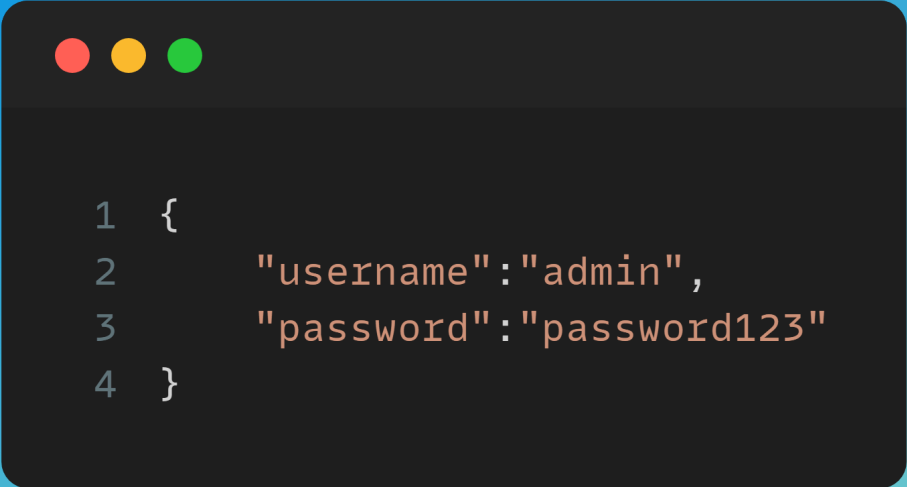


baka-pwapi-8

PUT API Request | API Chaining

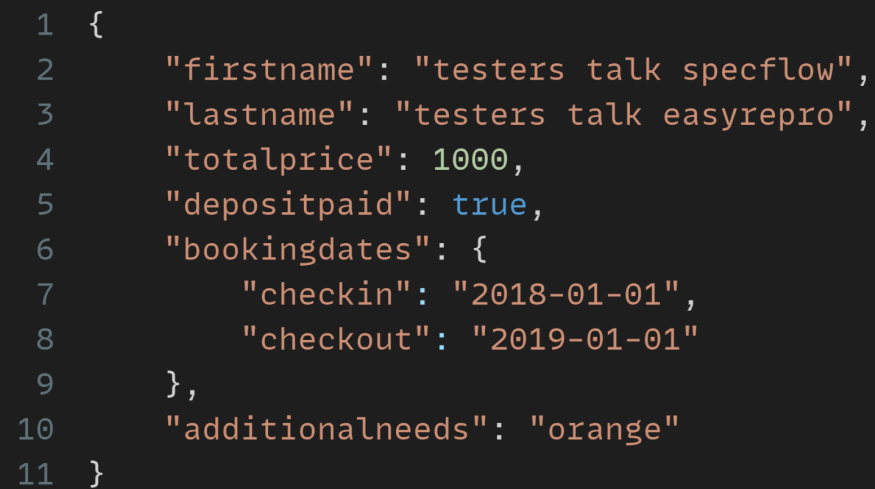
put needed a token so created token test data-



```
1  {  
2      "username": "admin",  
3      "password": "password123"  
4  }
```

codesnap.dev

put request body-



```
1  {
2    "firstname": "testers talk specflow",
3    "lastname": "testers talk easyrepro",
4    "totalprice": 1000,
5    "depositpaid": true,
6    "bookingdates": {
7      "checkin": "2018-01-01",
8      "checkout": "2019-01-01"
9    },
10   "additionalneeds": "orange"
11 }
```

codesnap.dev

post, get, token and put-

```
1 //
2 const { test, expect } = require("@playwright/test");
3 import { faker } from "@faker-js/faker";
4 const { DateTime } = require("luxon");
5 var dynamicPostRequest = require("../test-data/dynamicrequestbody.json");
6 import { stringFormat } from "../utils/commons";
7 const tokenRequestBody=require("../test-data/puttokenrequestbody.json");
8 const putRequestBody=require("../test-data/putrequestbody.json")
9
10 //first create data then get data.
11
12 test("create put api request", async ({
13   request,
14 }) => {
15
16
17   // const dynamicRequestBody=stringFormat(JSON.stringify(dynamicPostRequest),
18   // "testers talk cypress", "testers talk js", "banana")
19
20
21   //call the util function first.
22   var updatedRequestBody = stringFormat(
23     JSON.stringify(dynamicPostRequest),
24     "testers talk cypress",
25     "testers talk javascript",
26     "apple"
27   );
```

```
1 // create post api request using playwright
2 const postAPIResponse = await request.post("/booking", {
3   //convert the string received to json object.
4   //use json.parse and pass the string.
5   data: JSON.parse(updatedRequestBody),
6 });
7
8 // validate status code
9 console.log(await postAPIResponse.json());
10
11 expect(postAPIResponse.ok()).toBeTruthy();
12 expect(postAPIResponse.status()).toBe(200);
13
14 // validate api response json obj
15 const postAPIResponseBody = await postAPIResponse.json();
16
17 const bid=postAPIResponseBody.bookingid;
18
19 expect(postAPIResponseBody.booking).toHaveProperty("firstname", "testers talk cypress");
20 expect(postAPIResponseBody.booking).toHaveProperty("lastname", "testers talk javascript");
```

```
1 // validate api response nested json obj
2   expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
3     "checkin",
4     "2018-01-01"
5   );
6   expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
7     "checkout",
8     "2019-01-01"
9   );
10
11   console.log("=====post response=====")
12
13
14   console.log("=====get response=====")
15 //get api call using query params.
16   const getAPIResponse = await request.get(`/booking/${bid}`)
17
18   console.log(await getAPIResponse.json())
19
20   await expect(getAPIResponse.ok()).toBeTruthy();
21   await expect(getAPIResponse.status()).toBe(200);
```

```
1 //generate token
2 const tokenResponse= await request.post("/auth",{
3     data:tokenRequestBody
4 })
5
6 const tokenAPIResponseBody= await tokenResponse.json();
7 const tokenNo=tokenAPIResponseBody.token;
8 console.log("token number is " + tokenNo)
9
10 //put api call
11 const putResponse=await request.put("/booking/"+bid, {
12     headers:{
13         "Content-Type":"application/json",
14         "Cookie":`token=${tokenNo}`
15     },
16     data:putRequestBody
17 })
18 const putResponseBody= await putResponse.json();
19 console.log(putResponseBody)
20
21 //validate status code
22 await expect(putResponse.status()).toBe(200);
23 });
```