

Note-

Inside fixture block only we need to have the visit url whether its api or ui.

```
1 describe("api testing", ()=>{
2
3
4   it("Appraoch1- Hard coded json object", ()=>{
5     const requestBody={
6       tourist_name: "Mike",
7       tourist_email: "mike987645@gmail.com",
8       tourist_location: "Paris"
9     }
10    cy.request(
11      {
12        method: 'POST',
13        url: 'http://restapi.adequateshop.com/api/Tourist' ,
14        body:requestBody
15      }
16    )
17    .then( (response) =>{
18      expect(response.status).to.eq(201)
19      expect(response.body.tourist_name).to.eq("Mike")
20      expect(response.body.tourist_email).to.eq("mike987645@gmail.com")
21      expect(response.body.tourist_location).to.eq("Paris")
22    })
23  })
24
25 })
```

```
1 it("Appraoch2- Dynamically generating json object", ()=>{
2     const requestBody={
3         //math.random returns number between 0 to 1.
4         //to string - we can pass how many chars needed.
5         tourist_name: Math.random().toString(5).substring(2),
6         tourist_email: Math.random().toString(5)
7         .substring(2)+"@gmail.com",
8         tourist_location: "Paris"
9     }
10    cy.request(
11        {
12            method: 'POST',
13            url: 'http://restapi.adequateshop.com/api/Tourist' ,
14            body:requestBody
15        }
16    )
17    .then( (response) =>{
18        expect(response.status).to.eq(201)
19        expect(response.body.tourist_name)
20        .to.eq(requestBody.tourist_name)
21        expect(response.body.tourist_email)
22        .to.eq(requestBody.tourist_email)
23        expect(response.body.tourist_location)
24        .to.eq(requestBody.tourist_location)
25    })
26    })
27
28    })
```

```

1  it.only("Approach3- using Fixture", ()⇒{
2
3      cy.fixture('tourist').then( (data)⇒{
4          const requestBody=data;
5
6          cy.request(
7              {
8                  method: 'POST',
9                  url: 'http://restapi.adequateshop.com/api/Tourist' ,
10                 //this needed when your website is blocked for launch by
11                 //click on this to verify if your human.
12                 failOnStatusCode: false,
13                 body:requestBody
14             })
15
16             .then( (response) ⇒{
17                 expect(response.status).to.eq(201)
18                 expect(response.body.tourist_name)
19                 .to.eq(requestBody.tourist_name)
20                 expect(response.body.tourist_email)
21                 .to.eq(requestBody.tourist_email)
22                 expect(response.body.tourist_location)
23                 .to.eq(requestBody.tourist_location)
24
25                 //validate property name and its value.
26                 //two ways to use it.
27                 expect(response.body).has
28                 .property('tourist_email',requestBody.tourist_email)
29                 expect(response.body).to.
30                 have.property('tourist_email',requestBody.tourist_email)
31
32             })
33
34         })
35     } )
36
37
38 })

```

Test data for fixture-

```

1  {
2    "tourist_name": "John Canady",
3    "tourist_email": "canedy98987fycc@gmail.com",
4    "tourist_location": "USA"
5  }

```

codesnap.dev

With or without fixture extension it works the same way-

```

postrequestwithfixture.cy.js  postrequestwithfixture_1.cy.js X
cypress > e2e > postrequestwithfixture_1.cy.js > describe('api testing') callback > it('use fixture to generate data') callback
1  /// <reference types="cypress" />
2
3  describe('api testing', () => {
4
5    it('use fixture to generate data', () => {
6
7      //lets use fixture without extension.
8      //it works with or without extension.
9      cy.fixture("touristdetails"). then((data)=>{
10         const requestbody=data;
11

```

```

10         const requestbody=data;
11
12         cy.request(
13           {
14             method: 'post',
15             url: 'http://restapi.adequateshop.com/api/Tourist',
16             failOnStatusCode: false,
17             body: requestbody
18           })

```

```
18     })
19     .then((response)=>{
20         expect(response.status).to.equal(201)
21         expect(response.body.tourist_name).to.eq(requestbody.tourist_name)
22         expect(response.body.tourist_email).to.eq(requestbody.tourist_email)
23         expect(response.body.tourist_location).to.eq(requestbody.tourist_location)
24     })
```

```
24
25     //validate property name and its value.
26     //another two ways to check.
27     expect(response.body).has.property("tourist_email", requestbody.tourist_email)
28     expect(response.body).to.have.property("tourist_email", requestbody.tourist_email)
29
30 })
31
32 }
```