# **Developing and testing APIs using Postman**

- Overview
  - Concepts
    - Implementing tests
  - Integrating with Xray
- Requirements
- Example
- o Postman Echo API
- Tips
- References

### Overview

Postman, more than a utility, is a collaboration platform for developing APIs.

Normally, it is used as a way to quickly interact with existing APIs without having to code HTTP requests by hand.

It provides support for HTTP based APIs, including REST and GraphQL.

Postman also provides the ability to write tests and use Chai assertions, as seen on these Postman test examples.

With Postman, comes also a built-in (test) collection runner; it is also possible to execute tests from the outside, using a CLI tool named Newman.

#### Concepts

- request: an API request (e.g. HTTP POST on some URL, with some values)
- authentication: authentication for the API request (e.g. HTTP basic auth, etc); can be defined at multiple levels and inherited
   collection: a way of grouping multiple requests
- folders within the collection: a way to better organize requests within the collection
- variables: can be defined at multiple levels (e.g. global, collection, environment, local, ...)
- test: a test; can be defined at request, folder or collection level
- pre-request script: some code execute before each test; can also be defined at request, folder or collection level
- environment: an abstraction of some test environment that describes a context for running the requests; it consists of one description plus a set of
  variables with their corresponding values

### Implementing tests

Testing is achieved through the usage of scripts.

Tests can be implemented using Javascript and making use of Postman APIs/objects assisted by Chai assertions.

One or more tests can be defined at the request level, or even at the whole collection level.

					·					,	
	History	Collections	APIs	► GET F	Request						
+ 1	New Collection		Trash								
•	Postman Echo 37 requests			<b>GET</b> Params	<ul> <li>Auth</li> </ul>	https://po orization	stman-echo.com/ Headers (6)	'get?foo1=b Body	par1&foo2=bar2 Pre-request Script	Tests ●	Settings
)   				1 2 3			is ok", funct o.have.status				
- ► 1	Authentication Met	thods		4							2
▶ 1	Cookie Manipulatio	on		5 6					quest queries", f foo1', 'bar1')	<pre>function () {</pre>	
. ⊦ 1	Utilities			7 8	<pre>});</pre>	and have	.jsonBody('arg	gs.foo2',	'bar2');		
▶ 1	Utilities / Date and	Time		0	51;						
. ► 1	Utilities / Postman	Collection									
⊢ I	Auth: Digest										

Pre-request scripts may be useful as a means to initialize some data before the test or to implement some test setup code.

Variables can be defined at multiple levels and can be used to make maintenance easier; the sample applies to authentication, which can also make use of variables.

A test is defined by using "pm.test()".

```
example of a test that looks at the response's HTTP status code
```

```
pm.test("response is ok", function () {
    pm.response.to.have.status(200);
});
```

In Postman, quoting Postman documentation, the pm object encloses all information pertaining to the script being executed and allows one to access a copy of the request being sent or the response received. It also allows one to get and set environment and global variables.

Therefore, pm can be used to access the response, to perform assertions or even to make some requests.

### Integrating with Xray

Integrating with Xray, in order to have visibility of API testing results in Jira, can be done by simply submitting automation results to Xray through the REST API or by using one of the available CI/CD plugins (e.g. for Jenkins).

This can be achieved using Newman and one of its reporters capable of generating a JUnit XML file.

### Requirements

- Postman
- Newman
- newman-reporter-junitxray or newman-reporter-junitfull
- Xray Test Management Jenkins plugin (optional)

### Example

### Postman Echo API

In this example, we're going to use Postman' sample Echo API as a way to showcase some tests and their integration with Xray.

The Postman Echo API provides a set of endpoints that we'll exercise.

We start by cloning an existing Postman collection from a template and importing it to Postman.

ostman Echo		Run in Postman
ostman Echo is service you can use to	test your REST clients and make sample API calls.	View Documentation
	to test your REST clients and make sample API calls. uT, various auth mechanisms and other utility	PUBLISHER
The documentation for the endpoint https://postman-echo.com	Run in	an is a collaboration platform for API pment used by 8M+ developers world 400K+ organizations, to access 250M+
	Postman for Web     Supported on latest Chrome, Firefox & Edge browsers	GORIES
	Postman for Mac V3.2.14.1 or HIGHER	LOPER
	Don't have the app yet? Get the app	RTS

The collection contains a request per each endpoint, where each request has one or more tests.

+ New In	mport Runner 📭 🔻		🖬 My Workspace 👻 🧍	Invite	<u>o</u>
Q Filter			GET GET Request × + ····		
History	Collections	APIs	GET Request		
+ New Collecti	ion	Trash	· de request		
Postman I	Echo ★		GET • https://postman-echo.com/get?foo1=bar1&foo2=bar2		
37 requests			Params  Authorization Headers (6) Body Pre-request	cript Tests  Settings	
🕶 🖿 Request	t Methods	•••	Query Params		
GET GET Re	equest		KEY	VALUE	DESCRIPTION
POST POST R	Raw Text		foo1	bar1	
POST POST F	Form Data		foo2	bar2	
PUT PUT Re			Key	Value	Description
PATCH PATCH					
DEL DELETE					
Headers					
	tication Methods				
	Manipulation				
Utilities					
	/ Date and Time				
	/ Postman Collection				
Auth: Dig	igest				
CET	- https://w	actman acha	com/act2fco1-bcr19fco2-bcr2		
GET	<ul> <li>https://p</li> </ul>	iostman-echo.	.com/get?foo1=bar1&foo2=bar2		
Params	Authorization	Headers	(6) Body Pre-request Script	Tests  Settings	
Falallis	Authonization	Headers	(o) Body Pre-request script	Tests Settings	
1					
1	pm.test("respons				
2	pm.response.	to.have.sta	atus(200);		
3	});				
4					
5	pm.test("respons	e body has	json with request queries", fund	ction () {	
6		-	onBody('args.foo1', 'bar1')	· · ·	
7		-	<pre>'args.foo2', 'bar2');</pre>		
		e.jsonbody(	arysinouz, barz );		
8	<pre>});</pre>				

In the previous example, we can see two tests: one for validating a successful HTTP request based on the status code and another that checks the response's JSON content.

The collection (or a subset of its tests) can be run using the Collection Runner.

+ New Import	Runner 📭 🔻										н	ly Worksp	ace 🔻
<b>Q</b> Filter					Po	stma	an Ech	0				×	
History	Collections A	PIs			API				on is not	linked to	anv API		
New Collection		Tra	sh		_	Share		un		in web	••••		-
Postman Echo ★		- E	4	1		Share		un	view	in web			&foo2=b
37 requests		- L.	••••	ť_	Docu	ument	tation	Mon	nitors	Mocks	Ch	angelog	re-requ
Request Methods	5				=	Learn	how to	docum	ient you	r request	s		
GET GET Request				Ŀ									
POST POST Raw Text										use to tes i. It provi		REST 🖉	
POST POST Form Data	a						for GET			, various	auth		
PUT PUT Request										points as w	vell as		
PATCH PATCH Request						mple re <mark>o.com</mark>	esponse	s can b	e found	at https:	//postm	ian-	
DEL DELETE Request						0.00111							
Authentication M	ethods					Req	uest Me	thods					
Cookie Manipulat							ders						
Utilities							henticat						
🖿 Utilities / Date an	d Time						kie Man	ipulati	on				
🖿 Utilities / Postmar	n Collection					Utili	ities ities / Da		The				
Muth: Digest							ities / Da						
llection Runner			_	_						My W	orkspac	e 🔻	
hoose a collection or folder		RUM	ORDEF	R									
		RONG	JADE)			_							
Q Search for a collection	on or tolder						ET Reques						
Postman Echo	8						OST Form						
Request Methods				lin .			JT Reques						
Authentication Method	ds		<b>~</b>	in i	► PA	асн РА	TCH Requ	Jest					
Cookie Manipulation			<b>~</b>		► D	DEL DE	ELETE Req	uest					
Utilities				_			quest Hea						
Environment No En	nvironment 👻			_			isponse H	eaders					
-							isic Auth	Success					
Iterations 1							wk Auth						
Delay 0	ms				⊧ G	SET OA	Auth1.0 (v	erify sigr	nature)				
Data S	elect File						t Cookies						
Save re	isponses 🚯						rt Cookies						
	ariable values 🕕						elete Cook Isponse St		de				
	llection without using stored cookies			_			reamed R						
Save co	ookies after collection run 👔						alay Respo						
R	Run Postman Echo				⊧ G		rt UTF8 En		Response				
				_			ip Compr						
							flate Com	pressed	d Respons	e			
ction Runner Run Results				il My	Works	space •							
) (7 Postman	Echo No Environment											Ran Summary 🔸	Export Res
GET Object representation http	ps://postman-echo.co., / Utilities / Date and Time / Object							• 200	0.0K 🖷 11	4 ms 🔘 298			
response is ok													
response json should retur     GET Before comparisons hittos									о ок 🖷 👓	2 ms 🔘 226	8		
CET Before companions inteps     response is ok	A REPORT OF A REPORT OF A REPORT OF A REPORT OF							J 10		AD			
response json should say ti													
GET After comparisons https://	/ Utilities / Date and Time / After c							200	0 OK 🔘 12	1 ms 🌒 354	8		
response is ok	imestamp is not after target												
GET Between timestamps http								• 203	0 OK 🌒 10	9 ms 🔘 229	•		
response is ok													
	imestamp is not between start and end ostman-echo.co							• 200	0 OK 🌒 13	3 ms 🔘 348	8		
response is ok													
response json should say th													
	n format vf. to v2 https://postman-echo.co / Utilities / Postman Cole Error: expected response to have status code 200 but got 404	schon / Tr						• 404	4 Not Found	107 ms	1158		
	items   Assertionfirmer: expected undefined in response to contain property	'info.schema'											
	n format v2 to v1 https://postman-echo.co / Utilities / Postman Colle	ction / Tr						• 40	4 Not Found	116 ms	241.8		
	Error: expected response to have status code 200 but got 404 Items   AssertionError: expected undefined in response to contain property '	Tolders'											
	nems   Asserbanerror: expected underined in response to contain property: c//postman-echo.co							401	1 Unauthorize	id 🔵 106 mi	<ul> <li>350 B</li> </ul>		
response code is 401													

The runner shows the overall count for the number of passed and failed tests. We can also see the assertion error on failed tests; in this case, saving the response (setting the proper flag above) can help us better understand what is happening.

1

Running the tests can also be done from the command line or from within Jenkins (or any other CI/CD tool). This can be achieved using Newman.

In order to run Newman, we need to provide a path or a URL to our collection.

In this case, we'll obtain a public link to it.

			FUSUITALI LUTU	
History	Collections	APIs	API This col	
+ New Collection		Trash	Share Run	
Postman Echo      A     37 requests		•	Documentation	
<ul> <li>Request Methods</li> </ul>	A Share	Collection	Earn how to do	
GET GET Request POST POST Raw Text	A Manag	ie Roles ne ≇E	Postman Echo is serv clients and make san	
SHARE POSTMAN ECHO				×
To a workspace Emb	ed Get public link			
	omatically when changes are m	ade to the collection. Check	a complete list of your public collection links from your Postman Dashboard. out the <u>Postman Pro API</u> to access your synced collections.	

Then we need to decide which Newman reporter to use. Newman provides a built-in JUnit reporter; however, better alternatives exist such as junitxray or ju nitfull.

#### Which Newman reporter should I use?

The standard Newman junit reporter produces <testcase> entries in the JUnit XML report that can be misleading as tests will be identified on the Postman test description, which can be similar between different tests (e.g. "response is ok").

Therefore, two alternative reporters arise: newman-reporter-junitxray and newman-reporter-junitfull

"newman-reporter-junitxray" (simply known as "junitxray"), will create <testcases> per each request, which in the end will lead to corresponding Test issues in Xray. This means that there won't be explicit visibility for each Postman test on that request, as they will be treated just as one.

"newman-reporter-junitfull" (simply known as "junitfull"), on the other hand, will produce one <testcase> per each Postman test, which will lead to the same number of corresponding Test issues in Xray.

If you aim just to have high-level overview of the request, then "junitxray" reporter will be preferable; otherwise, "junitfull" may be a better option.

	junit	junitxray	junitfull
tests	<ul> <li>40 Tests (one per each PM test name /description)</li> </ul>	37 Tests (one per request)	90 Tests (one per each PM test)
generic definition field	<collection>.<pm_test_description></pm_test_description></collection>	<collection>.<request_name></request_name></collection>	<folder_path>/<request_name>. <pm_test_description></pm_test_description></request_name></folder_path>
neid	"PostmanEcho.response is ok"	"PostmanEcho.Object representation"	"Utilities / Date and Time / Object representation.response is ok"
notes	<ul> <li>leads to a collision of tests made for different requests</li> <li>ignores folder path, which can lead to the collision of requests having the same name</li> <li>one Test issue per each PM test</li> </ul>	<ul> <li>ignores folder path, which can lead to the collision of requests having the same name</li> <li>doesn't present the multiple PM tests</li> <li>few Tests, one per each request</li> </ul>	<ul> <li>can lead to many Test issues</li> <li>one Test issue per each PM test, identified by the full (folder) path of the request</li> </ul>

Installing Newman and its reporters

npm	install	-g n	ewmai	ı	
# ir	nstall or	ne of	the	following	ones
npm	install	-g n	ewmai	n-reporter-	-junitxray
npm	install	-g n	ewmai	n-reporter-	-junitfull

Whenever running Newman, we can specify one or more reporters (if we want to), including a CLI friendly one.

newman run https://www.getpostman.com/collections/c7334a5cf52a90639a48 -r 'cli,junitfull,junitxray' --reporterjunitfull-export postman\_echo\_junitfull.xml --reporter-junitxray-export postman\_echo\_junitxray.xml -n 1

If using Jenkins, we need to configure a build step to execute "newman" command.

Build	
Execute she	AL X
Command	newman run https://www.getpostman.com/collections/c7334a5cf52a90639a48 -r 'cli,junitfull,junitxray'reporter-junitfull-export postman_echo_junitf

Importing results is as easy as submitting them to the REST API with a POST request (e.g. curl), or by using one of the CI plugins available for free (e.g. Xr ay Jenkins plugin).

The following screenshots show the Jenkins configuration.

We could eventually fill/identify the Test Environment to associate to the Test Execution based on the Postman's Environment being used if it would make sense for us to analyze the results on a per-environment basis.

Post-build Ac	tions	
Xray: Results	Import Task	
Jira Instance	xray-vm	
Format	JUnit XML	
Parameters	Import to Same Test Execution	0
		When this option is check, if you are importing multiple execution report files using a glob expression, the results will be imported to the same Test Execution
	Execution Report File (file path with file name)	postman_echo*.xml
	Project Key	CALC
	Test Execution Key	
	Test Plan Key	
	Test Environments	
	Revision	
	Fix Version	

A Test Execution will be created containing results for all tests executed. Actually, in our specific case and only for demonstration purposes, two Test Executions would be created due to the fact that we're generating two JUnit XML files from the different Newman reporters.

Unstructured (i.e. "Generic") Test issues will be auto-provisioned the first time you import the results, based on the identification of the test (see notes for possible Newman reporters above). The "Generic Definition" field on the Test issue is used as a way to uniquely identify the test.

(î)

D	Please	note											
	Tests will be reused on subsequent result imports as long as you don't change what contributes to the calculation of the test's unique identifier (i. e. "Generic Definition" field); otherwise, new Tests will be auto-provisioned.												
	Therefore, and depending on the Newman reporter being used, if you change the Postman test description or the folder containing the test, it will lead to new Tests in Jira as Xray will consider them to be new.												
					and relate	d Tests	) create	ed as a cor	nsequence of im	porting	the	Init XML report prod	luced by the
vmar	n report	er newma	an-reporter-jun	itxray.									
	Iculator / C/		nostman ech	o_junitxray.xml	- [150787	055026	3/11						
Edit	Q Comme		-	gress Resolve Issue		Admin ~	J-+]						
etails	Q Comme	nt Assign	Wore - Start Pro	gress Resolve issue	Close Issue	Admin							
ype:		Test Exect	ution		Status:		OPEN (	View Workflow)					
riority:		Major			Resolut		Unresol	lved					
ffects \ ompon	/ersion/s:	None None			Fix Vers	sion/s:	None						
ompon	ent/s.	None											
abels:													
	ironments:	None											
est Env est Plar	1:	None											
est Env est Plar escript	n: tion	None None	ernal source										
est Env est Plar escript	n: tion	None	ernal source										
est Env est Plar escript	n: tion	None None	ernal source										
est Env est Plar escript xecutio	n: tion	None None	ernal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests	n: tion n results imp	None None Norted from exte	arnal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests	n: tion	None None Norted from exte	ernal source							+ Ad	id Y		
est Env est Plar escript xecutio ests verall E	n: tion n results imp	None None Norted from extension	arnal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests verall Ex	n: n results imp xecution Statu PASS 5	None None Norted from extension	ernal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests verall Es 32 p otal Test	1: ion n results imp xecution Statu PASS 5	None None Norted from extension	ernal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests verall Es 32 p otal Test	n: n results imp xecution Statu PASS 5	None None Norted from extension	ernal source							+ Ad	ld Y		
est Env est Plar escript xecutio ests verall Es 32 p otal Test	1: ion n results imp xecution Statu PASS 5	None None Norted from extension	ernal source						Show 100 • entries	+ Ad			
est Env est Plar escript ests verall E) 32 s stat ests	1: ion n results imp xecution Statu PASS 5	None None Norted from extension	ernal source		Test Type	#Req	#Def	Assignee	Show 100 ♥ entries				
est Env est Plar escript ests verall E) 32 s stat ests	n: ion n results imp execution Statu PASS 5 ts: 37 F Filter(s)	None None orted from exte s			Test Type Generic	#Req 0	#Def 0	Assignee Administrator					
est Env est Plar escript ests verall E 32 t s a t a T s a	n: ition results imp results imp results imp results imp results: 37 Filter(s) Rank	None None orted from extenses s FAIL	Summary	from format v1 to v2						Column	15 🕶		
est Env est Plar escript xecutio ests verall E) 32 p statement sta	n: tion n results imp eccution Statu PASS 5 Filter(s) Rank 1	None None orted from extr s FAIL Key CALC-7864	Summary Time addition		Generic	0	0	Administrator	Status	Column	15 -		
est Env est Plar rescript xecutio ests verall E> 32 s s tal Test = ~	n: results imp Accution Statu PASS 5 Filter(s) Rank 1 2	None None orted from extr s FAIL Key CALC-7864	<ul> <li>§ Summary</li> <li>Time addition</li> <li>Transform collection</li> </ul>		Generic	0	0	Administrator Administrator	Status PASS FAIL	Column			

Within the execution screen details, you can look at the Test Run details which include the duration, overall result, and also any eventual error message.

Calculator / Test Execution: CALC-7871 / Test: CALC-7834 Object representation		Export Test as Text	Return to Test Execution	xecute with Exploratory App	
Execution Status PASS end PASS Execution Status PASS PASS PASS PASS PASS PASS PASS PAS				Assignee: Administrator Executed By: Administrator Tests - environments:	Versions: - Revision: -
Affected Requirements					^
None					
Comment Preview Comment V	Execution Defects (0) Create	Defect Create Sub-Task Add Defects 🗸	Execution Evidence (0)		Add Evidence 🗸
Execution Details					
•					
Test Description					^
Custom Fields					^
There are no Test Run Custom Fields defined.					
Test Details					^
Test Type: Generic					
Definition: PostmanEcho.Object representation					
Results					^
Context	Output			Duration	Status
TestSuite c8c16569-a771-4ad3-874c-9072351ff79a - Utilities / Date and Time	-				PASS

			r <b>esults if I used "newma</b> vman-reporter-junitfull", yo				nore Te	est issues	s as seen ahea	d.					
			would have the same Su								otion.				
	Iculator / C		s - postman_echo_junitfull.	.xml - [159787	95273	3621									
🖋 Edit	Q Comm		ign More - Start Progress Resolv												
✓ Details															
Type: Priority: Affects V Compone Labels: Test Envi Test Plar	ent/s: ironments:	I Test €	Execution	Status Resolu Fix Ve			OPEN (Vie Unresolved None	w Workflow) I							
<ul> <li>Descript</li> <li>Execution</li> </ul>		norted from	external source												
	Tresures ini	Joined Hom	external source												
✓ Tests										+ •	dd ¥				
Overall Ex	xecution Stat	us													
82	pass 8														
		FAIL													
Total Test	Filter(s)														
v									Show 100 🗸 entries	Colum					
C.,	A Rank	🔶 Key	Summary	¢ Tes	t Type #	Req	#Def	Assignee	Status	Colum	115 •				
	1	CALC- 7743	the "foo1" cookie has correct value	Gene	ric 0	1	0	Administrator	PASS	►	]				
	2	CALC- 7820	response is ok	Gene	ric 0	1	0	Administrator	PASS	►	]				
	3	CALC- 7787	response is ok	Gene	ric 0	1	0	Administrator	PASS	►					
	4	CALC- 7742	response body has json with form data	Gene	ric 0	1	0	Administrator	PASS	►	]				
	5	CALC- 7786	the "foo1" cookie has correct value	Gene	ric O		0	Administrator	PASS	►	]				
	/ Test Exe Ise is ok		C-7741 / Test: CALC-7790					Q	Export Test as Text	🔺 Return	to Test Exe	ecution	Execute with	Exploratory App	Previo
	on Status		다 이 이 이 이 18/Aug/20 1:										Executed	nee: Administra I By: Administrate ests - ints:	
	ed Require	ments													
None Co	omment		Preview Cor	mment V Executio	on Defect	:s (0)	Create	Defect Create	Sub-Task Add Defects	Exe	cution Ev	idence (	0)		,
	Executio	n Details													
D E	escription														
Test D															
Test D None															
Test D None Custor	<b>m Fields</b> are no Test Ru	ın Custom Fie	lds defined.												
Test D None Custor There a		ın Custom Fie	lds defined.												
Test D None Custor There a Test I	are no Test Ru <b>Details</b> : Type:	Generic	:												
Test D None Custor There a Test I	are no Test Ru Details	Generic		se is ok											
Test D None Custor There a Test I	are no Test Ru <b>Details</b> : Type: nition:	Generic	:	se is ok											
Test D None Custol There a Test I Defi	Details : Type: nition: Its Context	Generic Utilities	:	se is ok	Output									Duration 96.000 ms	Statu

Tips

- · After importing results, you can link Test issues to existing requirements or user stories, so that you can track coverage on real-time directly on them
- ٠ You can map Postman's environment to Xray's Test Environment concept on Test Executions if you want to have visibility of the results on a perenvironment basis
- Multiple iterations/executions can be linked to an existing Test Plan, whenever importing the results
- If you run the tests multiple times with "newman -n <number\_of\_iterations>" parameter, multiple entries will appear within the Results section of the Test Run execution screen details

## References

- Postman
- Postman SDK
- Postman Echo API
- Using Newman

- newman-reporter-junitxray
  newman-reporter-junitfull
  Postman Quick Reference Guide