

User API Documentation

1. Database table description

Value	Type	Required	Description
id	int	Yes(auto-increment)	User id
name	String	Yes	User name
age	int	Yes	User age

2. API description

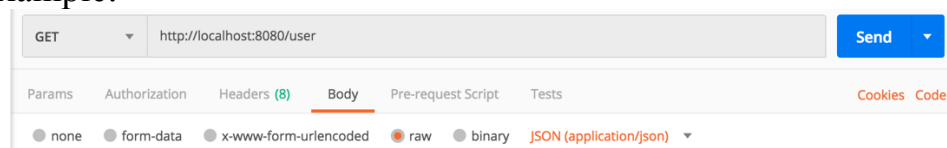
2.1 Common API prefix

test environment: localhost:8080/user

2.2 Modules

(1) Retrieve all users

- Address: localhost:8080/user
- Request method: get
- Parameter: none
- Data return: json
- Example:



GET http://localhost:8080/user Send

Params Authorization Headers (8) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
3  {
4    "id": 2,
5    "name": "Zhang",
6    "age": 30
7  },
8  {
9    "id": 3,
10   "name": "Wang",
11   "age": 30
12 },
13 {
14   "id": 4,
15   "name": "Zhao",
16   "age": 18
17 },
18 {
19   "id": 5,
20   "name": "Li",
21   "age": 50
22 },
23 {
24   "id": 6,
25   "name": "Zhao",
26   "age": 35
27 },
28 {
29   "id": 7,
30   "name": "Jiang",
31   "age": 55
32 }
```

(2) Retrieve user by id

- Address: localhost:8080/user/id
- Request method: get
- Parameter: id
- Data return: json
- Example:

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/user/3`. The response is a JSON object: `{ "id": 3, "name": "Wang", "age": 30 }`. The status is 200 OK, time is 103 ms, and size is 161 B.

GET `http://localhost:8080/user/3` Send

1

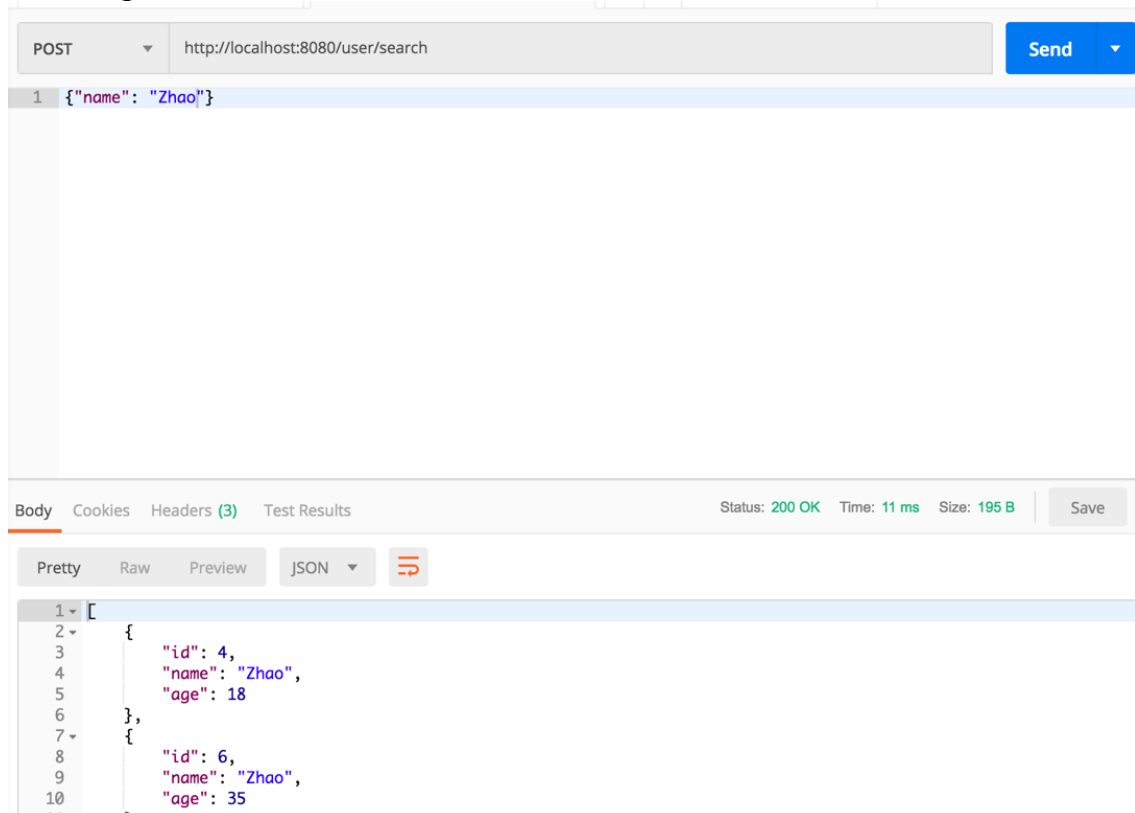
Body Cookies Headers (3) Test Results Status: 200 OK Time: 103 ms Size: 161 B Save

Pretty Raw Preview JSON ↺

```
1 {
2   "id": 3,
3   "name": "Wang",
4   "age": 30
5 }
```

(3) Retrieve all users with same name

- Address: localhost:8080/user
- Request method: get
- Parameter:
`{"name": ""}`
- Data return: json
- Example:



(4) Create new user

- Address: localhost:8080/user
- Request method: post
- Parameter:
{"name": "", "age": }
- Data return: json
- Example:

The screenshot shows a REST client interface with the following details:

- Request:**
 - Method: POST
 - URL: http://localhost:8080/user/
 - Content-Type: JSON (application/json)
 - Body: {"name": "Li", "age": 45}
- Response:**
 - Status: 200 OK
 - Time: 156 ms
 - Size: 159 B
 - Body: {"id": 8, "name": "Li", "age": 45}

(5) Update user

- Address: localhost:8080/user/id
- Request method: put
- Parameter:
{“name”: “”, “age”: }
- Data return: json
- Example:

The screenshot displays a REST client interface with the following details:

- Request Method:** PUT
- URL:** http://localhost:8080/user/2
- Body:** `{“name”: “”, “age”: }`
- Response Status:** 200 OK
- Response Time:** 41 ms
- Response Size:** 159 B
- Response Body (JSON):**

```
{
  "id": 2,
  "name": "Li",
  "age": 33
}
```

(6) Delete user

- Address: localhost:8080/user/id
- Request method: delete-
- Parameter: id
- Data return: boolean
- Example:

The screenshot displays a REST client interface with two panels. The top panel shows a DELETE request to `http://localhost:8080/user/2` with a status of 200 OK, time of 42 ms, and size of 134 B. The response body is `true`. The bottom panel shows a GET request to `http://localhost:8080/user` with a status of 200 OK, time of 18 ms, and size of 320 B. The response body is a JSON array of two user objects.

```
DELETE http://localhost:8080/user/2
```

Status: 200 OK Time: 42 ms Size: 134 B

```
true
```

```
GET http://localhost:8080/user
```

Status: 200 OK Time: 18 ms Size: 320 B

```
[{"id": 3, "name": "Wang", "age": 30}, {"id": 4, "name": "Zhao", "age": 18}]
```

(Results show that the user with id = 2 has been deleted)