**POSTMAN**

[Postman API Platform | Sign Up for Free](https://www.postman.com/)

[Introduction | Postman Learning Center](https://learning.postman.com/docs/getting-started/introduction/)

[Postman JavaScript reference | Postman Learning Center](https://learning.postman.com/docs/writing-scripts/script-references/postman-sandbox-api-reference/)

https://learning.postman.com/docs/writing-scripts/script-references/test-examples/

Оглавление

[SetUp 1](#_Toc90383310)

[Установка и получение переменных 1](#_Toc90383311)

[Tests 1](#_Toc90383312)

[Simple Tests 1](#_Toc90383313)

[Schema validation 2](#_Toc90383314)

[preRequest 2](#_Toc90383315)

# SetUp

## 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

pm.collectionVariables.clear();

*console*.log(pm.info.eventName); // вернет test или prerequest в зависимости от контекста

*console*.log(pm.info.iteration); // текущая итерация в Runner

*console*.log(pm.info.iterationCount); // общее количество итераций

*console*.log(pm.info.requestName); // название текущего запроса

*console*.log(pm.info.requestId); // внутренний идентификатор запроса

# Tests

## Simple Tests

Для анализа данных JSON используйте следующий синтаксис:

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

Для синтаксического анализа XML используйте следующее:

*const* responseJson = xml2Json(pm.response.text());

pm.test("200", *function*() {

    pm.response.to.have.status(200);

})

pm.test("OK", *function*() {

    pm.response.to.be.ok;  // проверка статуса

})

pm.test("Person is Jane", () *=>* {

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

  pm.expect(responseJson.name).to.eql("Jane");

  pm.expect(responseJson.age).to.eql(23);

});

pm.test("Successful POST request", () *=>* {

  pm.expect(pm.response.code).to.be.oneOf([201,202]);

});

pm.test("JSON", *function*() {

    pm.response.to.be.json; // проверка формата

})

try {

    pm.environment.set("userId", pm.response.json().args.userId);

} catch(e) {

*console*.log(e);

}

pm.test("Body", *function*() {

    pm.expect(pm.response.json().data.username).to.equal("Bob"); // Сравнение получаемого значения

})

## Schema validation

*var* dataJson = pm.response.json().data.profile;

*var* scheme = {

    "type": "object",

    "required": ["userId", "username"],

    "properties": {

        "userId": {"type": "integer"},

        "username": {"type": "string"},

        "scores": {

            "type": "array",

            "items": "integer"

        },

        "age": {"type": "integer"},

        "rating": {

            "type": "object",

            "properties": {

                "min": {"type": "integer"},

                "max": {"type": "integer"}

            }

        }

    }

}

pm.test("SHEMA", *function*() {

*var* isValidScheme = tv4.validate(dataJson, scheme, true, true)

    pm.expect(isValidScheme).to.be.true;

})

//Вывод в лог

*console*.log(dataJson)

// пример get запроса внутри Test (сам GET может выполняться на другой адрес)

pm.sendRequest("https://swapi.dev/api/people/1/", *function*(*err*, *res*) {

*console*.log(err);

*console*.log(res);

})

*var* v1 = pm.response.json(); // в виде json

*var* v2 = pm.response.text(); // в виде json

*var* v3 = responseBody; // в виде json

*console*.log("----Ответ получен в виде в виде json----")

*console*.log(v1)

*console*.log("----Ответ получен в виде в виде строки----")

*console*.log(v2)

*console*.log("----Ответ получен в виде в виде строки----")

*console*.log(v3)

## preRequest

*var* t = 0

*const* createRequest = {

  url: 'https://portal5test.cbr.ru/back/rapi2/messages/'+ pm.collectionVariables.get('MsgId'),

  method: 'GET',

  header: {

    'Content-type': 'application/json',

    'Authorization': pm.collectionVariables.get('Authorization')

  },

}

*console*.log("Ожидание получения статуса")

*var* myFunc03 = *function* () {

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

*var* s = res.json().Status

      setTimeout(*function* () {

          if (s != "processing" && s != "error" && t < 50 && s != "rejected") {

*console*.log(s)

              t++

        myFunc03();

    }

}, 3000);

})}

myFunc03();

# 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();