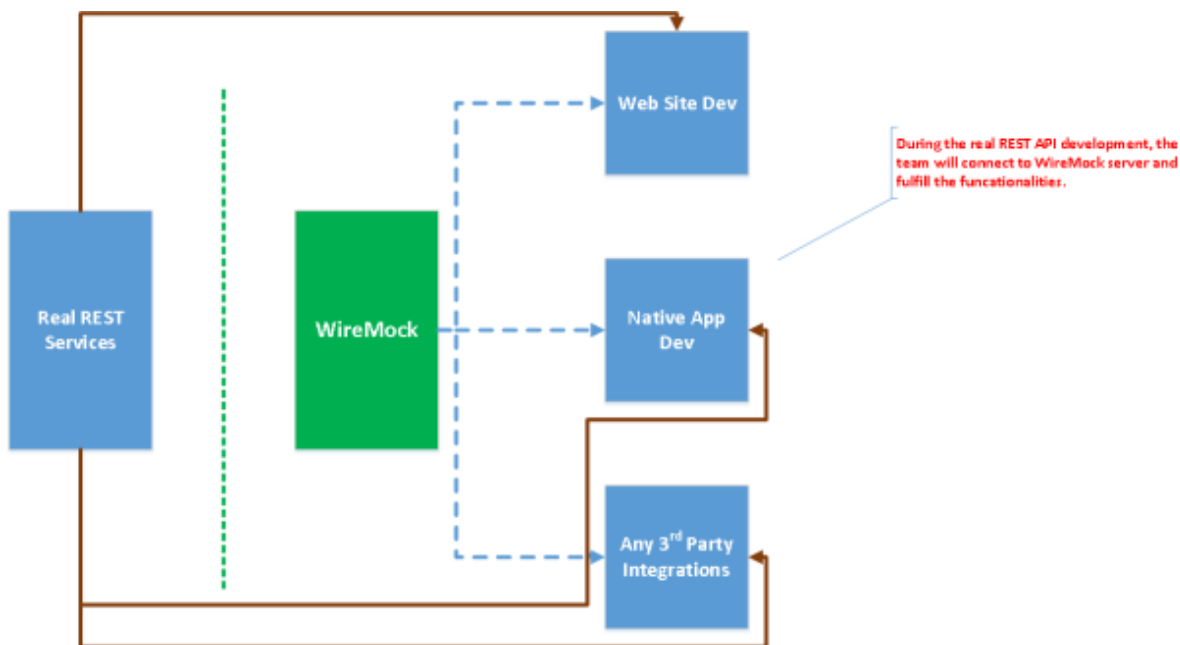


# WireMock: Mock Your REST APIs

by Siva Prasad Rao Janapati  MVB · Mar. 04, 16 · Integration Zone · Analysis

SnapLogic is the leading self-service enterprise-grade integration platform. Download the 2018 GartnerMagic Quadrant for Enterprise iPaaS or play around on the platform, risk free, for 30 days.

As we are extensively working on “OmniChannel” applications with REST APIs approach, mocking the REST services is essential in our day to day development. As part of our development first we are going to identify the REST APIs to expose and the request and response details. Once we finalize these details the backend development team use to create sample request/response (i.e XML/JSON response) for each API and the same can be consumed by the front end team and the native app development team. Till the backend development team provides the APIs the other teams use to hardcode the sample request and responses to complete the development. The “**WireMock**” provides an elegant way to mock the REST APIs. The below diagram depicts how “**WireMock**” fits in our development environment.



We can run **WireMock** as “Standalone mode” or “deployed into servlet container.” In this article, we will see running WireMock as standalone mode. To setup, WireMock follow the below steps.

**Step 1:** Download the WireMock standalone jar from here.




**Step 2:** Run the standalone jar by giving below command.

```
1 java -jar wiremock-1.57-standalone.jar --verbose
```

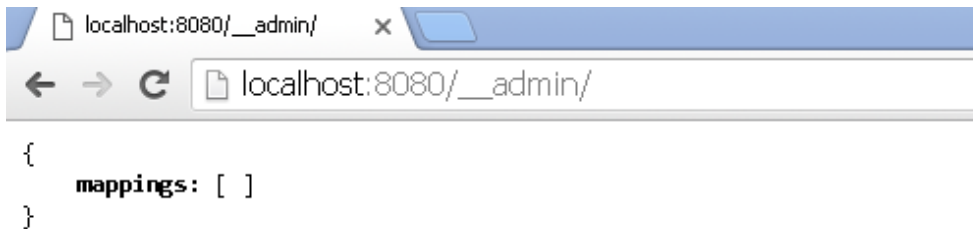
There are several command line options available. You can find the same here. If you are not passing `-port` <https://dzone.com/articles/wiremock-mock-your-rest-apis>

option by passing the port number, by default it will run on 8080.

After running the WireMock, you will see the below folder structure from where the WireMock standalone app is running.

 _files	2/24/2016 10:38 PM	File folder	
 mappings	2/25/2016 8:54 PM	File folder	
 wiremock-1.57-standalone.jar	2/24/2016 10:37 PM	Executable Jar File	6,774 KB

Now, if you send the request to [http://localhost:8080/\\_admin](http://localhost:8080/_admin), then you will empty mappings.



**Step 3:** Now, we will create mapping files for GET and POST requests. Here I want to simulate GET product and CREATE product requests. Now we will see GET request and response mappings.

Request	
End Point	<a href="http://localhost:8080/product/p0001">http://localhost:8080/product/p0001</a>
Method	GET
Response	
Status	200
Content Type	<a href="#">application/json</a>
body	<pre>{   "_id": "p0001",   "product_display_name": "Brother - MFC-J4320DW Wireless All-In-One Printer - Black/Gray",   "category_ids": [     "pcmcat247400050001"   ],   "is_active": true,   "on_sale": true,   "price": {     "list_price": 149.99,     "sale_price": 123.99   },   "upc": "012502637677",   "mpn": "MFC-J4320DW",   "manufacturer": "Brother",   "product_short_description": "4-in-1 functionalityBuilt-in wireless LAN (802.11b/g/n)Prints up to 20 ISO ppm in black, up to 18 ISO ppm in color (Print speeds vary with use. See mfg. for info on print speeds.)150-sheet tray2.7" }</pre>

Create product.json

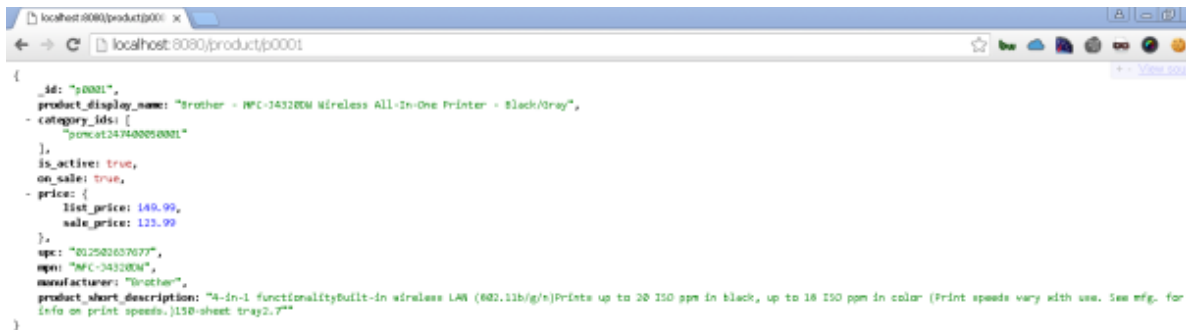
file under "mappings" folder depicted in step2. Make sure that the body content is properly formatted by keeping the content in a single line and the double quotes are escaped.

```

1
2 {
3   "request":
4   {
5     "urlPattern": "/product/p0001",
6     "method": "GET"
7   },
8
9   "response":
10  {
11    "status": 200,
12    "headers":
13    {
14      "Content-Type" : "application/json"
15    },
16    "body": "{ \"_id\" : \"p0001\", \"product_display_name\" : \"Brother - MFC-J4320DW Wireless
17  }
18 }

```

**Step 4:** Now, restart the WireMock and send the request to <http://localhost:8080/product/p0001>. You will get the product details JSON response.



```

{
  "_id": "p0001",
  "product_display_name": "Brother - MFC-J4320DW Wireless All-In-One Printer - Black/Gray",
  "category_ids": [
    "p0001"
  ],
  "is_active": true,
  "on_sale": true,
  "price": {
    "list_price": 149.99,
    "sale_price": 123.99
  },
  "sku": "012582637677",
  "mpn": "MFC-J4320DW",
  "manufacturer": "Brother",
  "product_short_description": "4-in-1 functionalityBuilt-in wireless LAN (802.11b/g/n)Prints up to 20 ISO ppm in black, up to 18 ISO ppm in color (Print speeds vary with use. See mfg. for info on print speeds.)150-sheet tray2.7\"
}

```

**Step 5:** We will see how to create POST request and response mapping.

Request	
url	http://localhost:8080/createProduct
method	POST
Request body	<pre> {   "product_display_name": "Brother - MFC-J4320DW Wireless All-In-One Printer - Black/Gray - New Product",   "category_ids": [     "p0001"   ],   "is_active": true,   "on_sale": true,   "price": {     "list_price": 149.99,     "sale_price": 123.99   },   "sku": "012582637677",   "mpn": "MFC-J4320DW",   "manufacturer": "Brother",   "product_short_description": "4-in-1 functionalityBuilt-in wireless LAN (802.11b/g/n)Prints up to 20 ISO ppm in black, up to 18 ISO ppm in color (Print speeds vary with use. See mfg. for info on print speeds.)150-sheet tray2.7\" } </pre>
Response	
status	200
Content type	application/json
body	<pre> {   "_id": "p0001",   "product_display_name": "Brother - MFC-J4320DW Wireless All-In-One Printer - Black/Gray- New Product",   "category_ids": [     "p0001"   ],   "is_active": true, </pre>

```

    "price": {
      "list_price": 149.99,
      "sale_price": 129.99
    },
    "upc": "812582637677",
    "mpn": "MFC-J4320DW",
    "manufacturer": "Brother",
    "product_short_description": "4-in-1 functionality&uilt-in wireless LAN (802.11b/g
/n)Prints up to 20 IPS ppm in black, up to 18 IPS ppm in color (Print speeds vary with use. See mfg. for info on print speeds.)150-sheet tray2.7"
  }
}

```

Place the mapping

JSON file under “mappings” folder and restart the server.

The mapping JSON is given below.

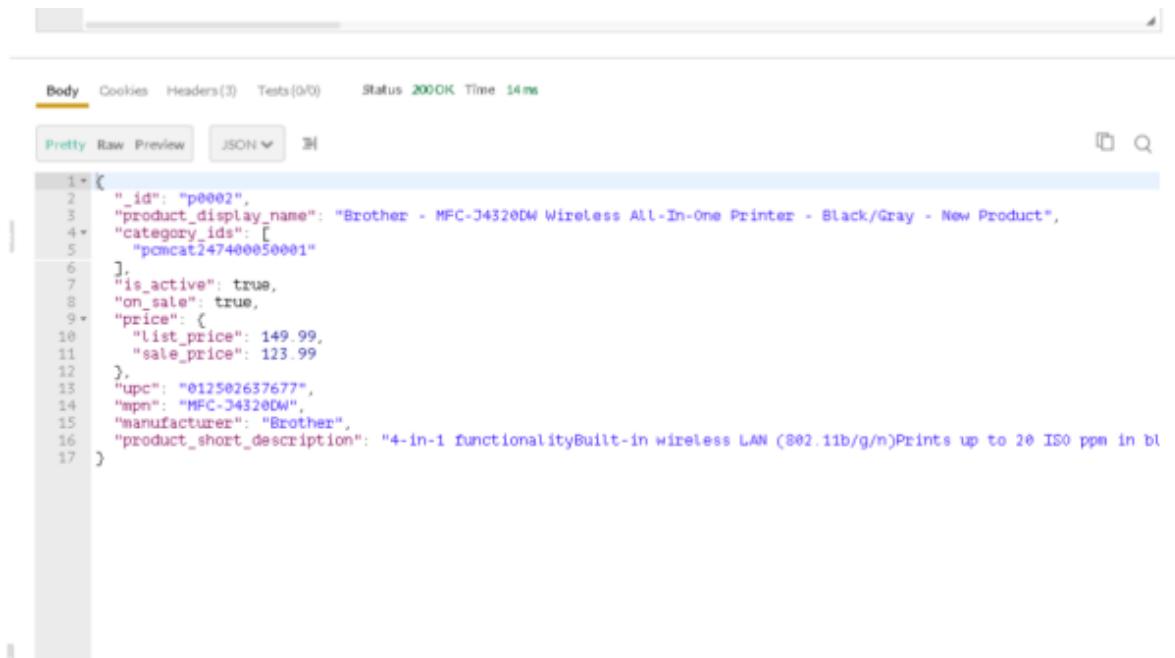
```

1
2  {
3    "request":
4    {
5      "urlPattern": "/createProduct",
6      "method": "POST",
7      "bodyPatterns" : [{
8        "equalToJson" : "{ \"product_display_name\" : \"Brother - MFC-J4320DW Wireless All-In-One Pr
9      }]
10   },
11
12   "response":
13   {
14     "status": 200,
15     "headers":
16     {
17       "Content-Type" : "application/json"
18     },
19     "body": "{ \"_id\" : \"p0002\", \"product_display_name\" : \"Brother - MFC-J4320DW Wireless /
20   }
21 }

```

**Step 6:** Send the POST request by using any request poster. Here I used “POSTMAN” and send the request by passing request body.





By following above steps you are equipped to mock the services, and WireMock is handy to that. There are similar alternative tools to try.

- Betamax
- REST-driver
- MockServer
- Moco

---

**Download A Buyer's Guide to Application and Data Integration, your one-stop-shop for research, checklists, and explanations for an application and data integration solution.**

---

## Like This Article? Read More From DZone



**Mocking REST API with WireMock  
— Recording and Manual Modes**



**Writing Mocks With WireMock and  
CSV Extension**



**Using WireMock to Mock  
Underlying Services for REST  
Testing in Spring Boot**



**Free DZone Refcard  
Foundations of RESTful  
Architecture**

Topics: WIREMOCK , MOCK SERVICES

Published at DZone with permission of Siva Prasad Rao Janapati , DZone MVB. [See the original article here.](#)

Opinions expressed by DZone contributors are their own

Opinions expressed by DZone contributors are their own.

# Integration Partner Resources

The State of API Integration 2018 [Report]

Cloud Elements

|

The State of API Integration 2019 [Survey]

Cloud Elements

|

How to Transform Your Business in the Digital Age [Whitepaper]

Cloud Elements

|

Reasons Why Your Legacy Data Integration Plan Must Change

MapLogic

|