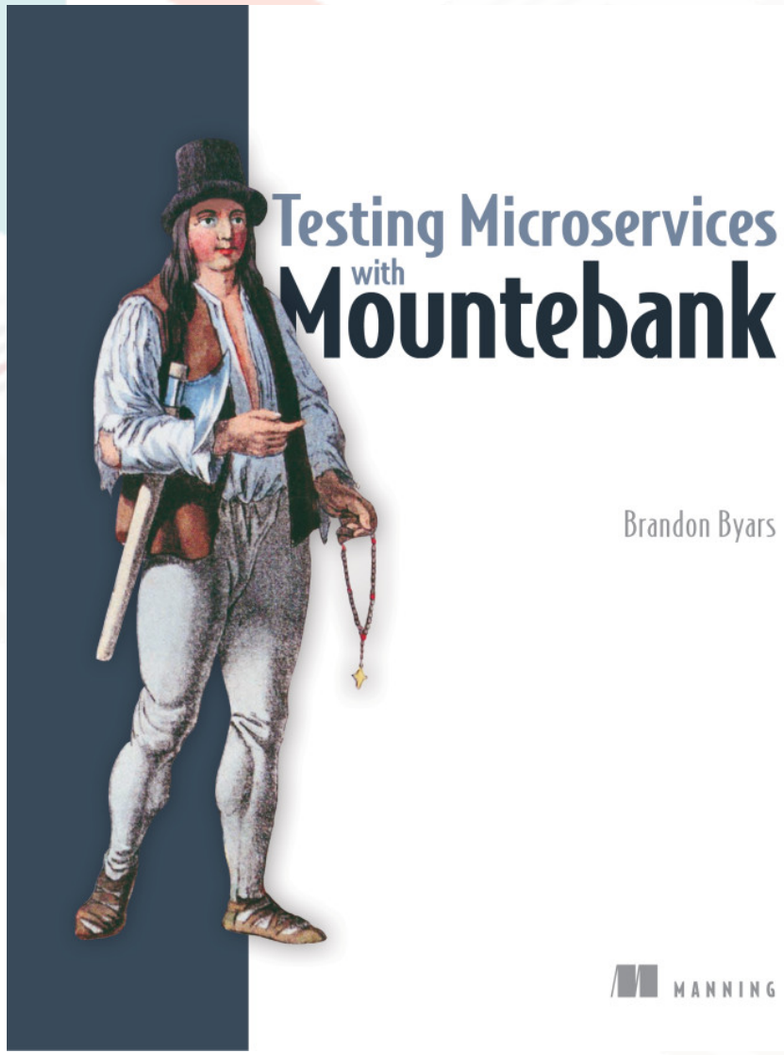
จำลองระบบเพื่อใช้ในการทดสอบ ด้วย

Mountebank 101



- Hello, World! Mountebank
- Predicates
- Behavior and Programming mountebank
- Adding Behaviors
- Record/Replay

Day3: Behavior and Programming mountebank

- Matching requests even when none of the standard predicates do the trick
- Adding dynamic data to mountebank responses
- Debugging your scripts within mountebank

Matching Requests Even When None of The Standard Predicates Do The Trick

Creating your own predicate

You can create your own predicate using JavaScript and the inject predicate, but first you have to start mountebank with the --allowInjection command line flag.

```
mb start --configfile matches-predicate.json --allowInjection
```

The safest option is to add the --localOnly flag, which dis-allows remote connections.

```
mb start --configfile matches-predicate.json --allowInjection --localOnly
```

Plugging the function into the predicates

Imposter: plugging-the-function-predicate.json

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
     "predicates": [{
          "inject": { "function (request) {\n if
(request.path.indexOf('/items/1') === 0) {\n return true;\n }\n
else {\n return false;\n}\n}
" }
        }
     ],
     "responses": [{
        "is": {
          "statusCode": 200,
          "headers": { "Content-Type": "text/plain" },
          "body": {
             "name": "43 Piece Dinner Set",
             "price": 12.95,
             "quantity": 10
           }
        }
     }]
  }]
}
```

Plugging the function into the predicates of stub involves JSON-escaping the function, which replaces newlines with '\n', and escaping double quotes

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile plugging-the-function-predicate.json --allowInjection
```

The Structure of an Inject Predicate

Imposter: filename**.ejs**

```
{
    "predicates": [{
        "inject": "<%- stringify(filename, 'inject-predicate-file') %>"
      }
    ],
    "responses": [{
      "is": {
        "statusCode": 200,
        "headers": { "Content-Type": "text/plain" },
        "body": {
          "items": [
            {
              "name": "TA Robot",
              "price": 109.99,
              "quantity": 50
            },
            {
              "name": "AlphaBot2",
              "price": 71.0,
              "quantity": 73
            }
          ]
        }
      }
    }]
}
```

Inject predicate: filename.js

```
function (request) {
    if (...) {
        return true;
    } else {
        return false;
    }
}
```

Recommend to use **EJS templating** and the **stringify** function mountebank adds to EJS.

Inject Predicate with stringify

Imposter: inject-predicate.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
     "predicates": [{
          "inject": "<%- stringify(filename, 'inject-predicate.js') %>"
        }
     ],
     "responses": [{
        "is": {
          "statusCode": 200,
          "headers": { "Content-Type": "text/plain" },
          "body": {
             "name": "43 Piece Dinner Set",
             "price": 12.95,
             "quantity": 10
           }
        }
     }]
  }]
}
```

Inject predicate: inject-predicate.js

```
function (request) {
    if (request.path.indexOf('/items/1') === 0) {
        return true;
    } else {
        return false;
    }
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile inject-predicate.json --allowInjection
```

Quiz Inject Predicate

Postman: request with

HTTP Method

POST

URL

http://localhost:3000/items

HEADER

"Content-Type": "application/json"

Body

```
{
   "name": "riderX",
   "price": 100,
   "quantity": 50
}
```

expected value

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

Body

```
{
   "name": "alpha bot",
   "price": 200,
   "quantity": 30
}
```

expected value

```
HTTP/1.1 401 OK
Date: Sun, 05 Apr 2020 10:10:10 GMT
Content-Type: application/json
```

Adding Dynamic Data to Mountebank Responses

Plugging The Function into The Response

Imposter: plugging-the-function-response.json

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
           "inject": "function (request) { if (request.path.indexOf('/items/1') === 0) { return { statusCode: 404 }; } return {}; }"
    }]
  }]
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

You also can create your own response in mountebank. It’s responsible for returning a response object that mountebank will merge with the default response. Think of it as creating an is response using a JavaScript function, as follows.

start Mountebank: unix/windows

```
mb start --configfile plugging-the-function-response.json --allowInjection
```

Inject Response with stringify

Imposter: inject-response.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
          "inject": "<%- stringify(filename, 'inject-
response.js') %>"
    }]
  }]
}
```

Inject response: inject-response.js

```
function (request) {
  if (request.path.indexOf('/items/1') === 0) {
    return { statusCode: 404 };
  }
  return {};
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile inject-response.ejs --allowInjection
```

JSON.stringify()

Imposter: inject-response-messages.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
          "responses": [{
             "inject": "<%- stringify(filename, 'inject-response-messages.js') %>"
    }]
  }]
}
```

Inject response: inject-response-messages.js

```
function (request) {
  if (request.path.indexOf('/items/1') === 0) {
    return { statusCode: 404,
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({'message': 'resource not found'})
    };
  }
  return {};
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile inject-response-messages.ejs --allowInjection
```

JSON.parse()

Imposter: inject-response-parse.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
           "inject": "<%- stringify(filename, 'inject-response-parse.js') %>"
    }]
  }]
}
```

Inject response: inject-response-parse.js

```
function (request) {
  const item = JSON.parse(request.body);
    if (item.id === 1) {
      return {
        statusCode: 404,
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({'message': 'resource not found'})
      };
    }

  return {};
}
```

Postman: request with

HTTP Method

POST

URL

http://localhost:3000/items

Body

```
{
   "id": 1
}
```

start Mountebank: unix/windows

```
mb start --configfile inject-response-parse.ejs --allowInjection
```

Quiz Inject Response

Postman: request with

HTTP Method

POST

URL

http://localhost:3000/items

HEADER

"Content-Type": "application/json"

Body

```
{
   "name": "riderX",
   "price": 100,
   "quantity": 50
}
```

expected value

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

{ "message": "item is created" }
```

Body

```
{
   "name": "alpha bot",
   "price": 200,
   "quantity": 30
}
```

expected value

```
HTTP/1.1 401 OK
Date: Sun, 05 Apr 2020 10:10:10 GMT
Content-Type: application/json

{ "message": "item is not allowed" }
```

Adding State

Imposter: inject-response-state.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
            "inject": "<%- stringify(filename, 'inject-response-state.js') %>"
    }]
  }]
}
```

Inject response: inject-response-state.js

```
function (request, state) {
  if (typeof state.requests === 'undefined') {
      state.requests = {};
    }
    const item = JSON.parse(request.body);
    if (typeof state.requests[item.id] === 'undefined') {
      state.requests[item.id] = 0;
    }
    state.requests[item.id] += 1;

    return {
      statusCode: 200,
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({'count': state.requests[item.id]})
    };
}
```

Postman: request with

HTTP Method

POST

http://localhost:3000/items

Body

```
{
   "id": 1
}
```

start Mountebank: unix/windows

```
mb start --configfile inject-response-state.ejs --allowInjection
```

Debugging Your Scripts Within Mountebank

console.log()

Imposter: inject-response-console.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
            "inject": "<%- stringify(filename, 'inject-response-console.js') %>"
    }]
  }]
}
```

Inject response: inject-response-console.js

```
function (request) {
  console.log(request.path.indexOf('/items/1') );
  if (request.path.indexOf('/items/1') === 0) {
    return { statusCode: 404 };
  }
  return {};
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile inject-response-console.ejs --allowInjection
```

logger

Imposter: inject-response-logger.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
            "inject": "<%- stringify(filename, 'inject-response-logger.js') %>"
    }]
  }]
}
```

A logger object with `debug`, `info`, `warn`, and `error` functions to write to the mountebank logs.

Inject response: inject-response-logger.js

```
function (request, state, logger) {
  logger.info(request.path.indexOf('/items/1') );
  if (request.path.indexOf('/items/1') === 0) {
    return { statusCode: 404 };
  }
  return {};
}
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

start Mountebank: unix/windows

```
mb start --configfile inject-response-logger.ejs --allowInjection --loglevel info
```

Quiz

Imposter: inject-**???**.ejs

```
{
  "protocol": "http",
  "port": 3000,
  "stubs": [{
        "responses": [{
            "inject": "<%- stringify(filename, '???.js') %>"
    }]
  }]
}
```

Inject response: inject-**???**.js

```
function (request, state, logger) {
  ???????
}
```

เงื่อนไข

ซื้อได้ไม่เกิน 3 ครั้งต่อคน
- ซื้อได้ statusCode 200
- ซื้อไม่ได้ statusCode 400

ซื้อได้ไม่เกิน 1000 บาทต่อคน
- ซื้อได้ statusCode 200
- ซื้อไม่ได้ statusCode 400

Postman: request with

HTTP Method

POST

URL

http://localhost:3000/items

HEADER

```
"Content-Type": "application/json"
```

Body

```
{
  "user": 101,
  "name": "riderX",
  "price": 100,
  "quantity": 50
}
```

Summary

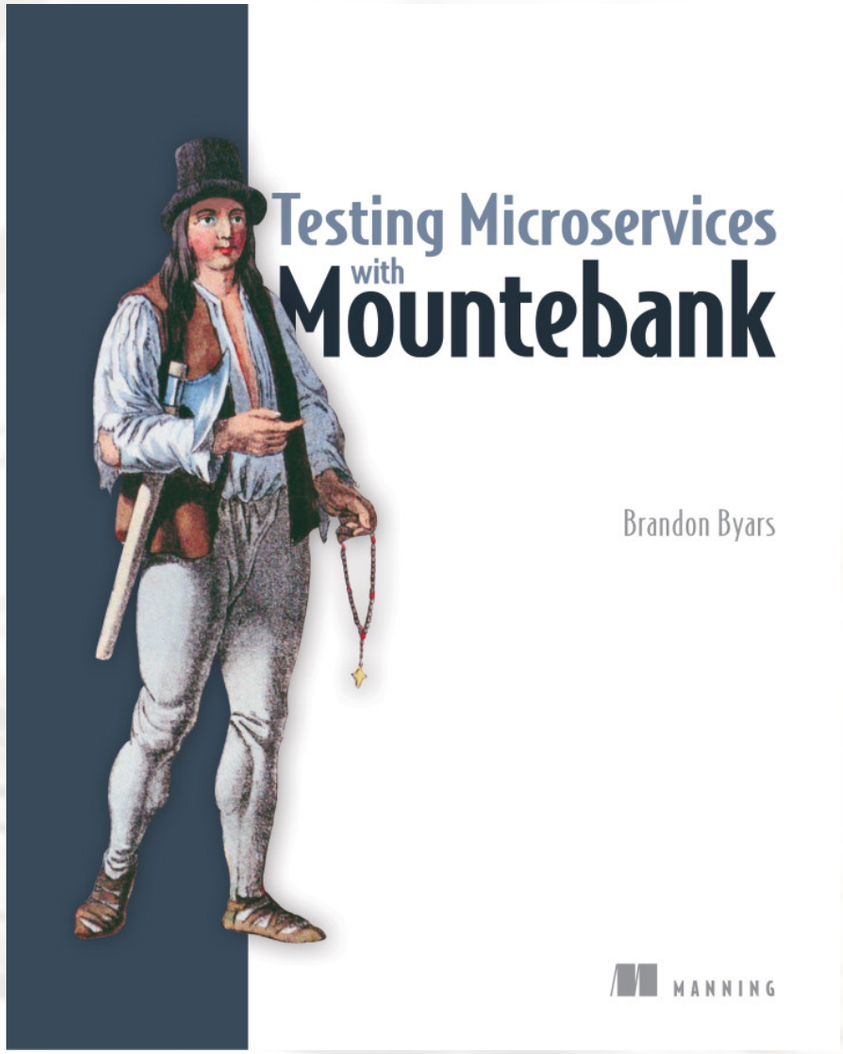
Deciding between response vs. predicate injection

	request	logger	state	callback
predicate	x	x		
response	x	x	x	x

If you need to send a static response back based on a dynamic condition, programming your own predicate and using an is response stays true to the intention of predicates in mountebank.

But response injection is considerably more capable, and you won't hurt my feelings if you move your conditional logic to a response function so you can take advantage of state or async support.

Books to Read and Practice



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