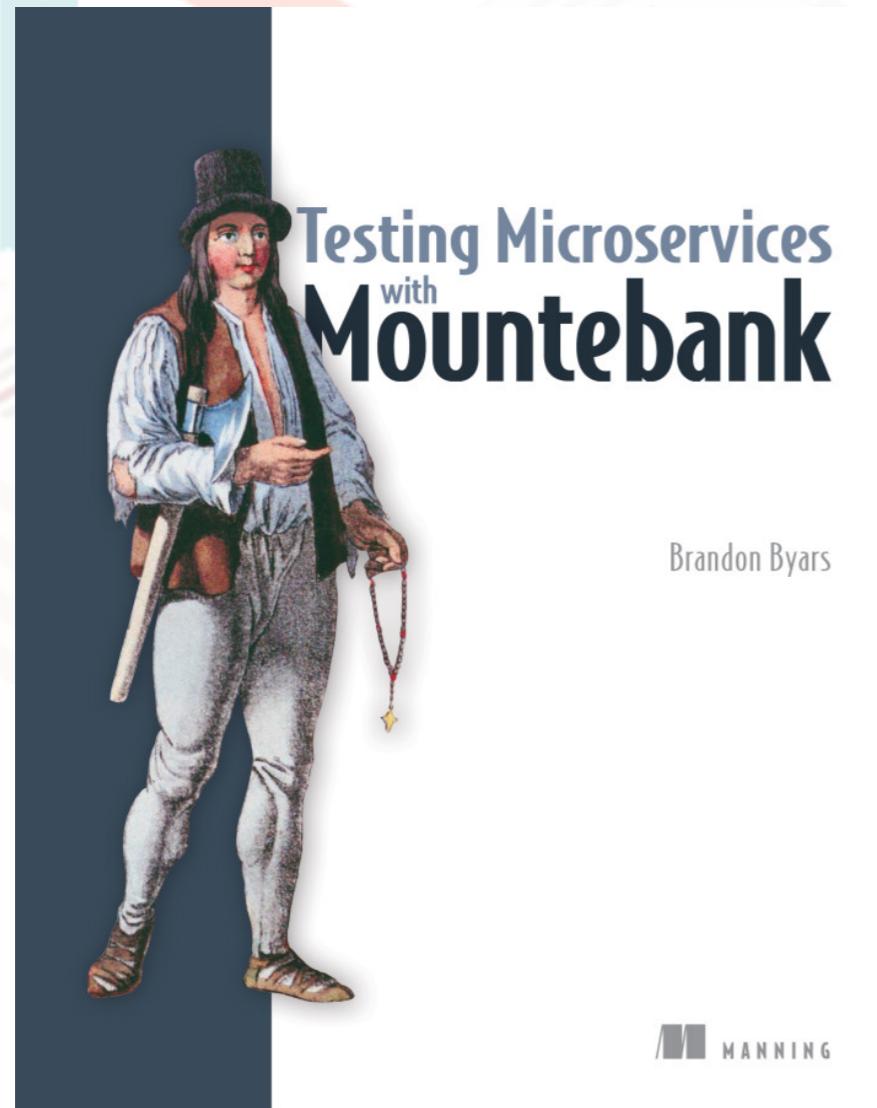


จำลองระบบเพื่อใช้ในการทดสอบ ด้วย

Mountebank 101



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

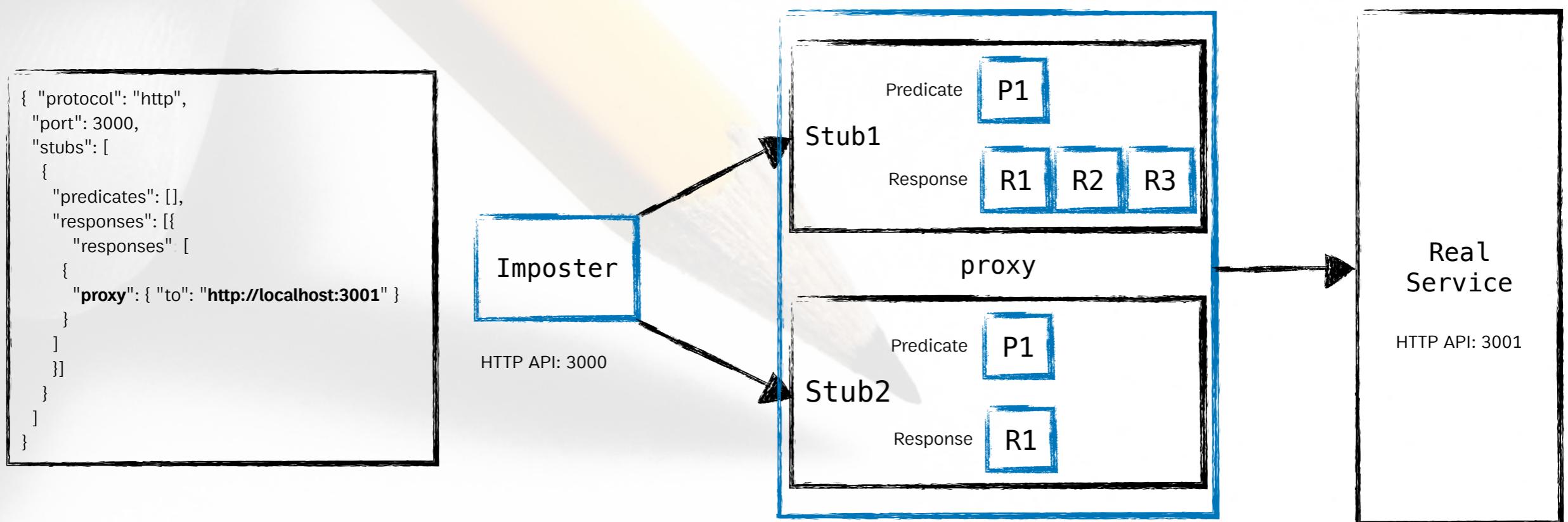


Day5: Record/Replay



Understand Proxy

- Using proxy responses to capture real responses automatically
- Replay the saved responses with the correct predicates



from: Testing Microservice with Mountebank page 131



Setting Up A Proxy



Start Real Service and Proxy Service

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": { "to": "http://localhost:8888" }  
        }  
      ]  
    }  
  ]  
}
```

config directory

start Mountebank: unix/windows

mb start --configfile proxy.json

A Real Server

```
{  
  "protocol": "http",  
  "port": 8888,  
}
```

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json



Call service with postman

request value

POST /items/1 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/2 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/3 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Proxy Mode

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "mode": "proxyAlways"  
          }  
        }  
      ]  
    }  
  ]  
}
```

config directory

start Mountebank: unix/windows

```
mb start --configfile proxy-mode.json
```

Proxy Mode:

- proxyOnce
- proxyAlways
- proxyTransparency

real-service directory

start Mountebank: unix/windows

```
mb start --allowInjection --port 2000 --configfile toy-service.json
```

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

POST /items/1 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/2 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/3 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Generating the correct predicates



Creating Predicates with predicateGenerators: Path

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true  
                }  
              ]  
            ]  
          }  
        ]  
      }  
    ]  
  ]  
}
```

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Predicate Matches:

- path
- query
- body

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

POST /items/1 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/2 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

request value

POST /items/3 HTTP/1.1
HOST localhost:3000
Content-Type: [application/json](#)

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Creating Predicates with predicateGenerators: Query

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true,  
                  "query": true  
                }  
              ]  
            ]  
          }  
        ]  
      }  
    ]  
  ]  
}
```

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Predicate Matches:

- path
- **query**
- body

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

POST /items/1?q=hello HTTP/1.1

HOST localhost:3000

Content-Type: **application/json**

request value

POST /items/2?q=hello HTTP/1.1

HOST localhost:3000

Content-Type: **application/json**

request value

POST /items/3?q=hello HTTP/1.1

HOST localhost:3000

Content-Type: **application/json**

request value

POST /items/3?q=hello&p=hello HTTP/1.1

HOST localhost:3000

Content-Type: **application/json**

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Creating Predicates with predicateGenerators: Specific Query

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true,  
                  "query": { "q": true }  
                }  
              ]  
            ]  
          }  
        ]  
      }  
    ]  
  }]
```

Predicate Matches:

- path
- **query**
- body

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

```
POST /items/1?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json
```

request value

```
POST /items/2?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json
```

request value

```
POST /items/3?q=hello&p=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json
```

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Creating Predicates with predicateGenerators: Body

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true,  
                  "query": true  
                }  
              },  
              {  
                "matches": {  
                  "body": true  
                }  
              }  
            ]  
          }  
        ]  
      }  
    }  
  ]  
}
```

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Predicate Matches:

- path
- query
- **body**

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

```
POST /items/1?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json  
  
{ "name": "hello" }
```

request value

```
POST /itemS/1?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json  
  
{ "name": "hello" }
```

request value

```
POST /items/2?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json  
  
{ "name": "hello" }
```

request value

```
POST /items/3?q=hello HTTP/1.1  
HOST localhost:3000  
Content-Type: application/json  
  
{ "name": "hello" }
```

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Creating Predicates with predicateGenerators: CaseSensitive

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true  
                },  
                "caseSensitive": true  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  ]  
}
```

Predicate Matches:

- path
- query
- body

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

POST /items/1 HTTP/1.1
HOST localhost:3000
Content-Type: application/json

request value

POST /itemS/1 HTTP/1.1
HOST localhost:3000
Content-Type: application/json

request value

POST /items/2 HTTP/1.1
HOST localhost:3000
Content-Type: application/json

request value

POST /items/3 HTTP/1.1
HOST localhost:3000
Content-Type: application/json

Postman: request with

HTTP Method

GET

URL

http://localhost:3000/items/1

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Creating Predicates with predicateGenerators: JSONPath

A Stub Proxy

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "proxy": {  
            "to": "http://localhost:8888",  
            "predicateGenerators": [  
              {  
                "matches": {  
                  "path": true  
                }  
              },  
              {  
                "matches": {  
                  "body": true  
                },  
                "jsonpath": {  
                  "selector": "$.name"  
                }  
              },  
              {  
                "matches": {  
                  "body": true  
                }  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  ]  
}
```

config directory

start Mountebank: unix/windows

mb start --configfile predicate-generators.json

Predicate Matches:

- path
- query
- **body**

real-service directory

start Mountebank: unix/windows

mb start --allowInjection --port 2000 --configfile toy-service.json

Take a look at:

<http://localhost:2525/imposters/3000>



Call service with postman

request value

POST /items/1?q=hello HTTP/1.1
HOST localhost:3000
Content-Type: application/json

```
{ "id": 1, "name": "hello" }
```

request value

POST /items/2?q=hello HTTP/1.1
HOST localhost:3000
Content-Type: application/json

```
{ "id": 2, "name": "hello" }
```

request value

POST /items/3?q=hello HTTP/1.1
HOST localhost:3000
Content-Type: application/json

```
{ "id": 3, "name": "hello" }
```

Postman: request with

HTTP Method

GET

URL

<http://localhost:3000/items/1>

Take a look at:

proxy terminal

Take a look at:

real-service terminal

Take a look at:

<http://localhost:2525/imposters/3000>



Ways to replay a proxy



Replay And Save Proxy

config directory: **open new terminal**

start Mountebank: unix/windows

```
mb replay
```

Take a look at:

```
http://localhost:2525/imposters/3000
```

config directory

start Mountebank: unix/windows

```
mb save --savefile imposters.json --removeProxies
```

Take a look at:

```
imposters.json
```



Tricks



Refactor with Stringify

```
{  
  "is": {  
    "body": {  
      "id": "$ID",  
      "name": "43 Piece Dinner Set",  
      "price": 12.95  
    }  
  }  
}
```

```
{  
  "is": {  
    "body": "<%- stringify(filename, 'template') %>"  
  }  
}
```

CORS: Cross-origin Resource Sharing

```
{  
  "protocol": "http",  
  "port": 3000,  
  "allowCORS": true  
  "stubs": [  
    {  
      "predicates": [  
        {  
          ....  
        },  
        "responses": [  
          {  
            ....  
          }  
        ]  
      ]  
    }  
  ]  
}
```



Use 3rd Party Library

```
{  
  "is": {  
    "statusCode": 200,  
    "body": {  
      "greeting": "Hello, World!",  
      "timestamp": "nothing"  
    },  
    "headers": {  
      "Content-Type": "application/json"  
    }  
  },  
  "_behaviors": {  
    "decorate": "<%- stringify(filename, 'timeStamp.js') %>"  
  }  
}
```

decoration directory

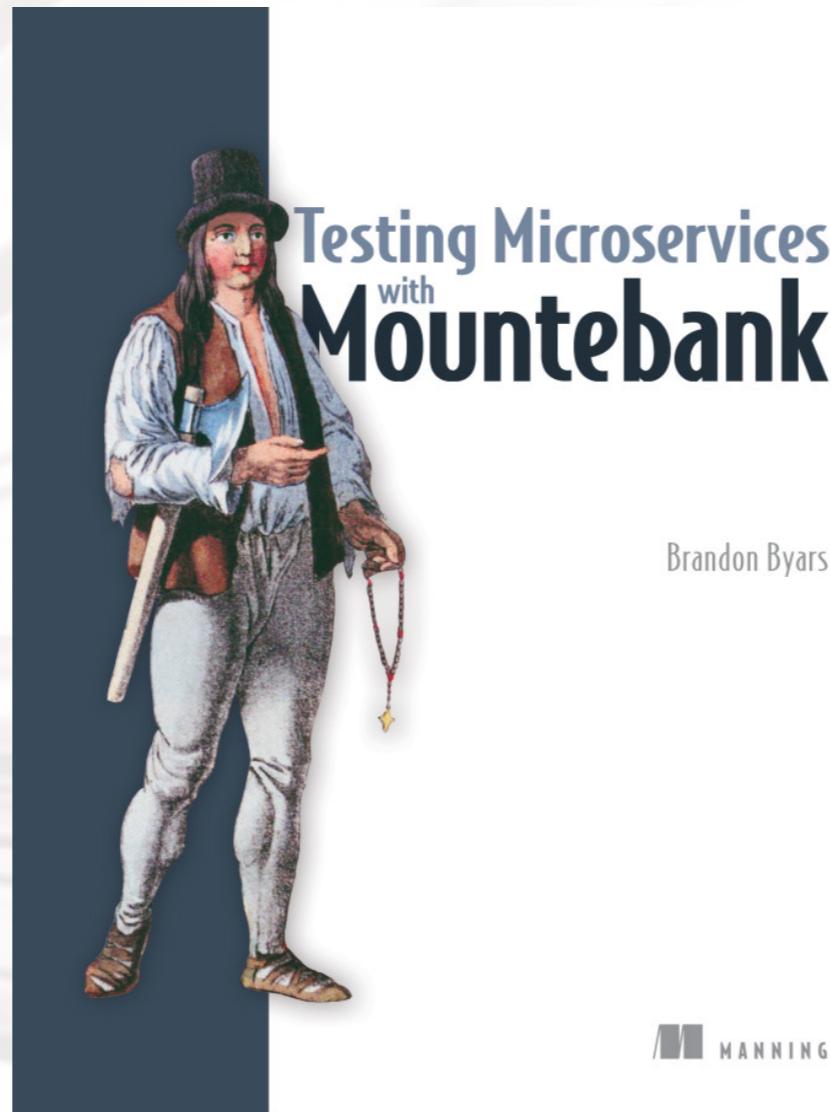
start Mountebank: unix/windows

```
mb start --allowInjection --port 2000 --configfile imposters.json
```

```
function (request, response, logger) {  
  response.body.timestamp =  
  '2018-12-03T10:57:37.842+07:00';  
  
  const path = require('path');  
  const dateFormat = require(path.resolve('node_modules/dateformat/lib/dateformat'));  
  response.body.ldatetime = dateFormat(new Date(), "yyyy-mm-dd'T'HH:MM:ss.l+07:00");  
  response.body.date = dateFormat(new Date(), "yyyy-mm-dd HH Z o");  
  response.body.time = dateFormat(new Date(), "HHMMss Z o");  
  response.body.longtime = dateFormat("longTime");  
}
```



Books to Read and Practice



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



บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่