

## How to drive rest using Postman and phpMyAdmin

The rest module should download with the following “vendor” dependencies:

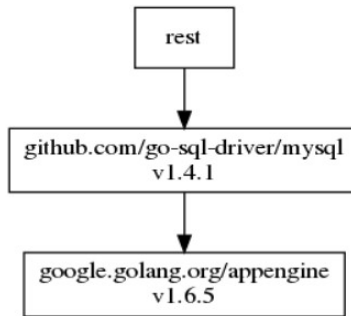
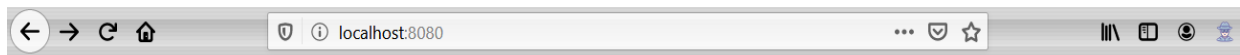


Figure 1 Dependency output from dep

Download and build as follows:

```
$ svn export https://github.com/rory911/misc/trunk/rest
$ cd rest
$ go build rest
$ rest
```

You might have to give firewall permission in order to continue. Now, using a web browser, navigate to the home page at `localhost:8080` as follows:



### Process SQL records with standard net/http services

This simple Golang program uses the following JSON records to feed the SQL database:

```
ID: string
Title: string
Desc: string
```

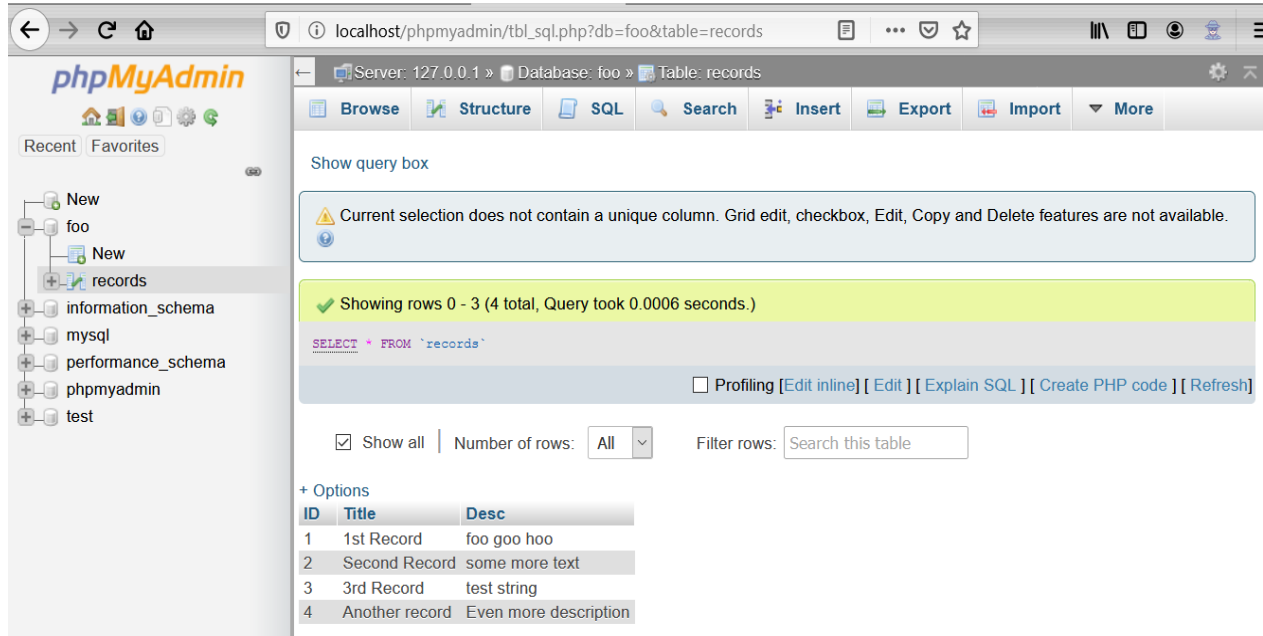
Navigate to [mysql-records](#) and use the RESTful interface for building and modifying the SQL database. Postman and phpMyAdmin are recommended for driving the interface with POST, GET, PUT, PATCH, and DELETE. Experience with URL query strings is required with Postman. Refresh mysql-records after each Postman method finishes. Use the GET method with Postman, or send SQL commands from phpMyAdmin to display the SQL database.

Before navigating to `localhost:8080/mysql-records`, use phpMyAdmin to create a database named, `foo` with a table named `records`. You can optionally add a JSON record to the mysql database.

Please be aware that the standard net/http services do not handle URL variables like Gorilla MUX. Postman will need fields, or table columns, to be passed inside a query string instead. Please note the following:

- POST (/mysql-records) requires the JSON fields only
- GET (/mysql-records) only requires that the database was created
- PUT (/mysql-records/query) requires both title and description in the query string
- PATCH (/mysql-records/query) requires either title or description in the query string
- DELETE (/mysql-records/query) requires the id in query string

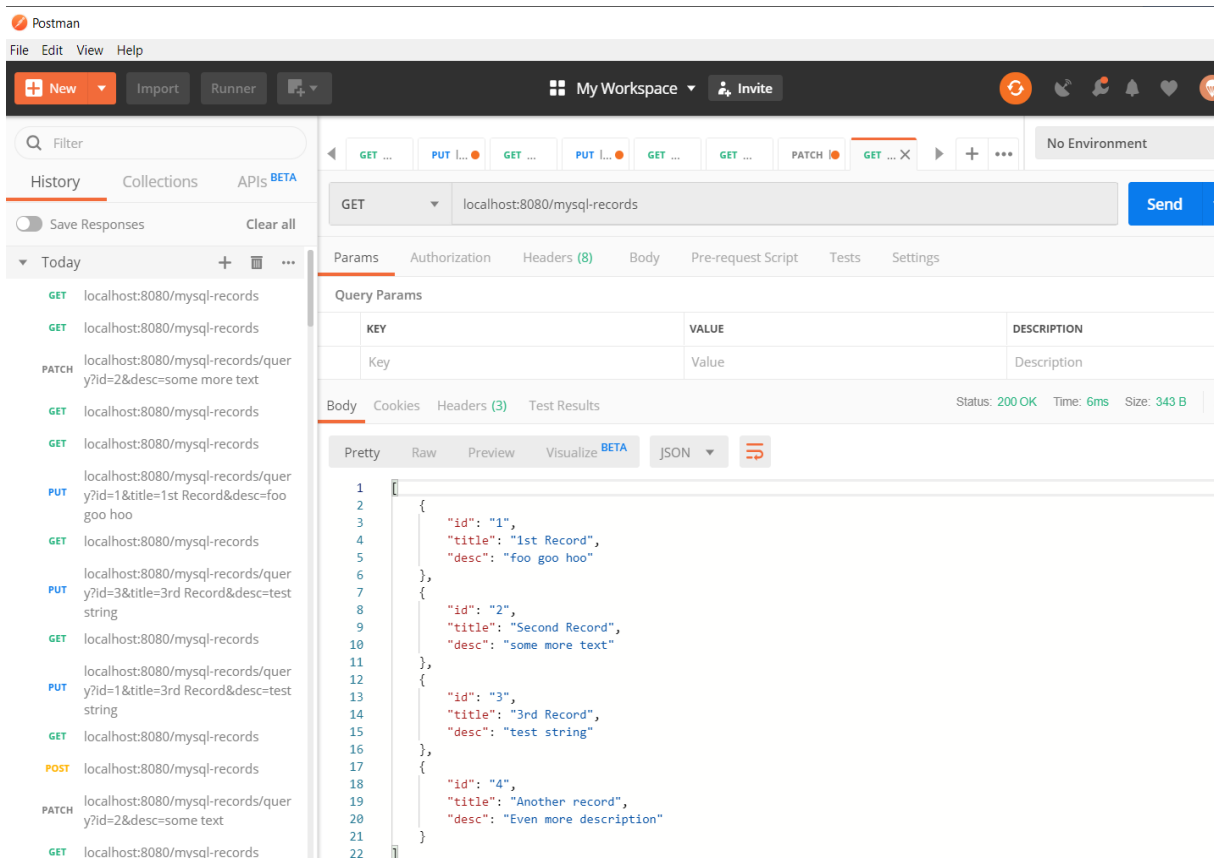
Now, let's have a look at a sample MySQL database using phpMyAdmin:



The screenshot shows the phpMyAdmin web interface. The left sidebar displays a database structure with 'foo' as the selected database, containing a 'records' table. The main panel shows the 'records' table with 4 rows. A message at the top states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.' Below this, a green bar indicates 'Showing rows 0 - 3 (4 total, Query took 0.0006 seconds.)'. The SQL query 'SELECT \* FROM `records`' is shown. The table data is as follows:

ID	Title	Desc
1	1st Record	foo goo hoo
2	Second Record	some more text
3	3rd Record	test string
4	Another record	Even more description

and from Postman:



The screenshot shows the Postman application interface. The left sidebar shows a list of requests. The main panel shows a GET request to 'localhost:8080/mysql-records'. The response is a JSON array of 4 records, displayed in the 'Body' tab. The status is '200 OK', time is '6ms', and size is '343 B'.

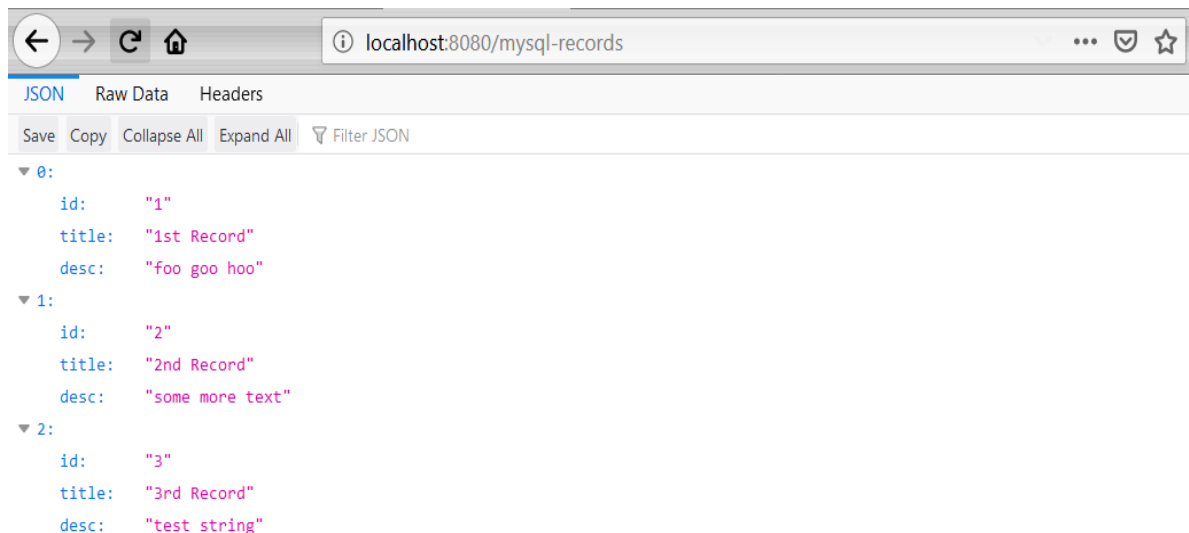
```
1 {
2   {
3     "id": "1",
4     "title": "1st Record",
5     "desc": "foo goo hoo"
6   },
7   {
8     "id": "2",
9     "title": "Second Record",
10    "desc": "some more text"
11  },
12  {
13    "id": "3",
14    "title": "3rd Record",
15    "desc": "test string"
16  },
17  {
18    "id": "4",
19    "title": "Another record",
20    "desc": "Even more description"
21  }
22 }
```

Let's delete record(id=4) and modify record(id=2) using Postman:



The image shows two Postman request entries. The first entry is a DELETE request to the URL `localhost:8080/mysql-records/query?id=4`. The second entry is a PATCH request to the URL `localhost:8080/mysql-records/query?id=2&title=2nd Record`. Both requests have a 'Send' button next to them.

Then, instead of Postman or phpMyAdmin, look at the JSON output in the browser:



in order to illustrate that all three options will display the MySQL database.

It's a trivial SQL database, but it illustrates the very basics of MySQL, JSON, Postman, phpMyAdmin, Apache, and Golang with net/http RESTful API.