

# baka-pwapi-10

## DELETE API Request in Playwright

```
1 //
2 //
3 //
4 const { test, expect } = require("@playwright/test");
5 import { faker } from "@faker-js/faker";
6 const { DateTime } = require("luxon");
7 var dynamicPostRequest = require("../test-data/dynamicrequestbody.json");
8 import { stringFormat } from "../utils/commons";
9 const tokenRequestBody=require("../test-data/puttokenrequestbody.json");
10 const putRequestBody=require("../test-data/putrequestbody.json")
11 const patchRequestBody=require("../test-data/patchrequestbody.json")
12
13 //first create data then get data.
14
15 test("create delete api request", async ({
16   request,
17 }) => {
18
19
20   // const dynamicRequestBody=stringFormat(JSON.stringify(dynamicPostRequest),
21   //"testers talk cypress", "testers talk js", "banana")
22
23
24   //call the util function first.
25   var updatedRequestBody = stringFormat(
26     JSON.stringify(dynamicPostRequest),
27     "testers talk cypress",
28     "testers talk javascript",
29     "apple"
30   );
```

```
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");
21
22 // validate api response nested json obj
23 expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
24   "checkin",
25   "2018-01-01"
26 );
27 expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
28   "checkout",
29   "2019-01-01"
30 );
31
32 console.log("====post response====")
33
34
35 console.log("====get response====")
```

```
1 //get api call
2   const getAPIResponse = await request.get(`/booking/${bid}`)
3
4   console.log(await getAPIResponse.json())
5
6   await expect(getAPIResponse.ok()).toBeTruthy();
7   await expect(getAPIResponse.status()).toBe(200);
8
9
10  //generate token
11  const tokenResponse= await request.post("/auth",{
12    data:tokenRequestBody
13  })
14
15  const tokenAPIResponseBody= await tokenResponse.json();
16  const tokenNo=tokenAPIResponseBody.token;
17  console.log("token number is " + tokenNo)
18
19  //patch api call
20  const patchResponse=await request.patch("/booking/"+bid, {
21    headers:{
22      "Content-Type":"application/json",
23      "Cookie":`token=${tokenNo}`
24    },
25    data:patchRequestBody
26  })
27  const patchResponseBody= await patchResponse.json();
28  console.log(patchResponseBody)
```

```
1 //validate status code
2 await expect(patchResponse.status()).toBe(200);
3
4 //delete api request
5 const deleteAPIResponse=await request.delete("/booking/"+bid,{
6   headers:{
7     "Content-Type":"application/json",
8     "Cookie":`token=${tokenNo}`
9   },
10 })
11
12 await expect(deleteAPIResponse.status()).toEqual(201);
13
14 //its simple plain text.
15 await expect(deleteAPIResponse.statusText()).toBe("Created")
16
17
18
19 });
```