Fake Services Inner Dev Loop

Introduction

Consumers of remote services often find that their development cycle is not in sync with development of remote services, leaving developers of these consumers waiting for the remote services to "catch up". One approach to mitigate this issue and improve the inner dev loop is by decoupling and using Mock Services. Various Mock Service options are detailed here.

This document will focus on providing an example using the Fake Services approach.

API

For our example API, we will work against a `/User` endpoint and the properties for `User` will be:

1. id - int
2. username - string
3. firstName - string
4. lastName - string
5. email - string
6. password - string
7. phone - string

8. userStatus - int

Tooling

For the Fake Service approach, we will be using Json-Server. Json-Server is a tool that provides the ability to fully fake REST APIs and run the server locally. It is designed to spin up REST APIs with CRUD functionality with minimal setup. Json-Server requires NodeJS and is installed via NPM.

```
npm install -g json-server
```

Setup

In order to run Json-Server, it simply requires a source for data and will infer routes, etc. based on the data file. Note that additional customization can be performed for more advanced scenarios (e.g. custom routes). Details can be found here.

For our example, we will use the following data file, `db.json`:

```
{
  "user": [
    {
      "id": 0,
      "username": "user1",
      "firstName": "Kobe",
      "lastName": "Bryant",
      "email": "kobe@example.com",
      "password": "superSecure1",
      "phone": "(123) 123-1234",
      "userStatus": 0
    },
    {
```

```
      "id": 1,
      "username": "user2",
      "firstName": "Shaquille",
      "lastName": "O'Neal",
      "email": "shaq@example.com",
      "password": "superSecure2",
      "phone": "(123) 123-1235",
      "userStatus": 0
    }
  ]
}
```

Run

Running Json-Server can be performed by simply running:

```
json-server --watch src/db.json
```

Once running, the User endpoint can be hit on the default localhost port: `http:/localhost:3000/user`

Note that Json-Server can be configured to use other ports using the following syntax:

```
json-server --watch db.json --port 3004
```

Endpoint

The endpoint can be tested by running curl against it and we can narrow down which user object to get back with the following command:

```
curl http://localhost:3000/user/1
```

which, as expected, returns:


```
{
  "id": 1,
  "username": "user2",
  "firstName": "Shaquille",
  "lastName": "O'Neal",
  "email": "shaq@example.com",
  "password": "superSecure2",
  "phone": "(123) 123-1235",
  "userStatus": 0
}
```

Last update: August 26, 2024