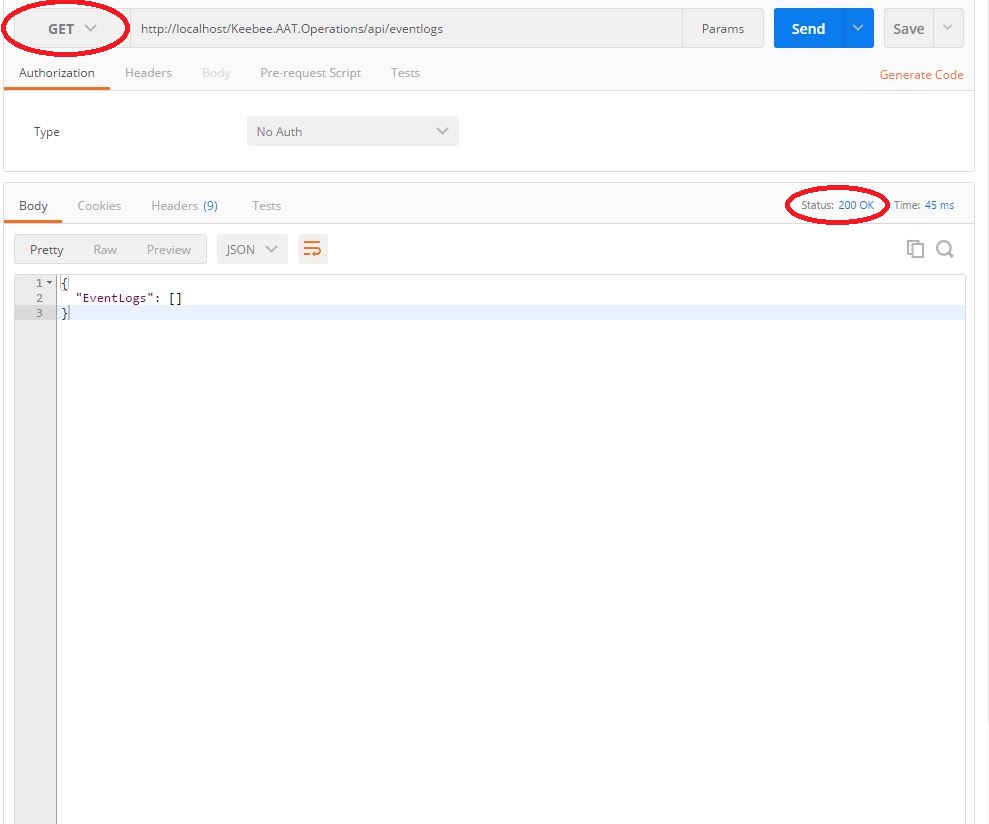
Examples of RESTful API Calls

**GET**

Perform a **GET** when there are no records in the table.

Notice with a GET there is only a *response* body and no *request* body. A status code is also returned. Anything in the 200 range indicates “success”.



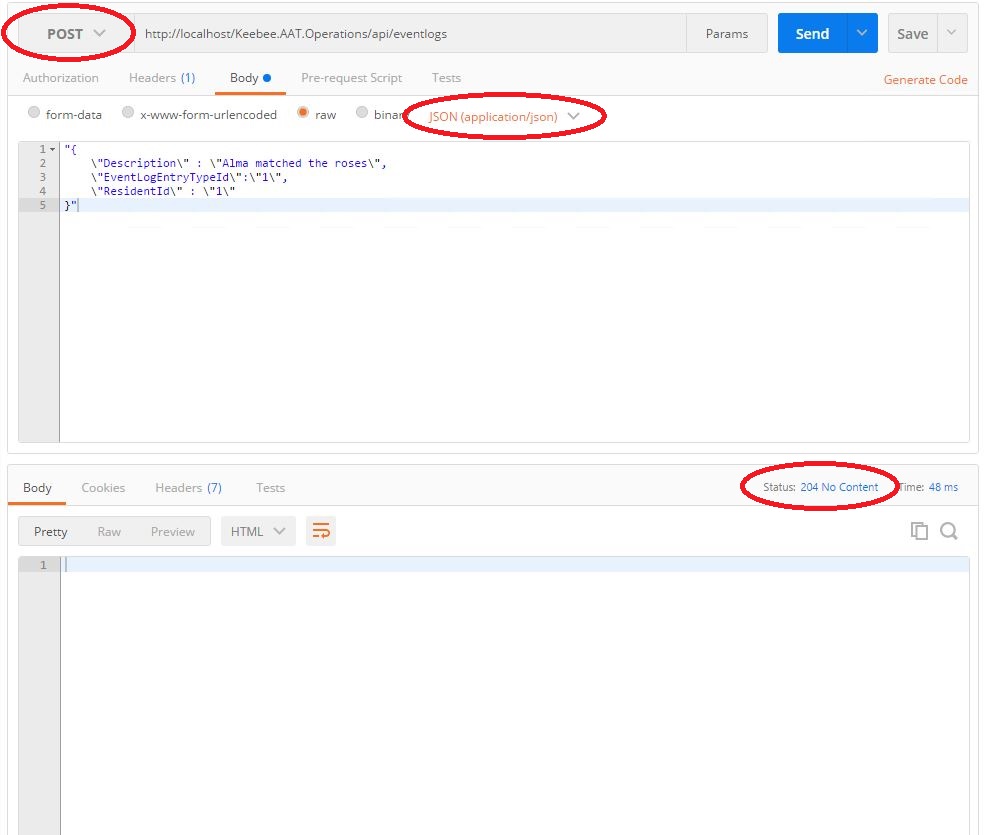
**POST**

Perform a **POST** to add a record to the table.

For a POST, the *request* body specifies the data to insert. It is just straight text, but it needs to conform to the JSON format. You can see that the *ContentType* is “application/json”. The backslash is an escape character to indicate that the double quotes need to be included in the request.

EventLogEntryTypeId = 1 simply means “Game”. It’s obviously not important for the sake of this example, but just in case you were wondering. ResidentId = 1 means “Alma”.

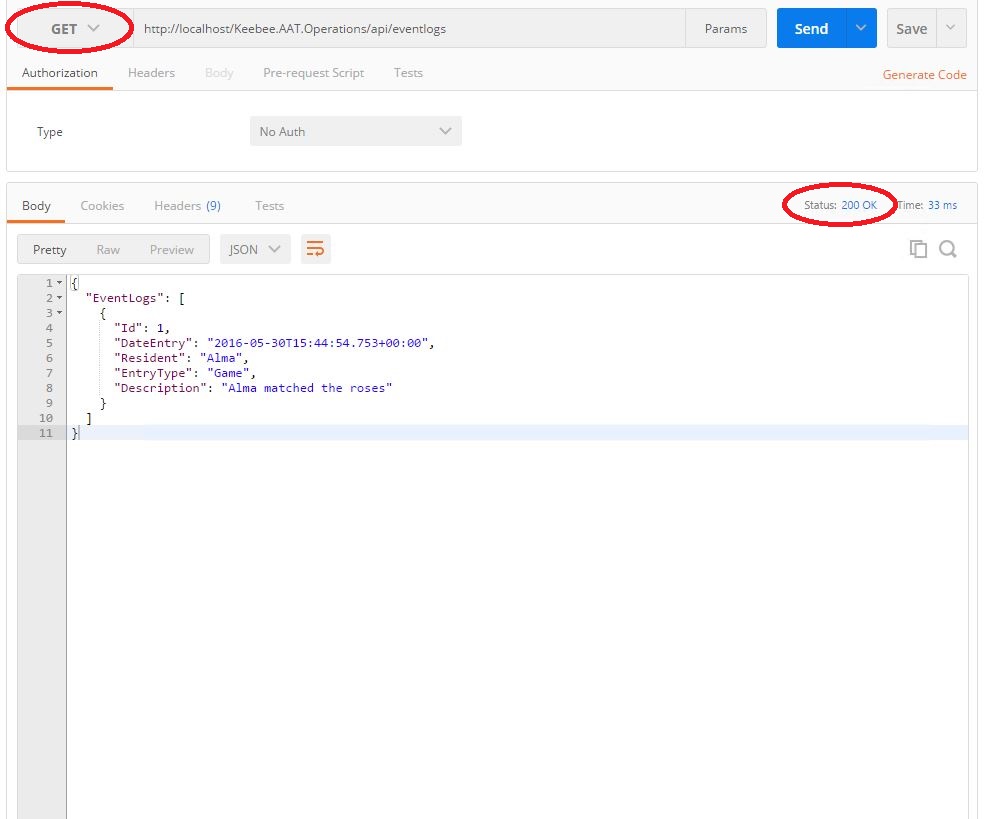
The returned status code is 204 (i.e. “success”) and the *response* body contains no content.



**GET**

Perform a **GET** after a record has been added to the table.

Again, only a *response* body (this time with data) and no *request* body. We also have a return status code of 200 (Yay!).

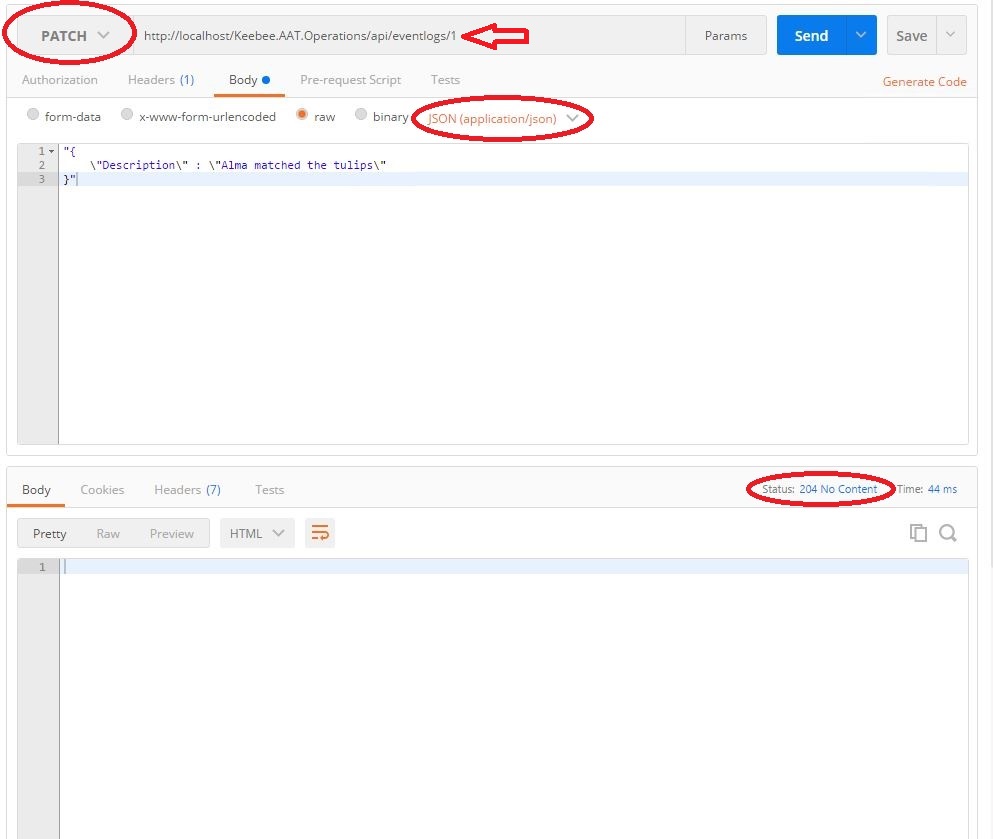


**PATCH**

Perform a **PATCH** to update a record in the table.

For a PATCH, the *request* body specifies the data we want to update. This time however we need to specify the Id of the target record. You can see the url now has the suffix “/1” which means update the EventLog table for EventLogId = 1. This will update the Description column (if for example we want to change the word “roses” to “tulips”). Notice again that the *ContentType* is “application/json”.

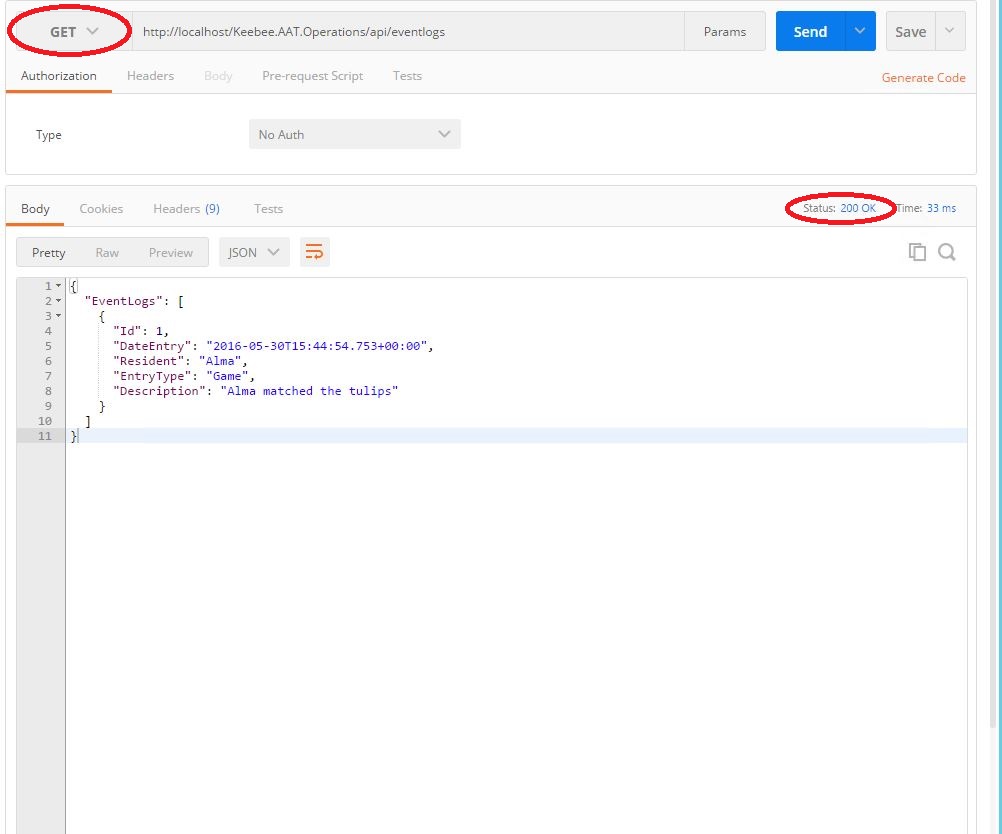
The returned status code is 204 (Wicked!) and the *response* body contains no content.



**GET**

Perform a **GET** after a record has been updated.

You know all about GETs now so no need to dwell on it. ☺ You can see that the Description column has been changed from “Alma matched the roses” to “Alma matched the tulips”.

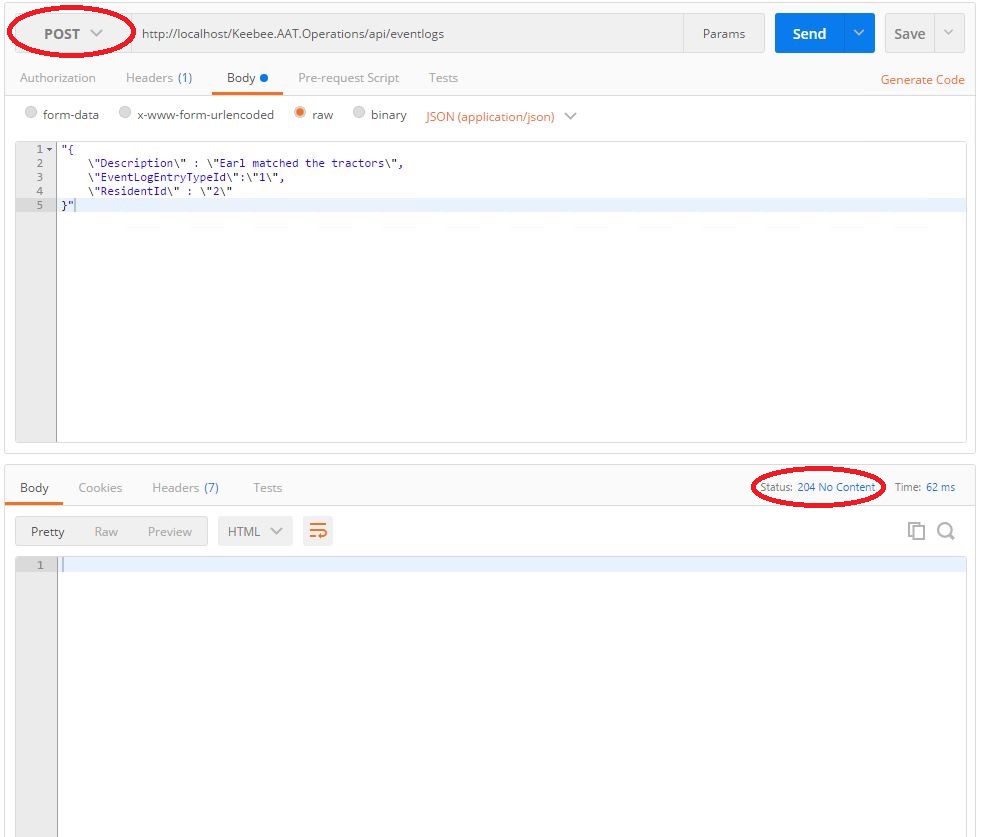


**POST**

Perform a **POST** to add another record to the table.

Let’s perform a POST to indicate that Earl is now playing a game and successfully matched the tractors. Notice that ResidentId = 2 in this case for “Earl”.

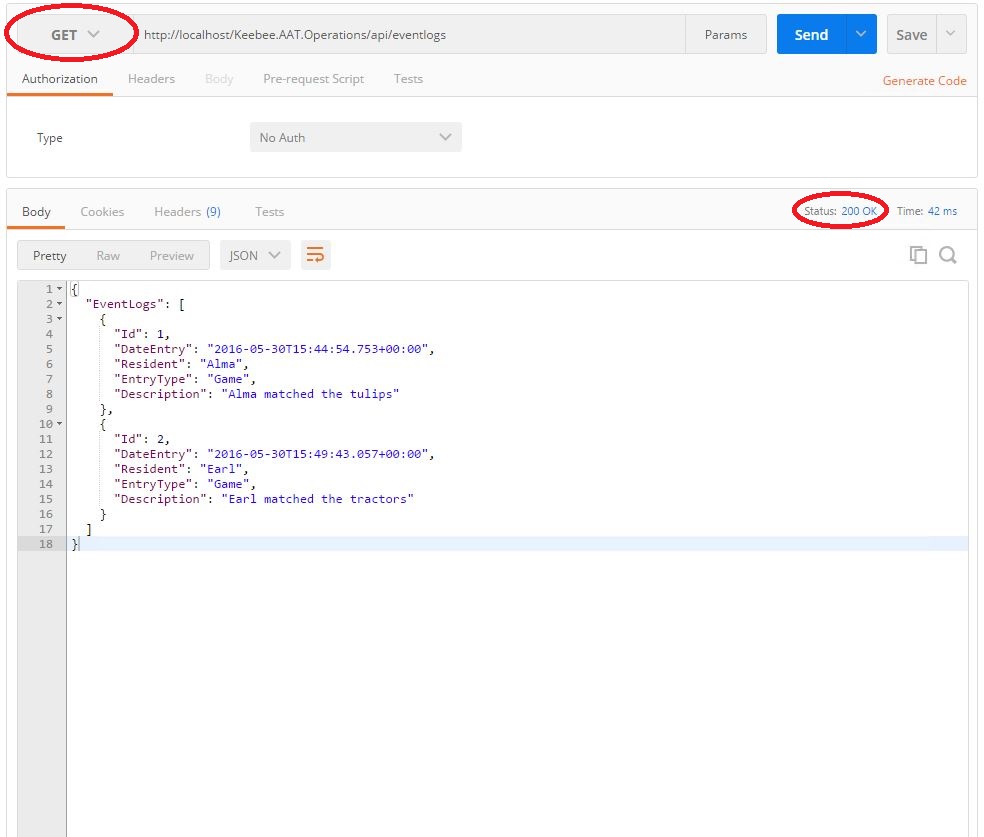
Again the returned status code is 204 (Yippee!) and the *response* body contains no content.



**GET**

Perform a **GET** after another record has been added to the table.

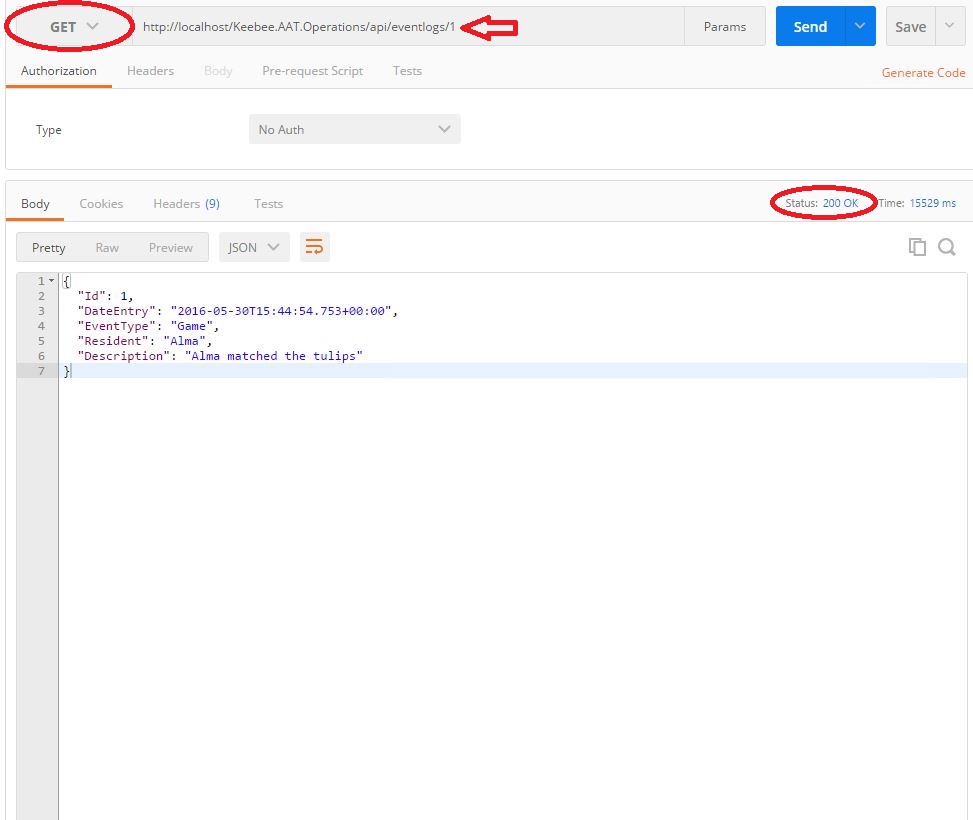
We now see the response contains the record for Earl that we added above, and the returned status code is 200 (Hooray!).



**GET**

Perform a **GET** on a single entity by specifying its Id.

Normally of course it wouldn’t be sensible to GET every record in the EventLog table. So to request a single record, simply add the suffix “/1” and that will return the EventLog record for EventLogId = 1.

****

Obviously we’re going to need more functionality in the API, like if we wanted to see only the EventLogs for Earl over a given date/time range. In that case the url might look something like:

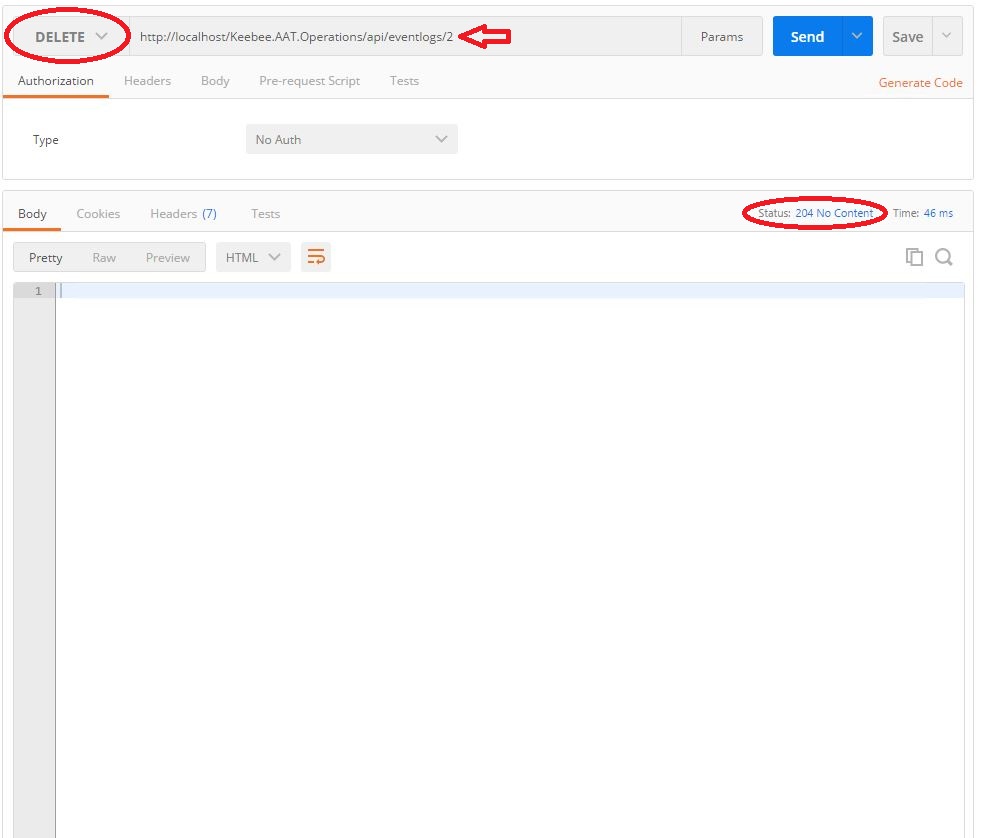
http://localhost/Keebee.AAT.Operations/api/eventlogs/residents/2/?From=’05/16/2016’&To=’05/17/2016’

**DELETE**

Perform a **DELETE** to remove a record from the table.

Last but not least is DELETE. This will delete the EventLog record corresponding to EventLogId = 2 (notice the url contains the suffix “/2”). A DELETE is similar to a GET in that there is no *request* body.

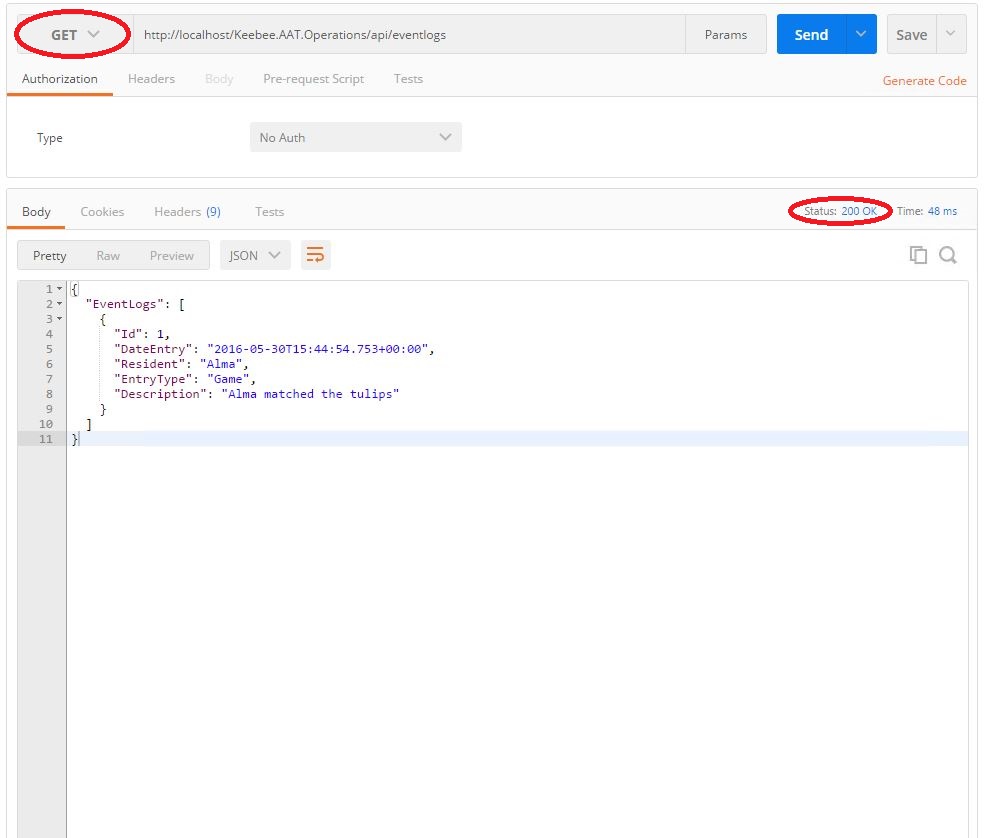
The response code is 204 which means it’s Miller time!



**GET**

Perform a **GET** after we have deleted a record from the table.

We now see that the record for Earl has been removed and the returned status code is 200 which means…well you know. ☺



**GET**

This is just to demonstrate that before all those soothing and gratifying 200 status codes start to arrive, there is a development period (hopefully short lived) where the calls aren’t formed correctly or something is wrong in the API.

Here is a typical *404 (Not Found*) response code. Theoretically this call should have returned all the EventLogs for ResidentId = 1, but’s it’s not done yet. ☺

