Portugal JUG

WireMock Demystified

André Martins

andremartins.pt

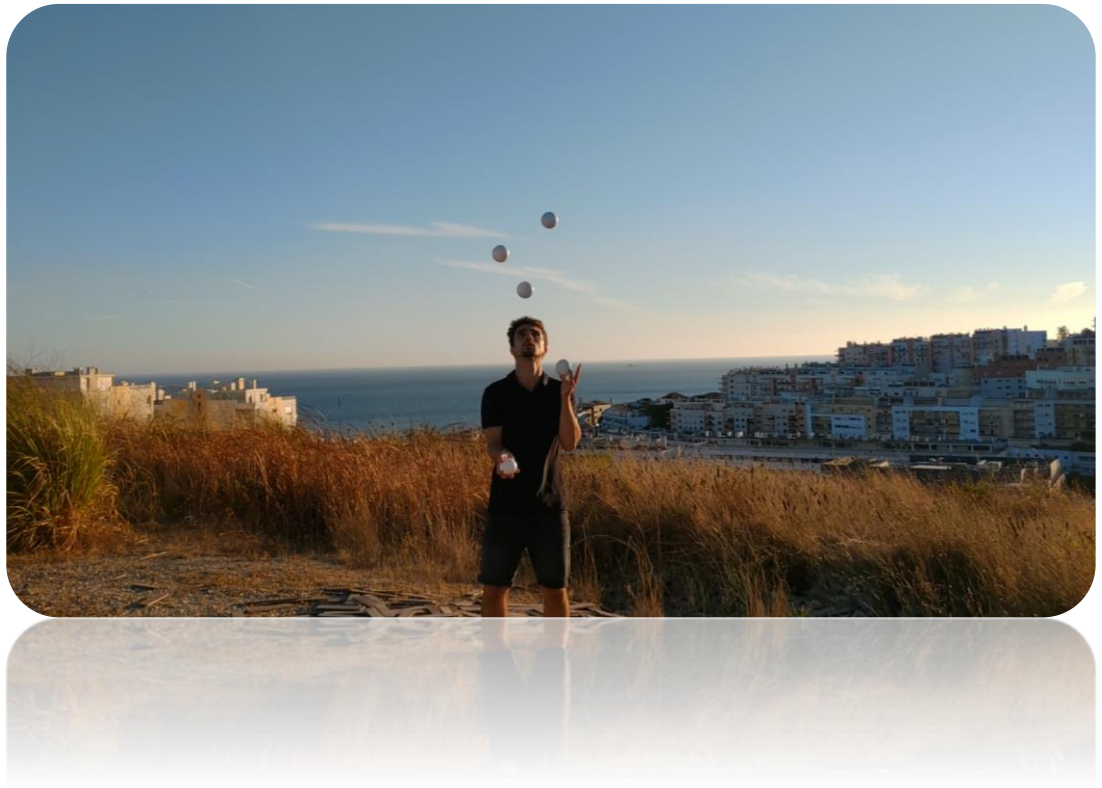
WHO_AM_I

- Bsc Computer Science and Engineering @ ISEL
- Msc Computer Science and Engineering @ IST
- Backend Software Engineer @ Sky Portugal
- Tutor / Teacher as Freelancer
- Juggler

chriptus13

/in/andremartins13

andre@andremartins.pt



Agenda

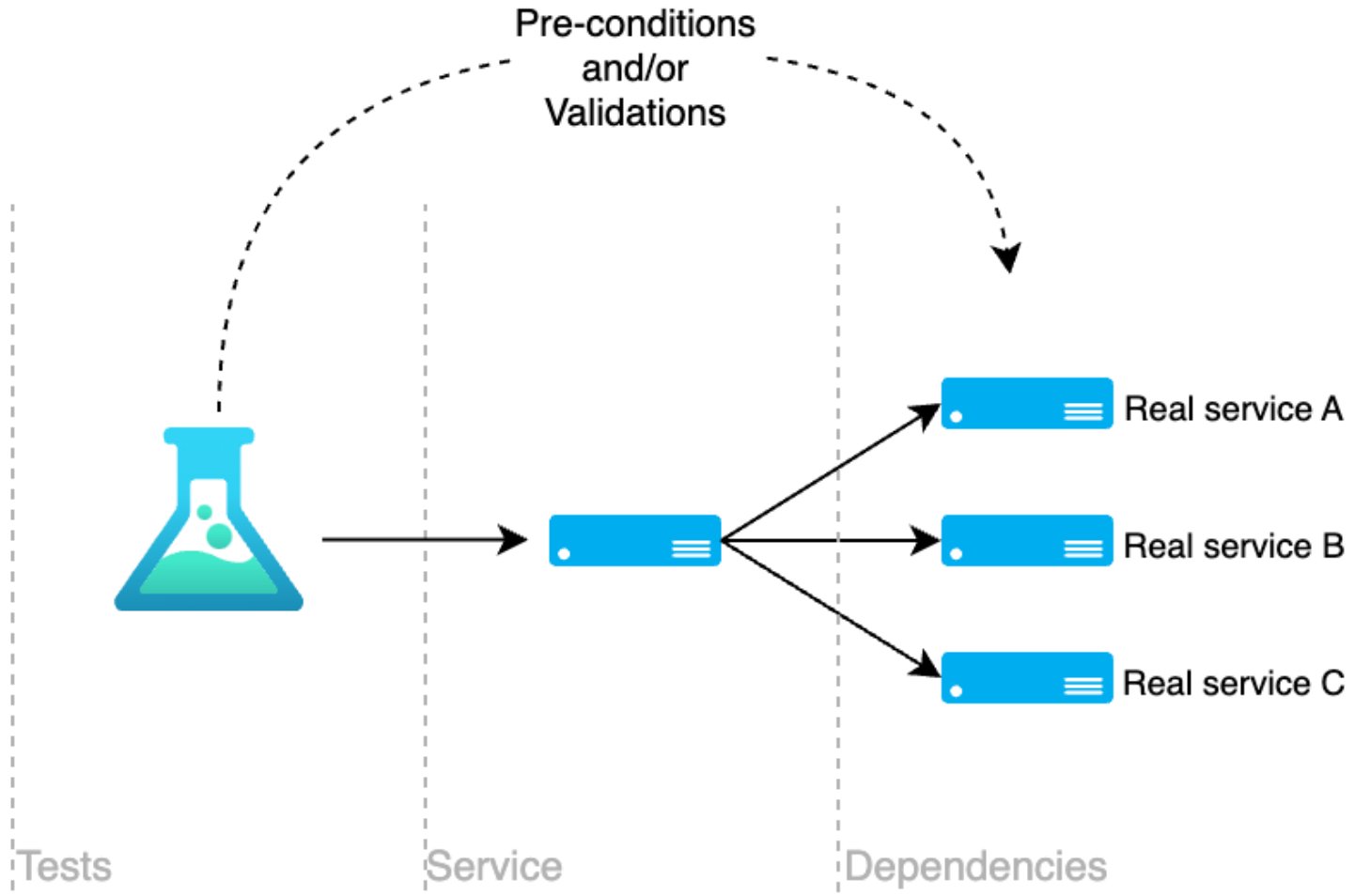
1. What?
2. Why?
3. How?
4. Stubbing
5. Extensions
6. Record & Playback
7. Admin API
8. Demo

What?

- Its a tool for API mock testing with support for **request matching** and **validation**.
- Supported **protocols**:
  - **HTTP(s)**
  - Webhooks & Callbacks
  - gRPC
  - GraphQL
  - (alternative Mountebank for more protocols)
- Stubs can be defined **statically** or **dynamically**.
- Stubs can be defined **programmatically** or declaratively using **JSON**.
- It can run as a **standalone** application (jar) or via **Docker container** (*wiremock/wiremock*)
- It can be extended with custom plugins (docs).

Why?

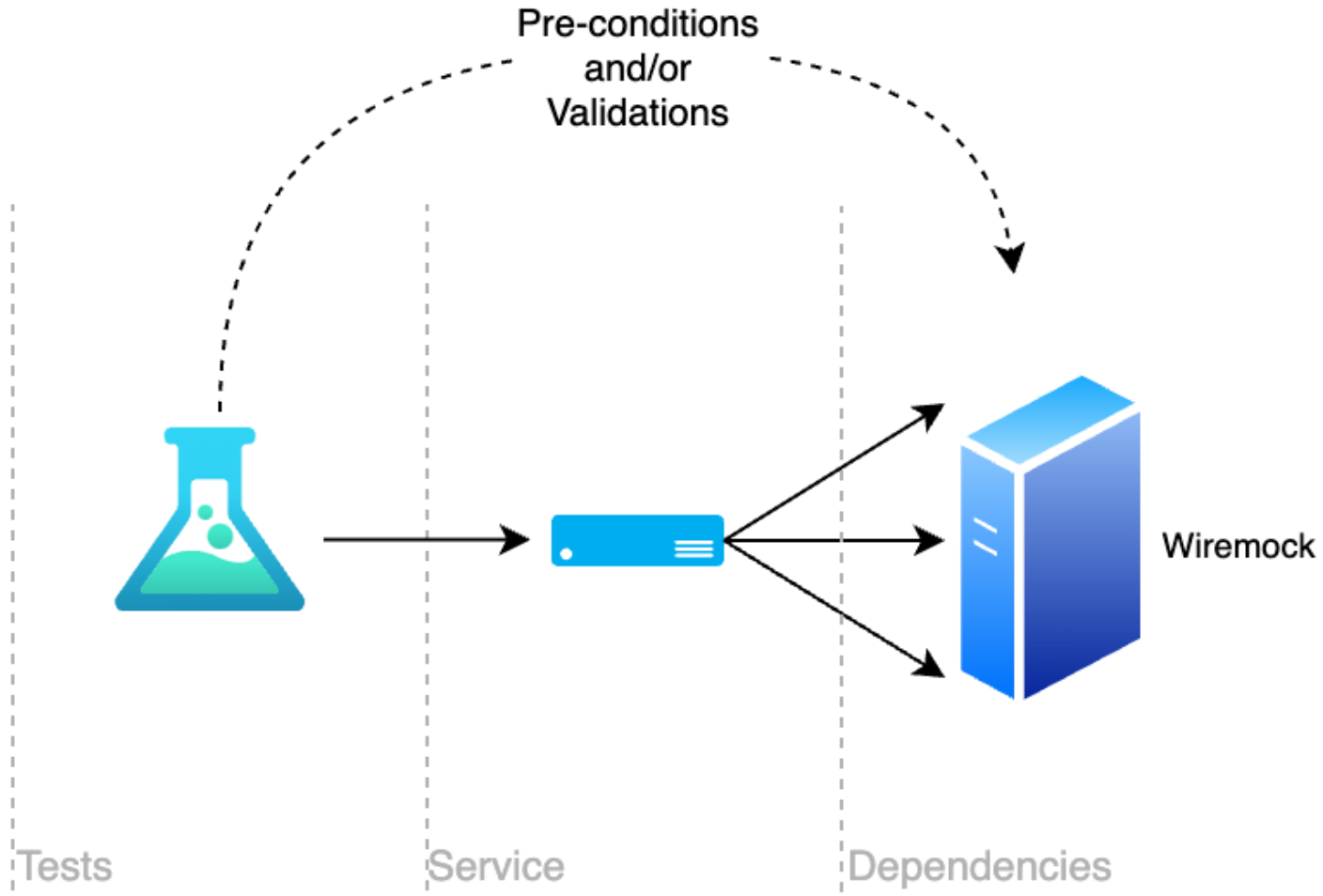
- **Standard** setup for any test:
  - Pre-loaded or dynamically loaded.
  - Common Gherkin steps.
- **Standard** validations.
- Dependency simplification:
  - **Single** dependency*
- Behavior simulation for complex business flows.



* When all dependencies use protocols supported by WireMock.

Why?

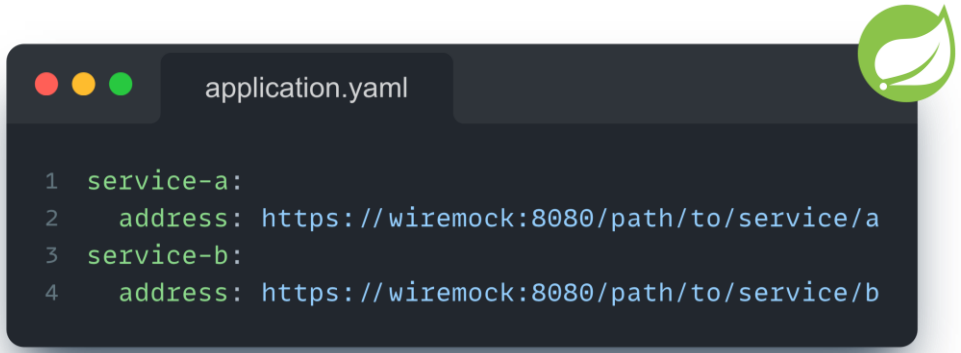
- **Standard** setup for any test:
  - Pre-loaded or dynamically loaded.
  - Common Gherkin steps.
- **Standard** validations.
- Dependency simplification:
  - **Single** dependency*
- Behavior simulation for complex business flows.

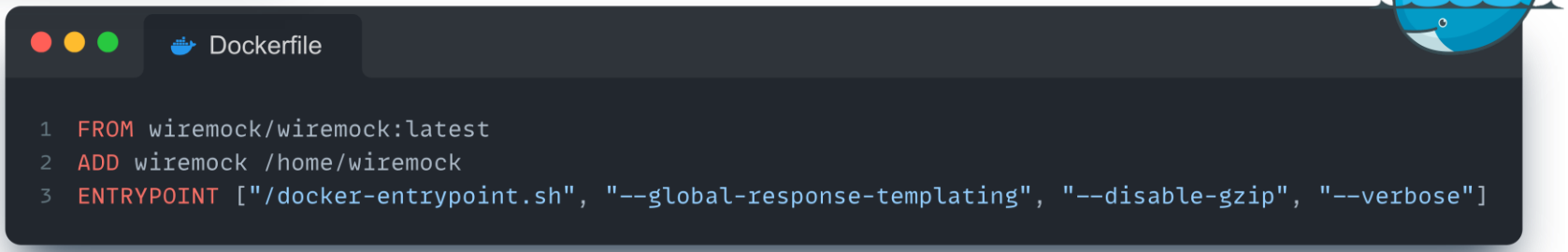


* When all dependencies use protocols supported by WireMock.

How?

- Run/Deploy WireMock.
- Configure application to target WireMock instead of actual service.





https://wiremock.org/docs/standalone/java-jar/#command-line-options

Stubbing

- Stub *uuid* or *id* (generated if not supplied).
- Request matching.
- Response definition/templating.
- Stub metadata*.
- Priority
  - Defaults to **5** where **1** is the highest priority and *Integer.MAX_VALUE* is the lowest.

my-first-stub.json

```
{
  "uuid": "066E5AB8-A8B6-4A85-9EA1-F3BC42A248C6",
  // or
  "id": "066E5AB8-A8B6-4A85-9EA1-F3BC42A248C6",
  "request": {
    "method": "GET",
    "urlPath": "/hello/world",
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "text/plain"
    },
    "body": "Hello World!",
    ...
  },
  "metadata": {
    "name": "stub-name",
    ...
  },
  "priority": 20 // default is 5
}
```

* Can be used to define stub name to find-by-metadata.

Request matching

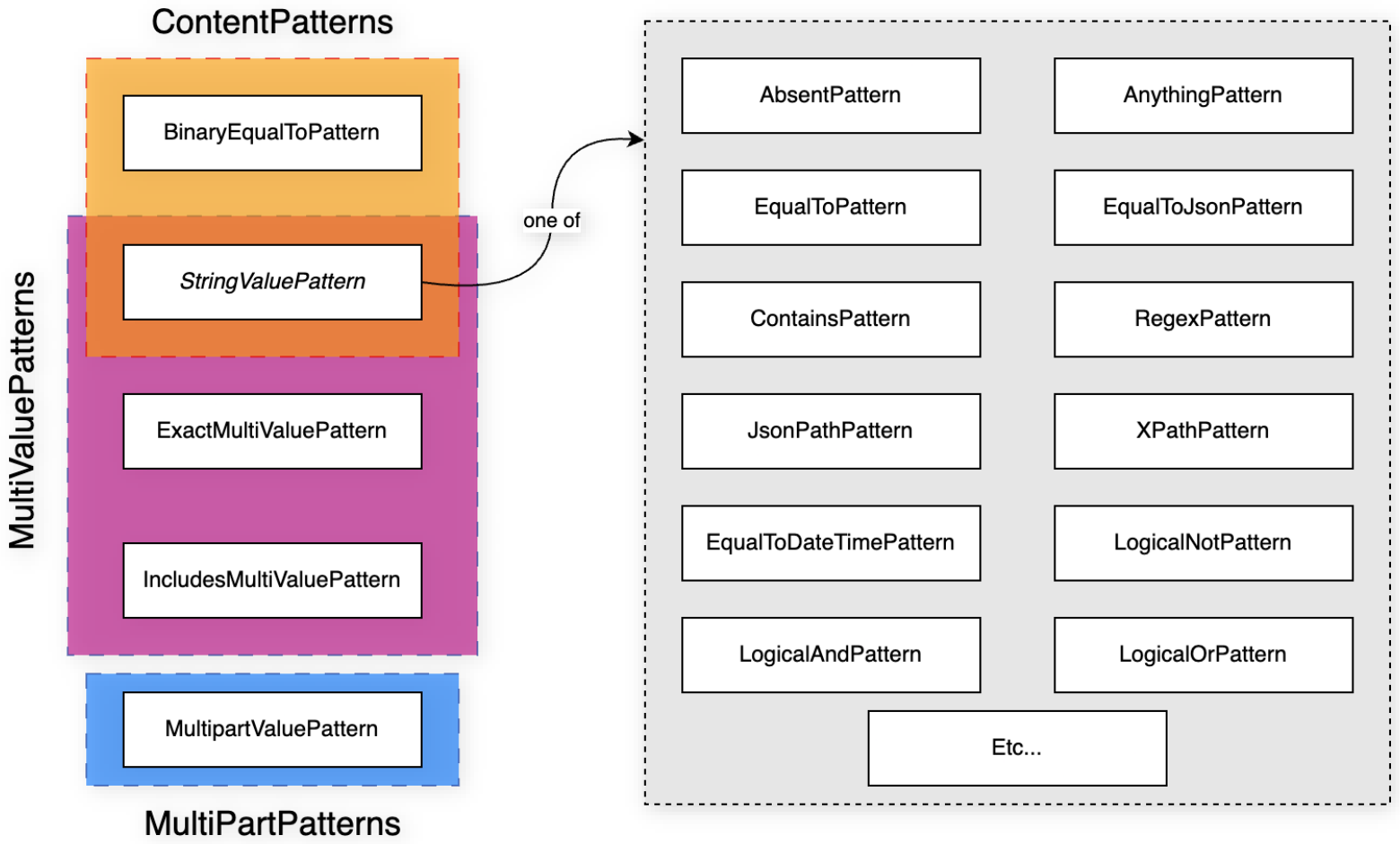
- URL matching with one of:
  - *url*
  - *urlPattern* (w/ Regex)
  - *urlPath*
  - *urlPathPattern* (w/ Regex)
  - *urlPathTemplate* (RFC 6570)
- All other request attributes can be matched by a set of operators/matchers.

request-matching.json

```
{
  "request": {
    "urlPathTemplate": "/users/{userId}/addresses/{addressId}",
    "method": "POST",
    "pathParameters": {
      "userId": {
        "equalTo": "1234"
      },
      "addressId": {
        "equalTo": "5678"
      }
    },
    "queryParameters": {
      "q-param": {
        "equalTo": "some-value"
      },
      ...
    },
    "headers": {
      "Content-Type": {
        "equalTo": "application/json"
      },
      ...
    },
    "cookies": {
      "my-cookie": {
        "equalTo": "my-value"
      },
      ...
    },
    "formParameters": {
      "f-param": {
        "equalTo": "param-value"
      },
      ...
    },
    "bodyPatterns": [
      {
        "equalToJson": "{...}",
        ...
      }
    ]
    ...
  },
  "response": {
    "status": 200
  }
}
```

Matchers

- *StringValuePattern*
  - Headers
  - QueryParams
  - Cookies
  - PathParams
  - FormParams
  - Body
- *ContentPatterns*
  - Body
- *MultiValuePatterns*
  - Headers
  - QueryParams
- *MultipartValuePatterns*
  - Multipart requests



ⓘ JSON schemas for matches reference API - link

Matchers - Fields

equality.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "my-param" : {
        "equalTo" : "some-value",
        // false by default
        "caseInsensitive": true
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

contains.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "other-param": {
        "contains": "my-substr"
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

not-contains.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "foo-param": {
        "doesNotContain": "my-substr"
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

Matchers - Fields

regex.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "bar-param": {
        "matches": "^(.*)\w+[A-Z]*$"
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

not-regex.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "xpto-param": {
        "doesNotMatch": "^my-name-is-(.+)$"
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

date-time.json

```
{
  "request": {
    ...
    "queryParameters" : {
      "date-param": {
        "equalToDateTime": "2024-02-28T00:00:00Z",
        // or
        "after": "2024-02-29T00:00:00Z",
        // or
        "before": "2024-02-27T00:00:00Z",
        // Default ISO8601 format
        "actualFormat": "dd/MM/yyyy",
      }
    }
    ...
  },
  "response": {
    ...
  }
}
```

Matchers - Body

body-json.json

```
{
  "request": {
    ...
    "bodyPatterns": [
      {
        // Json String
        "equalToJson": "{ \"id\": 4 }",
        // false by default
        "ignoreArrayOrder": true,
        // false by default
        "ignoreExtraElements": true
      }
    ]
  },
  "response": {
    ...
  }
}
```

binary.json

```
{
  "request": {
    ...
    "bodyPatterns": [
      {
        // Base64
        "binaryEqualTo": "ABC"
      }
    ]
  },
  "response": {
    ...
  }
}
```

body-json-path.json

```
{
  "request": {
    ...
    "bodyPatterns": [
      {
        "matchesJsonPath": "$.id"
      },
      {
        "matchesJsonPath": "$.name"
      },
      {
        "matchesJsonPath" : "$.[?(@.name == 'André')]"
      }
    ]
  },
  "response": {
    ...
  }
}
```

Matchers - Existential

anything.json

```
{
  "request": {
    ...
    "headers": {
      "my-other-header": {
        // value is irrelevant
        "anything": ""
      }
    }
  },
  "response": {
    ...
  }
}
```

absent.json

```
{
  "request": {
    ...
    "headers": {
      "my-custom-header": {
        // value is irrelevant
        "absent": ""
      }
    }
  },
  "response": {
    ...
  }
}
```

Matchers - Logical

not.json

```
{
  "request": {
    ...
    "queryParameters": {
      "some-query-param": {
        "not": {
          "equalTo": "not-this"
        }
      }
    }
  },
  "response": {
    ...
  }
}
```

and.json

```
{
  "request": {
    ...
    "queryParameters": {
      "some-query-param": {
        "and": [
          {
            "contains": "this"
          },
          {
            "contains": "that"
          }
        ]
      }
    }
  },
  "response": {
    ...
  }
}
```

or.json

```
{
  "request": {
    ...
    "queryParameters": {
      "some-query-param": {
        "or": [
          {
            "contains": "this"
          },
          {
            "contains": "that"
          }
        ]
      }
    }
  },
  "response": {
    ...
  }
}
```

You can compose these and do any combination you need!

Responses

- Specify response payload via one of:
  - *body*
  - *jsonBody*
  - *base64Body*
  - *bodyFileName*

simple-response.json

```
{
  "request": {
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "text/plain"
    },
    "body": "string-value-body"
  }
}
```

json-response.json

```
{
  "request": {
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "application/json"
    },
    "jsonBody": {
      "id": 1234,
      "name": "André"
    }
  }
}
```

Response templating

- Local response templating via ***response-template*** stub response transformer.
- Global* response templating is disabled by default.
  - To enable run WireMock with

  --global-response-templating
- Access to request model attributes for **body** and **headers** templating.
- **Warning**: When global response templating enabled and using *bodyFileName* it will assume a templated file. Either disable global flag or specify transformer parameter:
  - disableBodyFileTtemplating: true

response-templating.json

```
{
  "request": {
    "method": "GET",
    "urlPathTemplate": "/users/{userId}/addresses/{addressId}",
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "application/json",
      "my-custom-header": "{{ request.host }}"
    },
    "jsonBody": {
      "id": "{{ request.path.addressId }}",
      "user": "{{ request.path.userId }}",
      "name": "André",
      "address": {...}
    },
    "transformers": ["response-template"]
  }
}
```

* No need to specify *response-template* transformer on each stub.

Response templating – Request model

`request.url` – URL path and query

`request.path` – URL path. This can be referenced in full or it can be treated as an array of path segments (like below) e.g. `request.path.3`. When the path template URL match type has been used you can additionally reference path variables by name e.g. `request.path.contactId`.

`request.pathSegments.[<n>]` – URL path segment (zero indexed) e.g. `request.pathSegments.2`

`request.query.<key>` – First value of a query parameter e.g. `request.query.search`

`request.query.<key>.[<n>]` – nth value of a query parameter (zero indexed) e.g. `request.query.search.5`

`request.method` – request method e.g. `POST`

`request.host` – hostname part of the URL e.g. `my.example.com`

`request.port` – port number e.g. `8080`

`request.scheme` – protocol part of the URL e.g. `https`

`request.baseUrl` – URL up to the start of the path e.g. `https://my.example.com:8080`

`request.headers.<key>` – First value of a request header e.g. `request.headers.X-Request-Id`

`request.headers.[<key>]` – Header with awkward characters e.g. `request.headers.[$?blah]`

`request.headers.<key>.[<n>]` – nth value of a header (zero indexed) e.g. `request.headers.ManyThings.1`

`request.cookies.<key>` – First value of a request cookie e.g. `request.cookies.JSESSIONID`

`request.cookies.<key>.[<n>]` – nth value of a request cookie e.g. `request.cookies.JSESSIONID.2`

`request.body` – Request body text (avoid for non-text bodies)

https://wiremock.org/docs/response-templating/#the-request-model

Extensions

- WireMock can be customised via a variety of extension points.
- Extension JARs can be automatically loaded if inside */var/wiremock/extensions*
  - Still you need to enable it via *--extensions*
- E.g. *wiremock-faker-extension*

Dockerfile

```
FROM wiremock/wiremock:latest
ADD wiremock /home/wiremock
ADD https://maven.org/.../wiremock-faker-extension-standalone-0.1.1.jar /var/wiremock/extensions
ENTRYPOINT ["/docker-entrypoint.sh", "--extensions", "org.wiremock.RandomExtension"]
```

faker.json

```
{
  "request": {
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "application/json",
      "Link": "{{ random 'Internet.url' }}"
    },
    "jsonBody": {
      "id": "{{ random 'Internet.uuid' }}",
      "username": "{{ random 'Name.username' }}",
      "address": {
        "street": "{{ random 'Address.streetName' }}",
        "country": "{{ random 'Address.country' }}",
        "city": "{{ random 'Address.city' }}",
        "postCode": "{{ random 'Address.postcode' }}"
      }
    }
  }
}
```

https://wiremock.org/docs/extending-wiremock

https://github.com/wiremock/wiremock-faker-extension

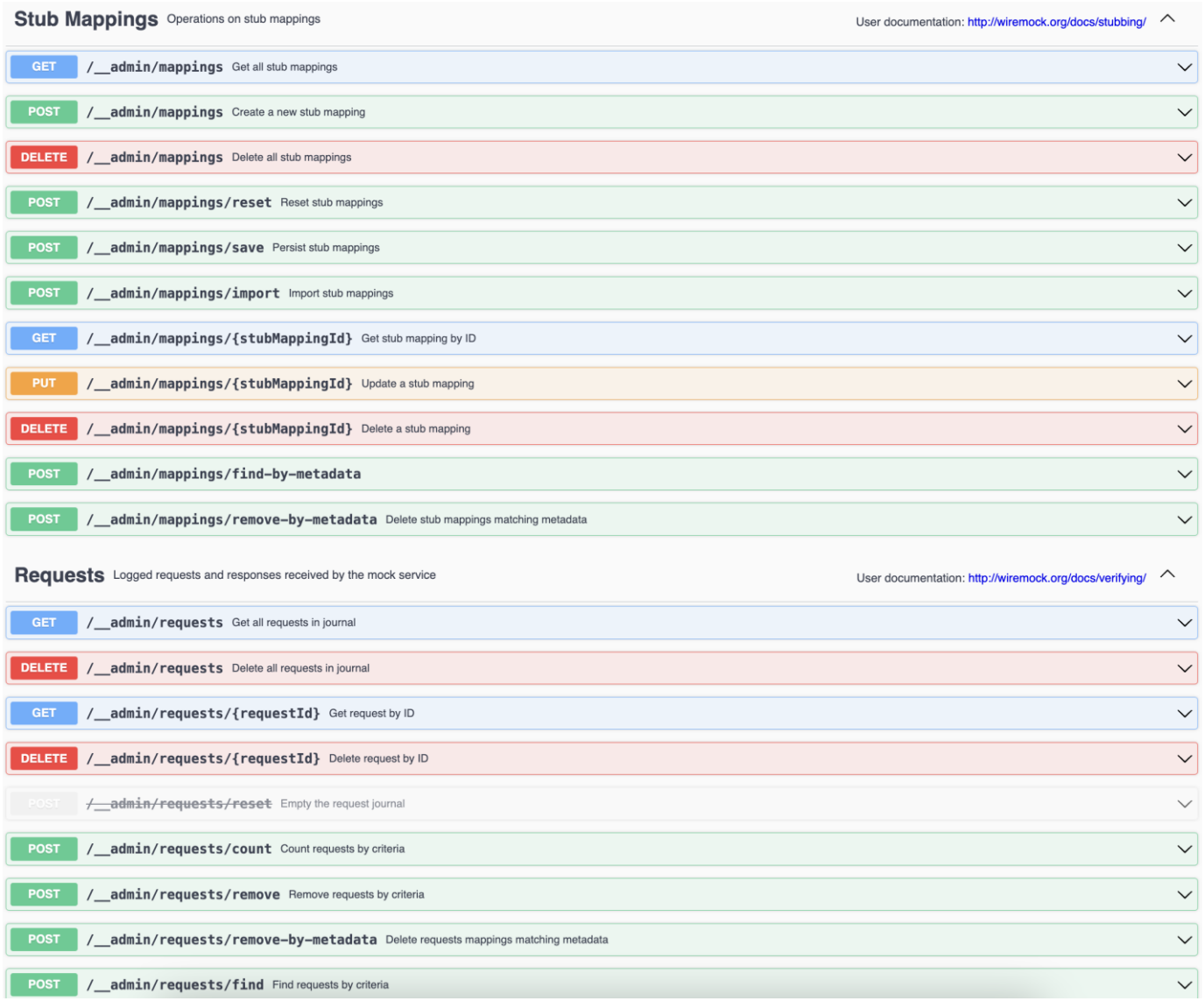
Record & Playback

- Run your service and make requests while **proxying** connections through WireMock.
  1. Go to */__admin/recorder*
  2. Enter real dependency address and hit **Record**.
  3. Point your service to WireMock.
  4. Make requests!
  5. GET */__admin/mappings* – returns created stubs.
  6. GET */__admin/requests* – returns requests made.
- WireMock will **create stubs automatically** with request/response attributes.
- Multiple dependencies?
  - Run multiple WireMock instances.

https://wiremock.org/docs/record-playback

Admin API

- Dynamic stub management – CRUD*.
- WireMock records all requests it receives.
- Verify requests:
  - Validate that your service called WireMock;
  - Validate if request was matched;
  - Validate request fields.
- Docs via */__admin/docs*
  - SwaggerUI */__admin/swagger-ui*
  - OpenAPI Spec** */__admin/docs/swagger*



* With same JSON scheme as stubs – reference API.

** A more complete OpenAPI Spec – link.

Admin API - Verify requests

- Verification can be done via:
  - Count API - */__admin/requests/count*
  - Find API - */__admin/requests/find*
- WireMock JVM library can be used:
  - *org.wiremock:wiremock*
  - *org.wiremock:wiremock-standalone*

count-matching-requests.http

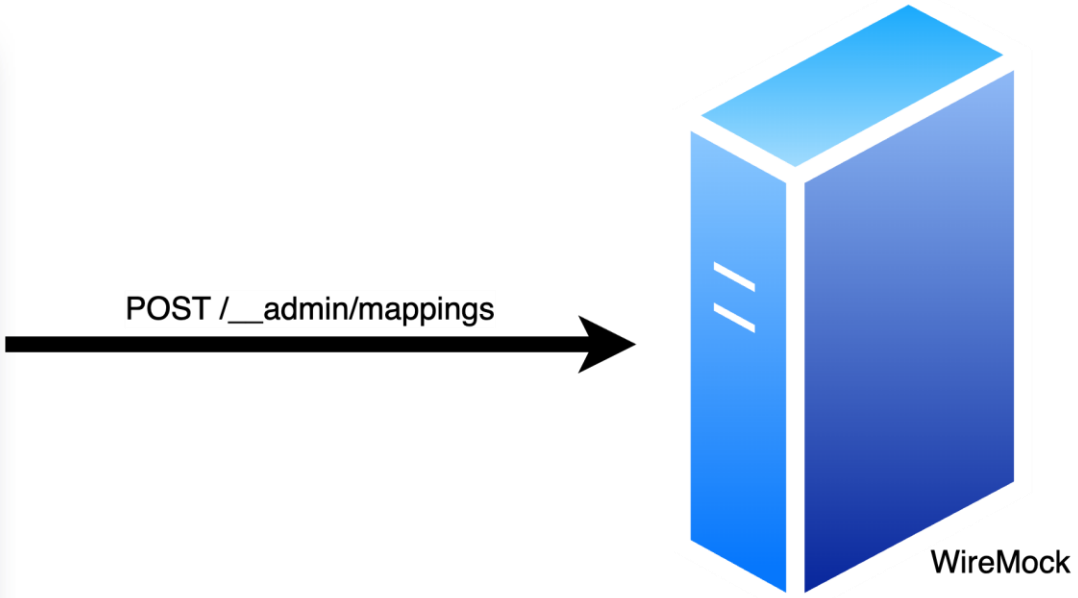
```
# Make actual call
POST https://wiremock:8080/users/123/addresses
Content-Type: application/json

{
  "username": "André",
  "address": {...}
}

# Count requests matching in WireMock
POST https://wiremock:8080/__admin/requests/count
Content-Type: application/json

{
  "method": "POST",
  "urlPath": "/users/123/addresses",
  "headers": {
    "Content-Type": {
      "equalTo": "application/json"
    }
  }
  "bodyPatterns": [
    {
      "matchesJsonPath": {
        "expression": "$.username",
        "equalTo": "André"
      }
    }
  ]
}
```

https://wiremock.org/docs/standalone/java-jar

Admin API – Dynamic stubbing

my-first-stub.json

```
{
  "uuid": "066E5AB8-A8B6-4A85-9EA1-F3BC42A248C6",
  // or
  "id": "066E5AB8-A8B6-4A85-9EA1-F3BC42A248C6",
  "request": {
    "method": "GET",
    "urlPath": "/hello/world",
    ...
  },
  "response": {
    "status": 200,
    "headers": {
      "Content-Type": "text/plain"
    },
    "body": "Hello World!",
    ...
  },
  "metadata": {
    "name": "stub-name",
    ...
  },
  "priority": 20 // default is 5
}
```



Admin API – Get/Update stubs

- Get and/or update stubs via stub uuid.
- If stub metadata name specified we can use it to find the stub.

get-update-stub.http

```
# Retrieve stub
GET https://wiremock:8080/__admin/mappings/{stub-uuid}

# Update stub
PUT https://wiremock:8080/__admin/mappings/{stub-uuid}
Content-Type: application/json

{
  <stub>
}

# Find stub by metadata name
POST https://wiremock:8080/__admin/mappings/find-by-metadata
Content-Type: application/json

{
    "matchesJsonPath": {
        "expression": "name",
        "equalTo": "stub-name"
    }
}
```

Demo

Q&A

Thank you!

André Martins