RESTful API Documentation for ToDo Sample App

Introduction

To access ToDo RESTful API, an HTTP connection should be established using an HTTP header with these parameters on each request:

HTTP header	**Description**
Content-Type	Should have the content "*application/json*"
HTTP Method	Entries can be POST, GET, PUT, or DELETE
URL	http://hostname/todo/api/v1.0/tasks

Minimal script to access API

curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks

This is a proof of Concept at this time. The -u miguel:python is the username and password that's been hardcoded into the tutorial API (https://gist.github.com/miguelgrinberg/5614326); the real world version of this will authenticate against a database of users and passwords.

Example Output

HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 559
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3
Date: Mon, 03 Aug 2020 19:21:06 GMT


```
{
  "tasks": [
    {
```

```
    "description": "Milk, Cheese, Pizza, Fruit, Tylenol",
    "done": false,
    "title": "Buy groceries",
    "uri": "http://localhost:5000/todo/api/v1.0/tasks/1"
   },
   {
    "description": "Need to find a good Python tutorial on the web",
    "done": false,
    "title": "Learn Python",
    "uri": "http://localhost:5000/todo/api/v1.0/tasks/2"
   },
   {
    "description": "",
    "done": false,
    "title": "Read a book",
    "uri": "http://localhost:5000/todo/api/v1.0/tasks/3"
   }
  ]
}
```


It should print the server response. In this case, a successful output will be an HTTP Error Code 200 OK, followed by the data that was requested (for a GET request).

Endpoints

These sections will describe API endpoints. Endpoints are the URI where you should point your requests to fetch, update or create data. Each endpoint has an HTTP verb, a URI and (maybe) parameters. For example, to list all tasks you need to use the GET verb, URI and no parameters, using the sample script in the section above this can translated to:

curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks

Methods

HTTP Method	URL	Output
GET This gets ALL Tasks	curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks	HTTP/1.0 200 OK **Content-Type**: application/json **Content-Length**: 559 **Access-Control-Allow-Origin**: * **Server**: Werkzeug/1.0.1 Python/3.7.3 **Date**: Mon, 03 Aug 2020 19:21:06 GMT {

		"tasks": [
		{
		"description": "Milk, Cheese, Pizza, Fruit, Tylenol",
		"done": false,
		"title": "Buy groceries",
		"uri": **"http://localhost:5000/todo/api/v1.0/tasks/1"**
		},
		{
		"description": "Need to find a good Python tutorial on the web",
		"done": false,
		"title": "Learn Python",
		"uri": "http://localhost:5000/todo/api/v1.0/tasks/2"
		},
		{
		"description": "",
		"done": false,
		"title": "Read a book",
		"uri": "http://localhost:5000/todo/api/v1.0/tasks/3"
		}
		]
		}

GET
 	
This gets only the Task with an ID of 1
	curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks/**1**	HTTP/1.0 200 OK
		Content-Type: application/json
		Content-Length: 142
		Access-Control-Allow-Origin: *
		Server: Werkzeug/1.0.1 Python/3.7.3
		Date: Mon, 03 Aug 2020 19:30:39 GMT
		
		{
		"task": {
		"description": "Milk, Cheese, Pizza, Fruit, Tylenol",
		"done": false,
		"id": 1,
		"title": "Buy groceries"
		}
		}

<table>
<tr>
<td>POST

This method allows you to add a Task to the list.

NOTE: ID field is auto generated, Description is an optional field, and Done is defaulted to false. Title is the only required field and MUST contain valid JSON.</td>
<td>curl -u miguel:python -i -H "Content-Type: application/json" -X POST -d '{"title":"Write documentation."}' http://localhost:5000/todo/api/v1.0/tasks</td>
<td>HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 114
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3
Date: Mon, 03 Aug 2020 19:37:44 GMT

{
"task": {
"description": "",
"done": false,
"id": 4,
"title": "Write documentation."
}
}</td>
</tr>
<tr>
<td>PUT

This method allows you to update an existing task.

NOTE: the only JSON you have to add is the portion that you are actually changing.

Constraints:
● Length must be greater than 0
● Content must be JSON
● Content must be Unicode
● Done must be a Boolean (true | false)</td>
<td>curl -u miguel:python -i -H "Content-Type: application/json" -X PUT -d '{"done":true}' http://localhost:5000/todo/api/v1.0/tasks/4</td>
<td>HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 113
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3
Date: Mon, 03 Aug 2020 19:39:53 GMT

{
"task": {
"description": "",
"done": true,
"id": 4,
"title": "Write documentation."
}
}

After using the PUT method:

Verifying our update:
curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks/4

HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 113
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3</td>
</tr>
</table>

		Date: Mon, 03 Aug 2020 19:44:54 GMT
		
		{
		  "task": {
		  "description": "",
		  "done": true,
		  "id": 4,
		  "title": "Write documentation."
		  }
		}

DELETE
 	
This method allows you to delete a task.
	curl -u miguel:python -i -X DELETE http://localhost:5000/todo/api/v1.0/tasks/3	HTTP/1.0 200 OK
		Content-Type: application/json
		Content-Length: 21
		Access-Control-Allow-Origin: *
		Server: Werkzeug/1.0.1 Python/3.7.3
		Date: Mon, 10 Aug 2020 11:56:58 GMT
		
		{
		  "result": true
		}
		
		To verify the deletion:
		$ curl -u miguel:python -i http://localhost:5000/todo/api/v1.0/tasks
		
		HTTP/1.0 200 OK
		Content-Type: application/json
		Content-Length: 407
		Access-Control-Allow-Origin: *
		Server: Werkzeug/1.0.1 Python/3.7.3
		Date: Mon, 10 Aug 2020 11:57:03 GMT
		
		{
		  "tasks": [
		  {
		  "description": "Milk, Cheese, Pizza, Fruit, Tylenol",
		  "done": false,
		  "title": "Buy groceries",
		  "uri": "http://localhost:5000/todo/api/v1.0/tasks/1"
		  },
		  {
		  "description": "Need to find a

		good Python tutorial on the web", "done": false, "title": "Learn Python", "uri": "http://localhost:5000/todo/api/v1.0/tasks/2" }] }

Server Responses

All endpoints should return one of these HTTP status codes depending on how the operation succeeded.

Code	HTTP meaning	Description
200	OK	The request was successful.
201	Created	The request was successful and a resource was created.
400	Bad Request	The request could not be understood or was missing required parameters.
401	Unauthorized	Authentication failed.
403	Forbidden	User does not have permissions for the requested operation.
404	Not Found	Resource was not found.
422	Unprocessable Entity	The request does not include all of the required params or the params does not match its type or any validation failed.

Bad Request response example

In case of bad requests, regardless of the HTTP status code, the ToDo API response includes JSON in the message body describing the error.

example:

{

"error": "Not found"

}

Another error type that is common is a 401 UNAUTHORIZED.

curl -i -H "Content-Type: application/json" -X POST -d '{"title":"Write documentation."}' http://localhost:5000/todo/api/v1.0/tasks

This returns the following:

HTTP/1.0 401 UNAUTHORIZED
Content-Type: application/json
Content-Length: 37
WWW-Authenticate: Basic realm="Authentication Required"
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3
Date: Mon, 03 Aug 2020 19:33:27 GMT

```
{
  "error": "Unauthorized access"
}
```

vs.

curl **-u miguel:python** -i -H "Content-Type: application/json" -X POST -d '{"title":"Write documentation."}' http://localhost:5000/todo/api/v1.0/tasks

Success request example

$ curl -u miguel:python -i -H "Content-Type: application/json" -X POST -d '{"title":"Read a book"}' http://localhost:5000/todo/api/v1.0/tasks

Response 201 (application/json)

HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 105

Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.1 Python/3.7.3
Date: Mon, 03 Aug 2020 19:09:56 GMT


```
{
  "task": {
    "description": "",
    "done": false,
    "id": 3,
    "title": "Read a book"
  }
}
```


Examples by Programming Language

JavaScript

HTTP Method	Code Sample

GET	const url = 'http://localhost:5000/todo/api/v1.0/tasks'; fetch(url) .then((resp) => resp.json()) .then(function(data) { API_result = JSON.stringify(data);
	
	document.getElementById('tasks').innerHTML = API_result; }) .catch(function(error) { console.log(error); });

POST	
PUT	
DELETE	

Python

HTTP Method	Code Sample
GET	
POST	
PUT	
DELETE	

C#

HTTP Method	Code Sample
GET	
POST	
PUT	
DELETE	