**HW\_2 Postman** ( [github.com/Strielka/Postman/tree/main/2\_Postman](http://github.com/Strielka/Postman/tree/main/2_Postman) )

*http://162.55.220.72:5005/first*

1. Отправить запрос.

2. Проверить статус код 200

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

pm.response.to.have.status(200);

});

3. Проверить, что в body приходит правильный string.

pm.test("Body is correct", *function* () {

pm.response.to.have.body("This is the first responce from server!");

});

*http://162.55.220.72:5005/user\_info\_3*

1. Отправить запрос.

2. Статус код 200

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

pm.response.to.have.status(200);

});

3. Спарсить response body в json.

*let*  jsonData = pm.response.json()

*console*.log(jsonData)

4. Проверить, что name в ответе равно name в request (name вбить руками.)

*let*  jsonData = pm.response.json()

*let*  test\_name = jsonData.name

pm.test("name test", *function* () {

pm.expect(test\_name).to.eql("Lana");

});

5. Проверить, что age в ответе равно age в request (age вбить руками.)

*let*  test\_age = jsonData.age

pm.test("age test", *function* () {

pm.expect(test\_age).to.eql("23");

});

6. Проверить, что salary в ответе равно salary в request (salary вбить руками.)

*let*  test\_salary = jsonData.salary

pm.test("salary test", *function* () {

pm.expect(test\_salary).to.eql(400);

});

7. Спарсить request.

*let* req = request.data

*console*.log(req)

8. Проверить, что name в ответе равно name в request (name забрать из request.)

*let*  name\_empty = request.name

pm.test("test\_name\_1", *function* (){

pm.expect(request.data.name).to.eql(name\_empty)

});

9. Проверить, что age в ответе равно age в request (age забрать из request.)

*let*  age\_empty = request.age

pm.test("test\_age\_1", *function* (){

pm.expect(request.data.age).to.eql(age\_empty)

});

10. Проверить, что salary в ответе равно salary в request (salary забрать из request.)

*let*  salary\_empty = request.salary

pm.test("test\_salary\_1", *function* (){

pm.expect(request.data.salary).to.eql(salary\_empty)

});

11. Вывести в консоль параметр family из response.

*console*.log(jsonData.family)

12. Проверить что u\_salary\_1\_5\_year в ответе равно salary\*4 (salary забрать из request)

pm.test("check\_1\_5\_salary", *function* () {

pm.expect(jsonData.family.u\_salary\_1\_5\_year).to.eql(+request.data.salary \* 4);

});

*http://162.55.220.72:5005/object\_info\_3*

1. Отправить запрос.

2. Статус код 200

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

pm.response.to.have.status(200);

});

3. Спарсить response body в json.

*let*  jsonData = pm.response.json()

*console*.log(jsonData)

4. Спарсить request.

*let* req = request.data

*console*.log(req)

5. Проверить, что name в ответе равно name в request (name забрать из request.)

*let*  params = pm.request.url.query.toObject()

pm.test("check name request", *function* () {

pm.expect(jsonData.name).to.eql(params.name)

});

6. Проверить, что age в ответе равно age в request (age забрать из request.)

pm.test("check age request", *function* () {

pm.expect(jsonData.age).to.eql(params.age)

});

7. Проверить, что salary в ответе равно salary в request (salary забрать из request.)

pm.test("check salary request", *function* () {

pm.expect(jsonData.salary).to.eql(+params.salary )

});

8. Вывести в консоль параметр family из response.

*console*.log(jsonData.family)

9. Проверить, что у параметра dog есть параметры name.

pm.test("dog\_have\_name", *function*() {

pm.expect(jsonData.family.pets.dog.name).not.eq(undefined)

})

10. Проверить, что у параметра dog есть параметры age.

pm.test("dog\_have\_age", *function*() {

pm.expect(jsonData.family.pets.dog.age).not.eq(undefined)

})

11. Проверить, что параметр name имеет значение Luky.

pm.test("dog\_have\_name\_Luky", *function*() {

pm.expect(jsonData.family.pets.dog.name).to.eq("Luky")

});

12. Проверить, что параметр age имеет значение 4.

pm.test("age\_is\_4", *function*() {

pm.expect(jsonData.family.pets.dog.age).to.eq(4)

});

*http://162.55.220.72:5005/object\_info\_4*

1. Отправить запрос.

2. Статус код 200

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

pm.response.to.have.status(200);

});

3. Спарсить response body в json.

*let*  jsonData = pm.response.json()

*console*.log(jsonData)

4. Спарсить request.

*let* req = request.data

*console*.log(req)

5. Проверить, что name в ответе равно name s request (name забрать из request.)

*let*  params = pm.request.url.query.toObject()

pm.test("check name request", *function* () {

pm.expect(jsonData.name).to.eql(params.name)

});

6. Проверить, что age в ответе равно age из request (age забрать из request.)

pm.test("check age request", *function* () {

pm.expect(jsonData.age).to.eql(+params.age)

­});

7. Вывести в консоль параметр salary из request.

*console*.log(params.salary)

8. Вывести в консоль параметр salary из response.

*console*.log(jsonData.salary)

9. Вывести в консоль 0-й элемент параметра salary из response.

*console*.log(jsonData.salary[0])

10. Вывести в консоль 1-й элемент параметра salary из response.

*console*.log(jsonData.salary[1])

11. Вывести в консоль 2-й элемент параметра salary из response.

*console*.log(jsonData.salary[2])

12. Проверить, что 0-й элемент параметра salary равен salary из request (salary забрать из request.)

pm.test("check salary\_0", *function* () {

pm.expect(jsonData.salary[0]).to.eql(+params.salary)

});

13. Проверить, что 1-й элемент параметра salary равен salary\*2 из request (salary забрать из request.)

pm.test("check salary\_1", *function* () {

pm.expect(+jsonData.salary[1]).to.eql(params.salary\*2)

});

14. Проверить, что 2-й элемент параметра salary равен salary\*3 из request (salary забрать из request.)

pm.test("check salary\_2", *function* () {

pm.expect(+jsonData.salary[2]).to.eql(params.salary\*3)

});

15. Создать в окружении переменную name

16. Создать в окружении переменную age

17. Создать в окружении переменную salary

18. Передать в окружение переменную name

pm.environment.set("name", jsonData.name)

19. Передать в окружение переменную age

pm.environment.set("age", jsonData.age )

20. Передать в окружение переменную salary

pm.environment.set("salary", jsonData. salary[0])

21. Написать цикл который выведет в консоль по порядку элементы списка из параметра salary.

*let*  elements = jsonData.salary

for (*let*  i = 0; i <= 2; i++) {

*console*.log(elements[i])

}

*http://162.55.220.72:5005/user\_info\_2*

1. Вставить параметр salary из окружения в request

2. Вставить параметр age из окружения в age

3. Вставить параметр name из окружения в name

4. Отправить запрос.

5. Статус код 200

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

pm.response.to.have.status(200);

});

6. Спарсить response body в json.

*let*  jsonData = pm.response.json()

*console*.log(jsonData)

7. Спарсить request.

*let* req = request.data

*console*.log(req)

8. Проверить, что json response имеет параметр start\_qa\_salary

pm.test('params start\_qa\_salary ', *function*(){

pm.expect(jsonData).to.have.property('start\_qa\_salary')

})

9. Проверить, что json response имеет параметр qa\_salary\_after\_6\_months

pm.test('params 6\_m\_salary ', *function*(){

pm.expect(jsonData).to.have.property('qa\_salary\_after\_6\_months')

})

10. Проверить, что json response имеет параметр qa\_salary\_after\_12\_months

pm.test('params 12\_m\_salary ', *function*(){

pm.expect(jsonData).to.have.property('qa\_salary\_after\_12\_months')

})

11. Проверить, что json response имеет параметр qa\_salary\_after\_1.5\_year

pm.test('params 1,5\_y\_salary ', *function*(){

pm.expect(jsonData).to.have.property('qa\_salary\_after\_1.5\_year')

})

12. Проверить, что json response имеет параметр qa\_salary\_after\_3.5\_years

pm.test('params 3,5\_y\_salary ', *function*(){

pm.expect(jsonData).to.have.property('qa\_salary\_after\_3.5\_years')

})

13. Проверить, что json response имеет параметр person

pm.test('response have person ', *function*(){

pm.expect(jsonData).to.have.property('person')

})

14. Проверить, что параметр start\_qa\_salary равен salary из request (salary забрать из request.)

pm.test("check salary", *function* () {

pm.expect(jsonData.start\_qa\_salary).to.eql(+request.data.salary)

});

15. Проверить, что параметр qa\_salary\_after\_6\_months равен salary\*2 из request (salary забрать из request.)

pm.test("check salary\_6\_m", *function* () {

pm.expect(jsonData.qa\_salary\_after\_6\_months).to.eql(request.data.salary\*2)

});

16. Проверить, что параметр qa\_salary\_after\_12\_months равен salary\*2.7 из request (salary забрать из request.)

pm.test("check salary\_12\_m", *function* () {

pm.expect(jsonData.qa\_salary\_after\_12\_months).to.eql(request.data.salary\*2.7)

});

17. Проверить, что параметр qa\_salary\_after\_1.5\_year равен salary\*3.3 из request (salary забрать из request.)

pm.test("check salary\_1.5\_y", *function* () {

pm.expect(jsonData["qa\_salary\_after\_1.5\_year"]).to.eql(request.data.salary\*3.3)

});

18. Проверить, что параметр qa\_salary\_after\_3.5\_years равен salary\*3.8 из request (salary забрать из request.)

pm.test("check salary\_3.5\_y", *function* () {

pm.expect(jsonData["qa\_salary\_after\_3.5\_years"]).to.eql(request.data.salary\*3.8)

});

19. Проверить, что в параметре person, 1-й элемент из u\_name равен salary из request (salary забрать из request.)

pm.test("u\_name-salary", *function* () {

pm.expect(jsonData.person.u\_name[1]).to.eql(+request.data.salary)

});

20. Проверить, что что параметр u\_age равен age из request (age забрать из request.)

pm.test("test age", *function* () {

pm.expect(jsonData.person.u\_age).to.eql(+request.data.age)

});

21. Проверить, что параметр u\_salary\_5\_years равен salary\*4.2 из request (salary забрать из request.)

pm.test("test salary\_5", *function* () {

pm.expect(jsonData.person.u\_salary\_5\_years).to.eql(request.data.salary\*4.2)

});

22. \*\*\*Написать цикл который выведет в консоль по порядку элементы списка из параметра person.

*let* element = jsonData.person

for (*let*  key in element) {

*console*.log('Key: ' + key + ', value: ' + element[key])

}