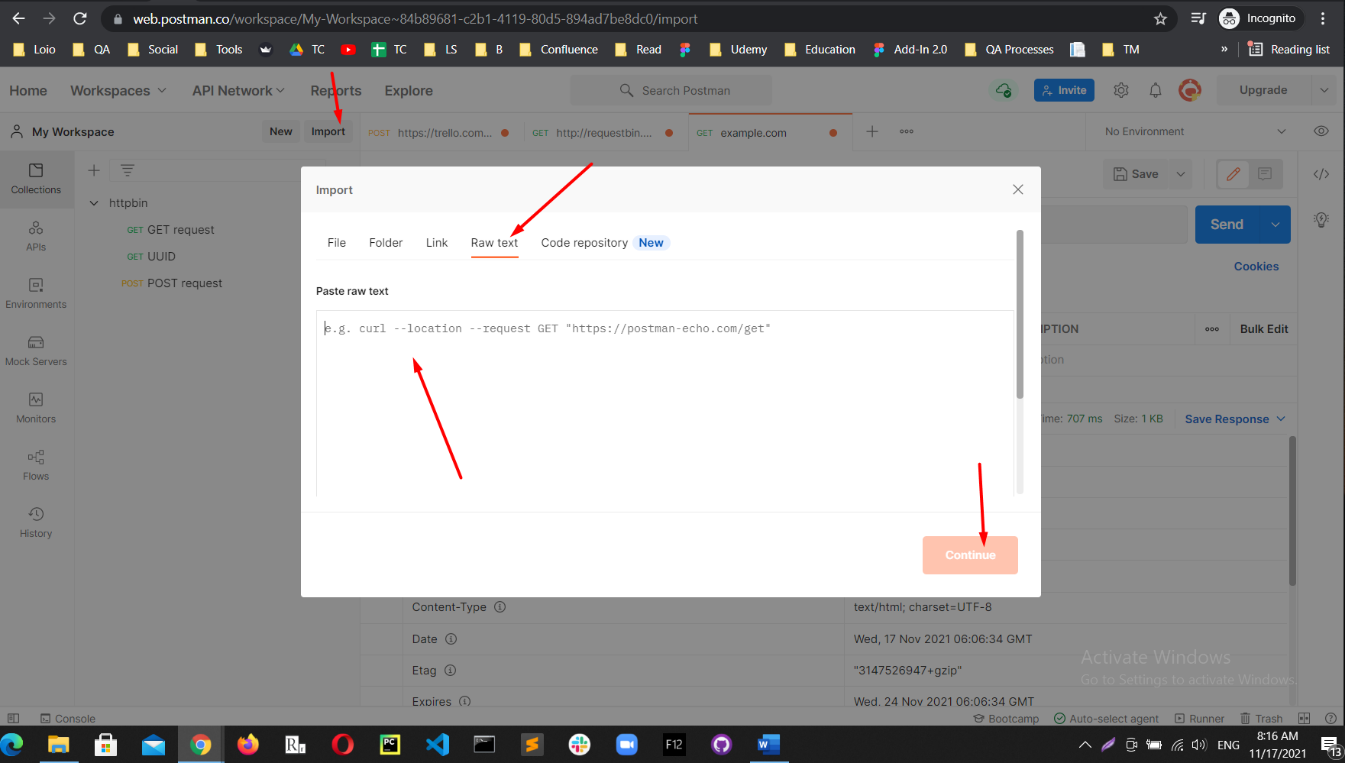
Text

Description automatically generated

Copy API request from browser and import it on Postman



In postman



eGDvNuNzjWORru49z2tuyzL8Uvjd20lB

**Section 3: Writing tests and scripts**

pm.**test**("Status code is 200", **function** () {

    pm.response.to.have.status(200);

});

#### **Установка и получение переменных**

// глобальные переменные

pm.globals.set(“key”, “value”);

pm.globals.get(“key”);

// переменные окружения

pm.environment.set(“key”, “value”);

pm.environment.get(“key”);

// локальные переменные

pm.variables.set(“key”, “value”);

pm.variables.get(“key”); // если нет локальной, будет искать на уровне выше

#### **Тестирование или asserts**

// с использованием анонимной функции и специальных assert конструкций

pm.test(“Название теста”, **function** () {

pm.response.to.be.success;

pm.expect(“value”).to.be.true;

pm.expect(“other”).to.equal(“other”);

});

// с использованием простого условия и массива tests

tests[“Название теста”] = (“a” != “b”);

tests[“Название теста 2”] = true;

#### **Создание запросов**

// пример get запроса

pm.sendRequest(“https://postman-echo.com/get”, function (err, res) {

console.log(err);

console.log(res);

});

// пример post запроса

**let** data = {

url: “https://postman-echo.com/post”,

method: “POST”,

body: { mode: “raw”, raw: JSON.stringify({ key: “value” })}

};

pm.sendRequest(data, **function** (err, res) {

console.log(err);

console.log(res);

});

#### **Получение ответа для основного запроса**

pm.response.json(); // в виде json

pm.response.text(); // в виде строки

responseBody; // в виде строки

#### **Работа со встроенными библиотеками**

Документация регламентирует наличие некоторого количества встроенных библиотек, среди которых — tv4 для валидации json, xml2js конвертер xml в json, crypto-js для работы с шифрованием, atob, btoa и др.

// подключение xml2js

**var** xml2js = require(“xml2js”);

// преобразование простого xml в json объект

xml2js.parseString("<root>Hello xml2js!</root>", **function**(err, res) {

console.log(res);

});

Некоторые из библиотек, например, как tv4 не требуют прямого подключения через require и доступны по имени сразу.

#### **Получение информации о текущем скрипте**

pm.info.eventName; // вернет test или prerequest в зависимости от контекста

pm.info.iteration; // текущая итерация в Runner

pm.info.iterationCount; // общее количество итераций

pm.info.requestName; // название текущего запроса

pm.info.requestId; // внутренний идентификатор запроса

#### **Управление последовательностью запросов из скрипта**

Стоит отметить, что данный метод работает только в режиме запуска всех скриптов.

// установить следующий запрос

postman.setNextRequest(“Название запроса”); // по названию

postman.setNextRequest(ID); // по идентификатору

// остановить выполнение запросов

postman.setNextRequest(null);

После перехода на следующий запрос Postman возвращается к линейному последовательному выполнению запросов.

#### **Создание глобального хелпера**

В некоторых случаях вам захочется создать функции, которые должны быть доступны во всех запросах. Для этого в первом запросе в секции “Pre-request Script” напишите следующий код:

// создаем и сохраняем хелпер для глобального использования

pm.environment.set("pmHelper", **function** **pmHelper**() {

**let** helpers = {};

helpers.usefulMethod = **function**() {

console.log(“It is helper, bro!”);

};

**return** helpers;

} + '; pmHelper();');

А в последующих скриптах пользуемся им так:

// получаем объект

**var** pmHelper = eval(pm.environment.get("pmHelper"));

// вызываем наш метод

pmHelper.usefulMethod();

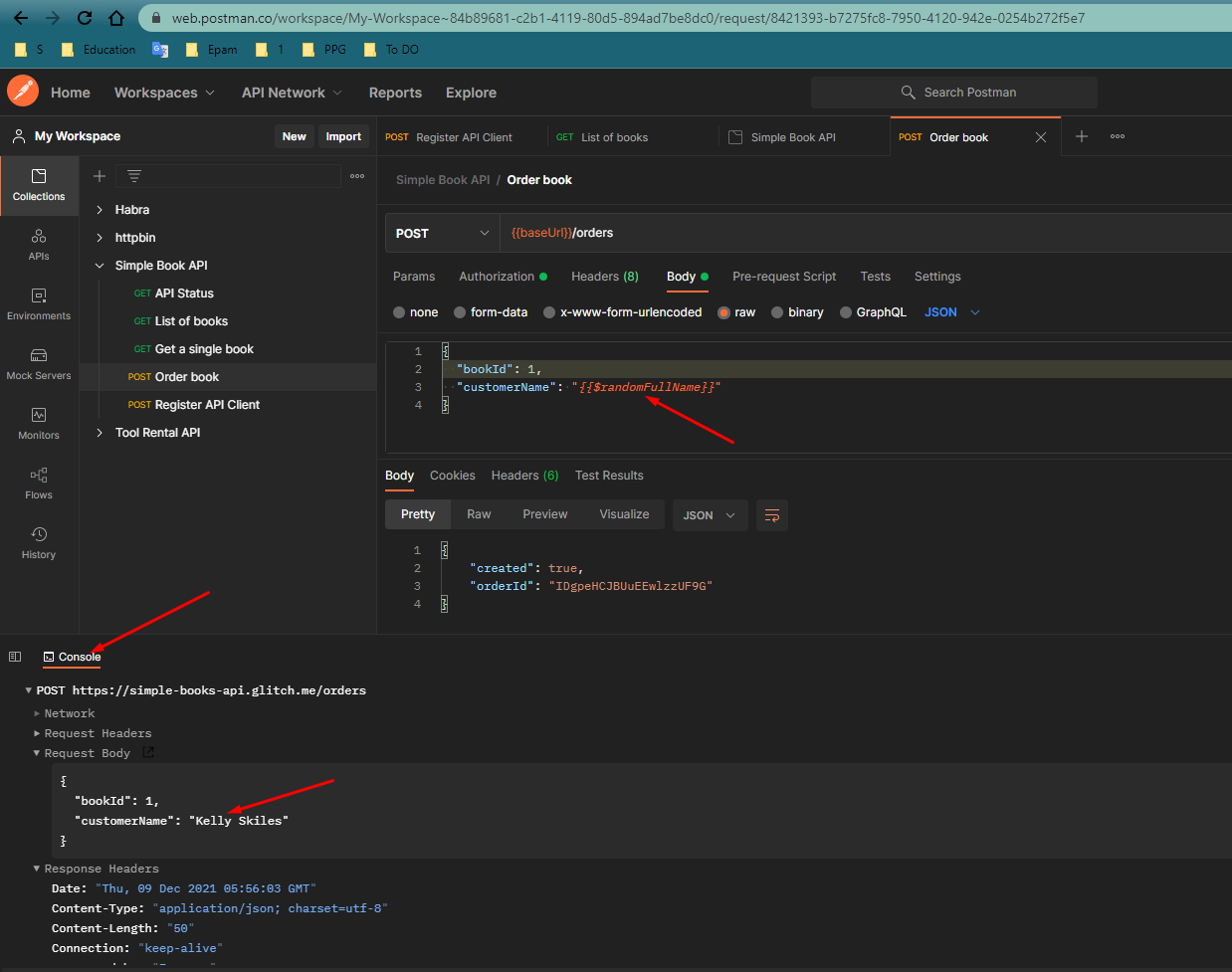
Random Test Data

{

  "bookId": 1,

  "customerName": "*{{$randomFullName}}*"

}



Обновление части информации

PATCH

{{baseUrl}}/orders/zJVpqu33DVrUiJ4NeO5NU

{

  "customerName": "John *{{$randomLastName}}*"

}

Парсим ответ

*const* response = pm.response.json();

*console*.log(response.status);

*console*.log(response['status']);

pm.test("Status should be OK", () *=>* {

   pm.expect(response.status).to.eql("OK");

Затем проверяем что ответ равняется нужному нам значению

Extracting data from the response

*const* response = pm.response.json();

pm.globals.set("orderId", response.orderId);

Request execution order

postman.setNextRequest("List of books");

postman.setNextRequest(null); // STOP !!!

Response body: JSON value check

pm.test("Board should be private", *function* () {

*var* jsonData = pm.response.json();

    pm.expect(jsonData.prefs.permissionLevel).to.eql("private");

});

**Section 4: Writing tests and scripts using variables**

Accessing variables in the scripts

pm.globals.get(“variable\_key”);

pm.globals.set(“variable\_key”, variable\_value); // environment

pm.globals.unset(“variable\_key”); // Delete variable

pm.globals.clear(); // Delete all global variables

Pre-request script in Postman

pm.environment.set(“boardName”, getRandomBoardName());

function getRandomBoardName() {

const boardName = “My new board” + parseInt(Math.rendom() \* 10000);

console.log(boardName);

return boardName;

}

var currentBoardNumber **=** postman.getEnvironmentVariable("currentBoardNumber");

**if** (currentBoardNumber) {

    currentBoardNumber**++**;

} **else** {

    currentBoardNumber **=** 1;

}

var boardName **=** "My board " **+** currentBoardNumber;

postman.setEnvironmentVariable("boardName", boardName);

postman.setEnvironmentVariable("currentBoardNumber", currentBoardNumber);