
Take control of your mock !
Don’t worry, I have a backup plan !

Johnathan MEUNIER

Lead front @AXA France

@MonsieurNohj

John.meunier

Table of contents

01
Introduction

My demo is broken !


gif-finder.com

My demo seems broken

How can I prevent that ?

Internet issue

Work without Internet, in a train, when my company proxy is not available

API issue

My API is down or has changed

Prepare scenarios

I don’t want to fill all my forms everytime or I want to improve the resilience of my app

02

Mocks

What is it ? What’s now ? Demo

What is a mock ?

- Simulation of what would happen in a normal state
- Not request directly the API
- Simplify response
- Have to be static (most of the time)

Pros & Cons

Pros

- Work offline
- Simulate your local dev with different scenarios
- Stable environment, whatever the state of your service
- A mock is almost instantaneous …

Cons

- … but the real behavior of your application can be distorded
- Your local data have to be up to date permanently with the real API (path, values, schema, etc)

Current solution

- Package like json-server can create a simple rest API which will serve static json file (or complex in some cases)
- Powerful, this solution will redirect all the calls to a local webserver …

- … but you need to launch a specific webserver in addition to your application
- in your application, you will have specific code to manage the switch between this local server and the real API

Current solution

- Simplier, you can create static json file with your component and use this file.
- Useful to initiate a development but impossible to use this code in production
- Really far from the real behavior of your application

Demo of my working application

03

Mock Service Worker

Setup, usage, tests, msw-ui,
scenarios, advanced usage

MSW is a service worker

- A proxy between a web app and your browser
- Sandboxed
- Lower level API

Why MSW is the best solution ?

- MSW will intercept some requests of the network layer of your browser
- Totally transparent during the development phase
- Nothing to do in the functionnal code, in your compoments !
- Every is centralized in a specific part of your codebase

Front setup

Request handler

```
import { rest } from 'msw';

rest.[get|post|put|•••](url,callback)
```

- get
- post
- put
- patch
- delete
- options

- **Exact** : https://api.backend.dev/users
- **Wildcard** : users/*
- **Parameters** : `https://api.backend.dev/users/:userId`
- **RegExp** : /\/users\//

Request resolver

When the request is caught, you have to return a response. The response is the return provided by the callback of the request handler.

Parameter name	Description
req	Information about the request
res	Function that create the mock
ctx	Transformation function

```
res(ctx.status(200), ctx.json({"id": 42}))
```

Request resolver

A complete but simple mock looks like this :

```
rest.get("https://my.api", (req, res, ctx) ⇒
  res(
    ctx.status(200),
    ctx.json({res : "hello"})
  )
)
```

Request resolver

Add delay (always add delay !)

```
res(ctx.status(200), ctx.delay(300), ctx.json({"id": 42}))
```

Change status

```
res(ctx.status(500), ctx.json({ "error":  "an error message"}))
```

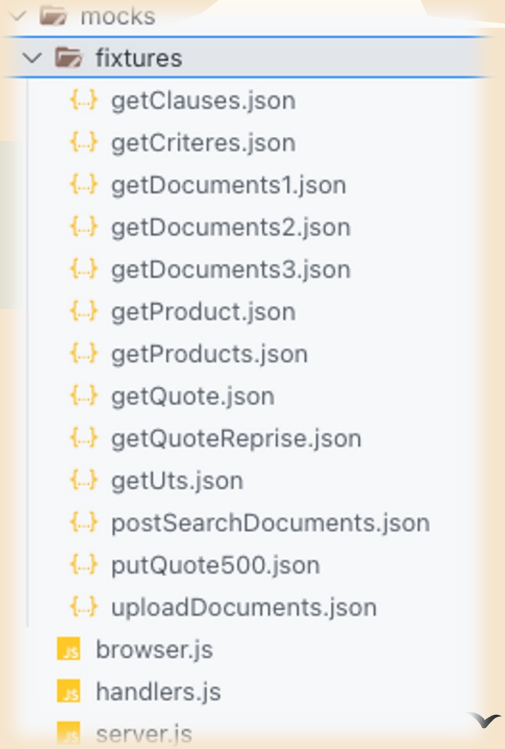
Provide headers

```
ctx.set({
        'correlation-id': '4122-5c6d-000023c1681c',
        'feature': 'name=MY_PRODUCT',
})
```

Request resolver

Organization

```
src/mocks/fixtures
[verbHTTP][ressource][metaInfo].json
```



Request resolver

Complex response

```
1 rest.post("/documents/:documentId", (req, res, ctx) ⇒ {
2     const document = documents[req.params.documentId]
3     document.list.sort(() ⇒ Math.random() - 0.5);
4     return res(ctx.status(200), ctx.json(getDocuments));
5 }),
6
```

Demo : fetch, simple mock, delay, status, headers, complex response

Test

Avoid mocking directly fetch, axios or your api consumption hook because you will create a dependance.

Mock your call throught MSW, regardless of how you interrogate it.

Specific mock in a test

Simple solution

```
// monTest.steps.js
import { server } from 'mocks/server'
//  ...
server.use(
  rest.put(`users/:userId`, (req, res, ctx) => res(ctx.status(200))),
  rest.post(`${conf.apiEbusinessUrl}/parameters`, (req, res, ctx) =>
    res(ctx.status(200), ctx.json(getParametersMachineDeductibleAmountList))
  )
)
//  ...

```

Specific mock in a test

With a gherkin feature file

```
given('Je charge un devis en reprise', async () => {
  server.use(
    rest.get('/insurance-quotes/:quoteId', (req, res, ctx) =>
      res(ctx.status(200), ctx.json(getQuoteReprise))
    )
  )
})

```

Specific mock in a test

Variable throught .feature

```
Feature: Limite inférieure à la franchise

    Scenario Outline: Gestion de la validation du champ franchise
    Given Un devis rempli et une offre commerciale complète sont chargés avec une franchise limitée à "<minValue>"
    And J'accède à la page Tarif en tant que Siège
    When Je renseigne pour le champ "Franchise" la valeur "<franchiseValue>"
    Then Le message d'erreur "<errorMessage>" lié à la valeur de la franchise "<status>" affichée
    Examples:
      | minValue | franchiseValue | status | errorMessage |
      | 500 | 400 | est | Montant mini: 500€ |
      | 600 | 700 | n'est pas | Montant mini: 600€ |
      | 500 | 1000.6 | est | La valeur doit être un nombre entier |
      | 600 | 700 | n'est pas | La valeur doit être un nombre entier |
      | 600 | 800.6 | est | La valeur doit être un nombre entier |

```

Specific mock in a test

```
1 const givenDevisWithFranchiseLimit = (given) => {
2   given(
3     /^Un devis rempli et une offre commerciale complète sont chargés avec une franchise limitée à
  "(.*)"$/,
4     (deductibleMinValue) => {
5       const offersCopy = cloneDeep(getOffers)
6       set(offersCopy, deductibleMinValuePath, deductibleMinValue)
7       server.use(
8         rest.post('/insurancequotes/:quoteId', (req, res, ctx) =>
9           res(ctx.status(200), ctx.json(getOffers))
10        ),
11        rest.get(
12          `${conf.apiEbusinessUrl}/insurance-quotes/:quoteId`,
13          (req, res, ctx) => res(ctx.status(200), ctx.json(getQuote))
14        )})}
15
```

Demo MSW & Gherkin

MSW-UI

- MSW-UI works on top of MSW
- Programmatically Change wich mocks are used
  - Debug tool
  - Create scenarios

Demo MSW-UI

Scenarios

- Especially with MSW-UI, you can create complete scenarios that match your needs.
- For examples :
  - Start from scratch a new project
  - Modify a finished project
  - Resume a current project
  - Simulate differents cases of errors

How I use a scenarios pattern in my project

How far should we go in the mocks ?

- In most of cases, serving static files is enough
- But sometimes, the service enriches the data sent
- Don’t redevelop the service in your handler, keep it as simple as possible. *If you have this data in your request, so add this data in your answer.*

Typescript

- Typescript can help you.
- Your mocked data have the same model as your web service

Develop the consumer before the service

- In some cases, in can be very useful to develop the service consumer before the service :
  - You have a back for front that transform your data
  - The back-end team is late on the project
  - …
- Your contracts between your services and your consumers can’t change

Cohabitation with authentication

- Only one service worker per scope is allowed
- Sometimes, your authentification will work with a service worker (hello auth-worker and the latest version of react-oidc)
- During the local dev, use the MSW and mock the auth
- When you deploy, use the auth and disable mock

04

Conclusion

Ressources, annexes, thanks


Typescript
MSW-UI
Delay
Request resolver
Service worker
Request handler
MSW
Enrichments
Scenarios
In advance
Test
Fixtures
Resilience

Thanks

Do you have any questions?

https://bit.ly/mock-msw

Johnathan MEUNIER
@MonsieurNohj

Resources

- Slide template from Slidesgo
- Repo github : johnmeunier/dojo-msw (github.com)
- MSW : MSW – Seamless API mocking library for browser and Node | Mock Service Worker (mswjs.io)
- MSW-UI : fvanwijk/msw-ui: UI for Mock Service Worker to set mock scenarios on runtime (github.com)
- AXA France GuilDev on Github (auth-worker, react-oidc, etc)