จำลองระบบเพื่อใช้ในการทดสอบ ด้วย

Mountebank 101

Testing Microservices
with
Mountebank

Brandon Byars

MANNING
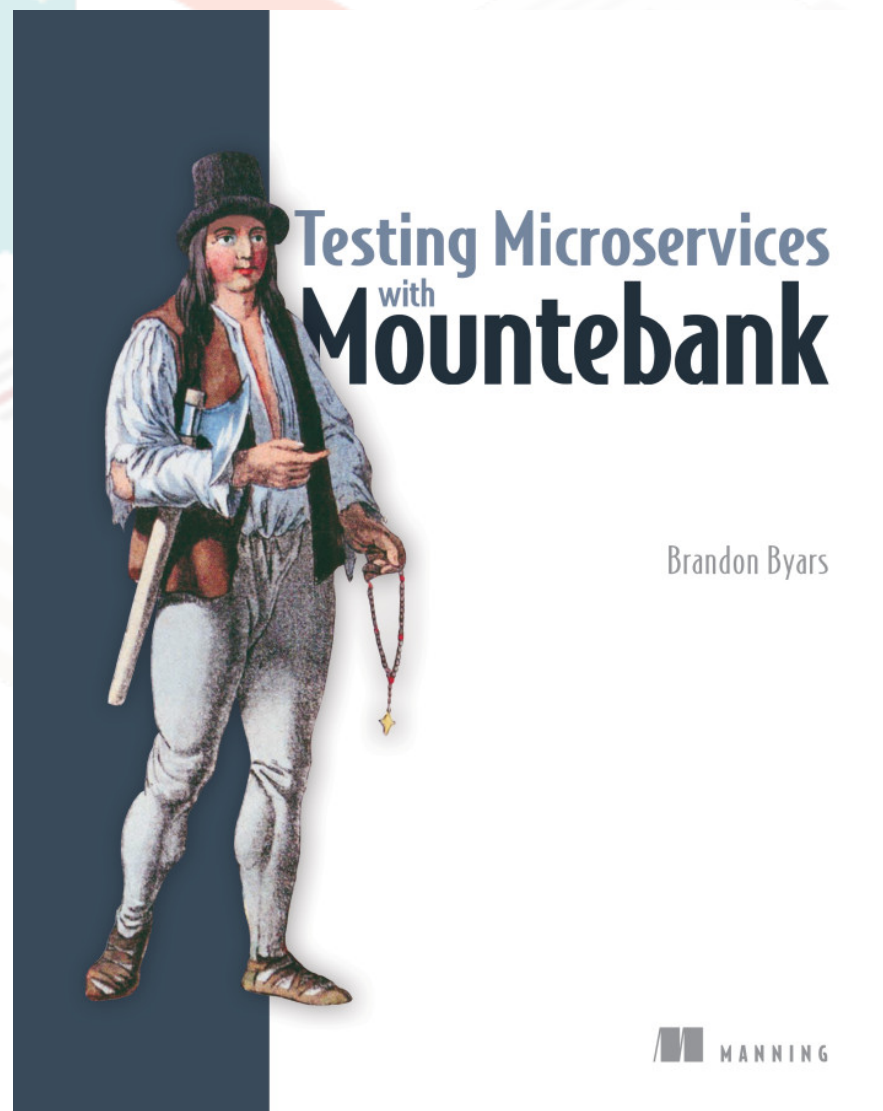
- Hello, World! Mountebank

- Predicates

- Behavior and Programming mountebank

- Adding Behaviors

- Record/Replay

Day3: Adding Behaviors

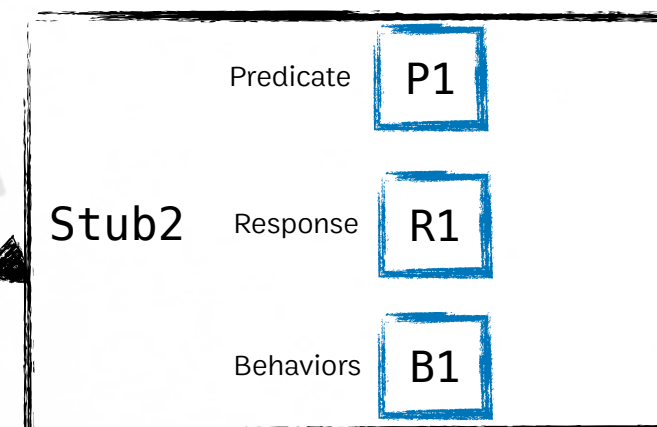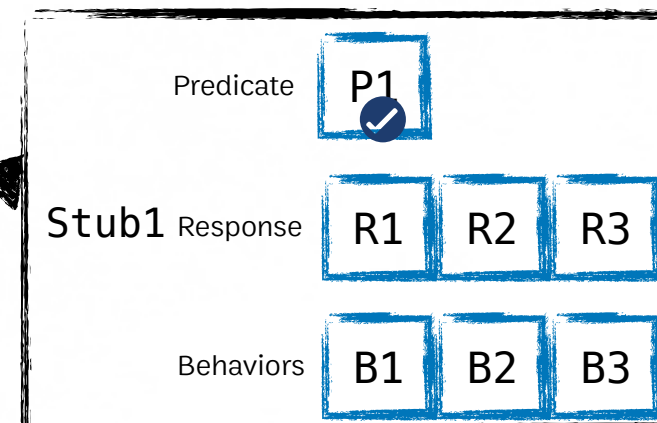# Understand behaviors

- **Predicates** help route requests on the way in.
- **Responses** generate the responses on the way out.
- *Behaviors* post-process the responses before shipping them over the wire

```
{  "protocol": "http",
   "port": 3000,
   "stubs": [
     {
       "predicates": [],
       "responses": [
         {
           "is": {},
           "_behaviors": {
             "decorate": {},
             "wait": {},
             "copy": []
           }
         }
       ]
     }
   ]
}
```

Imposter

Stub1
Predicate P1 ✓
Response R1 R2 R3
Behaviors B1 B2 B3

Stub2
Predicate P1
Response R1
Behaviors B1

from: Testing Microservice with Mountebank page 131

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Decorating a response: allows you to post-process the response.

Using the decorate function

An *inject* response

```
{
  "responses": [
    {
      "inject": "function (request, state, logger) {}"
    }
  ]
}
```

A *decorate* behavior

```
{
  "responses": [
    {
      "is": {
        "body": {}
      },
      "_behaviors": {
        "decorate": "function (request, response, logger) {}"
      }
    }
  ]
}
```

Plugging the decoration into the behaviours

Imposter: decorate.json

```json
{
  "protocol": "http",
  "port": 3000,
  "stubs": [
    {
      "responses": [
        {
          "is": {
            "statusCode": 201,
            "headers": { "Content-Type": "application/json" },
            "body": {}
          },
          "_behaviors": {
            "decorate": "function (request, response, logger) { const item =
JSON.parse(request.body); response.body.message = item.name }"
          }
        }
      ]
    }
  ]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000

start Mountebank: unix/windows

mb start --allowInjection --configfile decorate.json

Quiz Behaviors

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items

HEADER

"Content-Type": "application/json"

Body

```
{
  "name": "riderX",
   "price": 100

}
```

expected value

```
HTTP/1.1 201 OK
Date: Sun, 05 Apr 2020 10:10:10 GMT
Content-Type: application/json

{
  "message": "RiderX is created",
  "timestamp": "Thu Apr 09 2020 00:21:32 GMT+0700 (Indochina Time)"
}
```

Adding latency to a Response

Using a wait behavior to add latency

Imposter: sleep.json

```json
{
  "protocol": "http",
  "port": 3000,
  "stubs": [
    {
      "responses": [
        {
          "is": {
            "statusCode": 201,
            "headers": { "Content-Type": "application/json" },
            "body": { "name": "sleep" }
          },
          "_behaviors": {
            "wait": 3000
          }
        }
      ]
    }
  ]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000

start Mountebank: unix/windows

mb start --allowInjection --configfile sleep.json

Repeating a Response Multiple Times

Using a repeat behavior to return an error after a small number of successes

Imposter: repeating_a_response.json

Postman: request with

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
    "predicates": [
     { "matches": { "path": "/items/1" } }
    ],
    "responses": [
     {
       "is": {
         "body": { "name": "43 Piece Dinner Set", "price": 12.95 }
       },
       "_behaviors": { "repeat": 3 }
     },
     {
       "is": {
         "body": { "name": "RiderX", "price": 2.95}
       }
     },
     {
       "is": {
         "body": { "name": "Alpha Bot", "price": 33.95 }
       },
       "_behaviors": { "repeat": 5 }
     }
    ]
  }]
}
```

HTTP Method

GET

URL

http://localhost:3000

start Mountebank: unix/windows

mb start --allowInjection --configfile repeating_a_response.json

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Replacing Content in The Response

Replacing Content in The Response

You can always add dynamic data to a response through an inject response, or through the decorate and shellTransform behaviors. But two additional behaviors support inserting certain types of dynamic data into the response **without the overhead of programmatic control**.

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Using a copy behavior to insert the ID from the URL into the response body

Imposter: replacing_content_in_the_response.json

```json
{
  "protocol": "http",
  "port": 3000,
  "stubs": [
    {
      "responses": [
        {
          "is": {
            "body": {
              "id": "$ID",
              "name": "43 Piece Dinner Set",
              "price": 12.95
            }
          },
          "_behaviors": {
            "copy": [{
              "from": "path",
              "into": "$ID",
              "using": {
                "method": "regex",
                "selector": "\\d+$"
              }
            }]
          }
        }
      ]
    }
  ]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

mb start --allowInjection --configfile replacing_content_in_the_response.json

- **\d** A digit, 0–9 (you have to double-escape the backslash in JSON)
- **\w** A word character
- **+** One or more times
- **$** The end of the string

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Copying Request Data to the Response

- The **copy** behavior accepts an array, which means you can make multiple replacements in the response.

- Each replacement should use a different token, and each one can select from a different part of the request.

- You never specify where the token is in the response. That's by design. You could have put the token in the headers or even the statusCode, and mountebank would replace it.

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Using a Group Match

Imposter: using_a_grouped_match.json

```json
{
  "protocol": "http",
  "port": 3000,
  "stubs": [
    {
      "responses": [
        {
          "is": {
            "body": {
              "id": "$ID",
              "name": "43 Piece Dinner Set",
              "price": 12.95
            }
          }
        },
        "_behaviors": {
          "copy": [{
            "from": "path",
            "into": "$ID",
            "using": {
              "method": "regex",
              "selector": "items/(\\w+)"
            }
          }]
        }
      ]
    }
  ]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

mb start --allowInjection --configfile using_a_grouped_match.json

regex: items/(\\w+)

string: items/123

result: ['items/123', '123']

Quiz Behaviors

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items

HEADER

"Content-Type": "application/json"

Body

```
{
  "name": "riderX",
   "price": 100

}
```

expected value

```
HTTP/1.1 201 OK
Date: Sun, 05 Apr 2020 10:10:10 GMT
Content-Type: application/json


{
  "message": "RiderX is created",
  "timestamp": "Thu Apr 09 2020 00:21:32 GMT+0700 (Indochina Time)"
}
```

Looking Up Data from an External Data Source

Imposter: duplicate_stubs.json

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [
    {
      "predicates": [{
        "equals": { "path": "/items/1" }
      }],
      "responses": [{
        "is": {
          "statusCode": 200,
          "headers": { "Content-Type": "application/json" },
          "body": { "id": "1", "name": "43 Piece Dinner Set", "price": 12.95
          }
        }
      }]
    },
    {
      "predicates": [{
        "equals": { "path": "/items/2" }
      }],
      "responses": [{
        "is": {
          "statusCode": 200,
          "headers": { "Content-Type": "application/json" },
          "body": { "id": "2", "name": "RiderX", "price": 2.95
          }
        }
      }]
    },
    {
      "predicates": [{
        "equals": { "path": "/items/3" }
      }],
      "responses": [{
        "is": {
          "statusCode": 200,
          "headers": { "Content-Type": "application/json" },
          "body": { "id": "3", "name": "Alpha Bot", "price": 33.95
          }
        }
      }]
    }
  ]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

mb start --allowInjection --configfile duplicate_stubs.json

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Looking Up Data from an External Data Source

Imposter: lookup.json

```json
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
    "responses": [{
      "is": {
        "statusCode": 200,
        "headers": { "Content-Type": "application/json" },
        "body": {
          "id": "${row}['id']",
          "name": "${row}['name']",
          "price": "${row}[price]"
        }
      },
      "_behaviors": {
        "lookup": [
          {
            "key": {
              "from": "path",
              "using": { "method": "regex", "selector": "items/(\\w+)" },
              "index": 1
            },
            "fromDataSource": {
              "csv": { "path": "toys.csv", "keyColumn": "id" }
            },
            "into": "${row}"
          }
        ]
      }
    }]
  }]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

mb start --allowInjection --configfile lookup.json

Data toys.csv

```
"id","name","price"
"1","43 Piece Dinner Set","12.95"
"2","RiderX","2.95"
"3","Alpha Bot","33.95"
```

บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Quiz Looking Up Data

request value

> **POST /items** HTTP/1.1
> HOST localhost:3000
> Content-Type: application/json
>
> { "reqId": 12345, "name": "43 Piece Dinner Set is created", "price" 12.95 }

expected value

> HTTP/1.1 **201** OK
> Date: Sun, 05 Apr 2020 10:10:10 GMT
> Content-Type: application/json
>
> { "message": "43 Piece Dinner Set is created", "reqId": 12345 }

request value

> **POST /items** HTTP/1.1
> HOST localhost:3000
> Content-Type: application/json
>
> { "reqId": 78901, "name": "RiderX", "price" 2.95 }

expected value

> HTTP/1.1 **400** OK
> Date: Sun, 05 Apr 2020 10:10:10 GMT
> Content-Type: application/json
>
> { "message": "RiderX is already exists", "reqId": 78901 }

request value

> **POST /items** HTTP/1.1
> HOST localhost:3000
> Content-Type: application/json
>
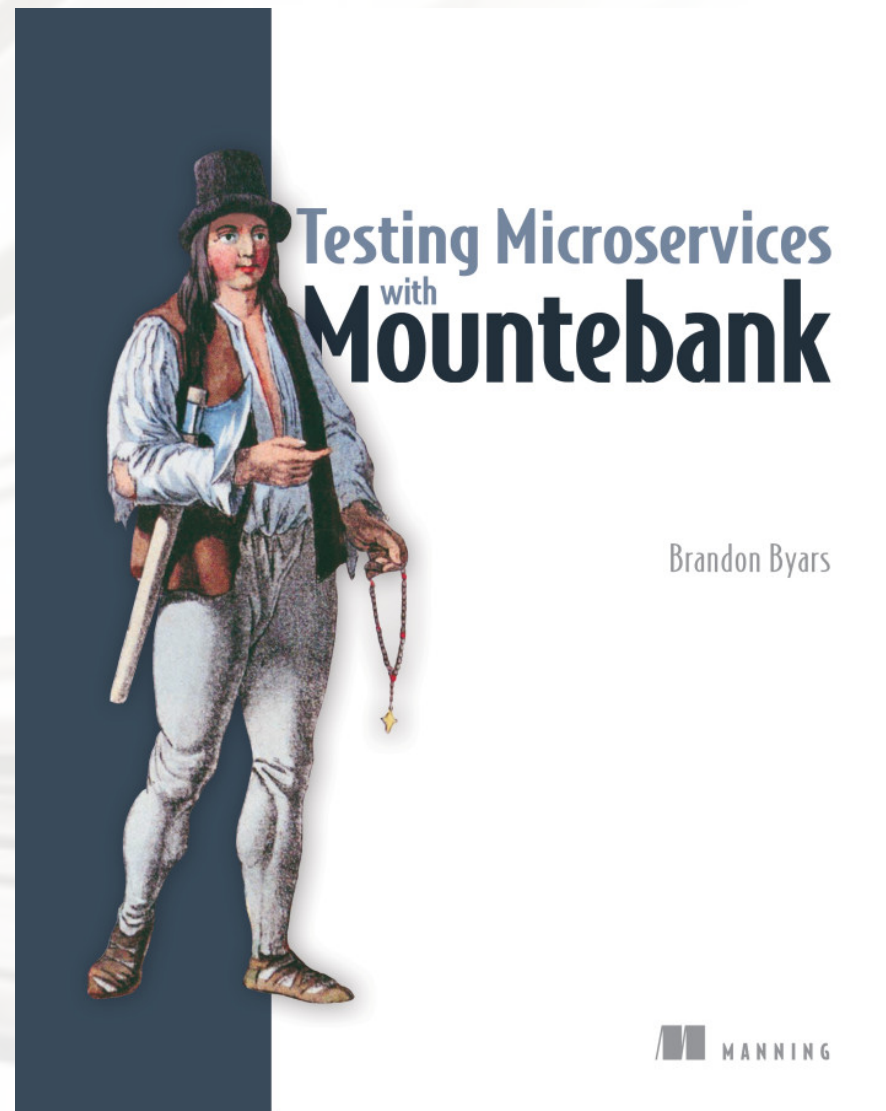> { "reqId": 90328, "name": "Alph@ Bot", "price" 33.95 }

expected value

> HTTP/1.1 **400** OK
> Date: Sun, 05 Apr 2020 10:10:10 GMT
> Content-Type: application/json
>
> { "message": "Alph@ Bot has invalid characters", "reqId": 90328 }

Books to Read and Practice

https://www.manning.com/books/testing-microservices-with-mountebank