# Exercise: API Testing with Postman

This document defines the exercise assignments for   
the ["Back-End Technologies Basics"](https://softuni.bg/trainings/4726/back-end-technologies-basics-september-2024) Course @ SoftUni.

## Create a Trello account

In the upcoming exercise we will use the **API provided by Trello**. If you don't already have an account in Trello, create one by going to **trello.com** and then clicking on "**Get Trello for Free**" button. **Don't use your corporate e-mail**, because your organization may already have other Atlassian products associated with your email and you may get errors if you try to use Trello. You can use a **disposable mail service** like:   
<https://temp-mail.org/en/> <https://www.disposablemail.com/> <https://www.emailondeck.com/>

The following tasks are **not mandatory:**

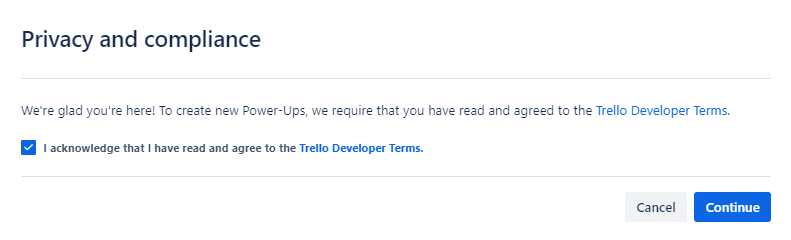
* After account setup, create a new board named **"Learning Postman".**
* On the board, create lists titled **"To Do" and "Done" to organize tasks**.
* Add tasks such as **"Sign up for Trello"**, **"Read API documentation"**, and **"Use the Trello API"** to the   
  **"To Do"** list.
* Move tasks to the **"Done" list**, once they are **completed.**

The **API will allow us instead of using the trello.com website**, to use **Postman to interact with it and create boards, lists, tasks, manage and test them.**

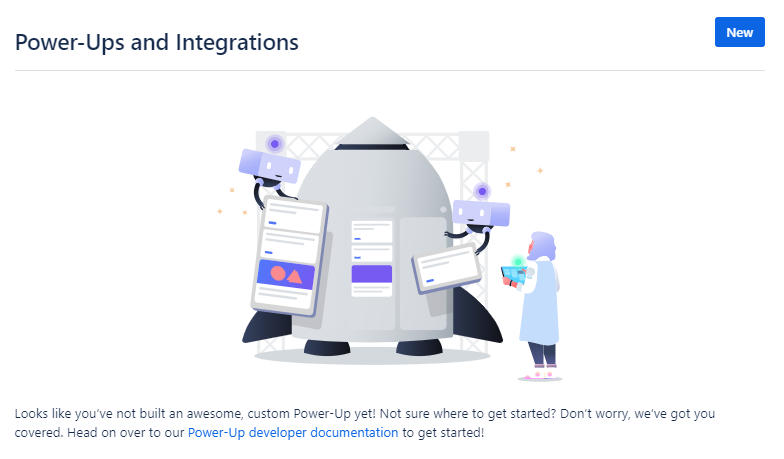
## Trello API Authentication and Authorization

In order to **use** the **Trello API**, we need to have an **API Key** and **Token**. Trello API **doesn't use passwords**. In order to have API Key and Token we need to **create Trello Power-Up**. At its core a Power-Up is **just a configuration**. So just follow the next steps to generate all the attributes needed or you can read [Trello documentation](https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/).

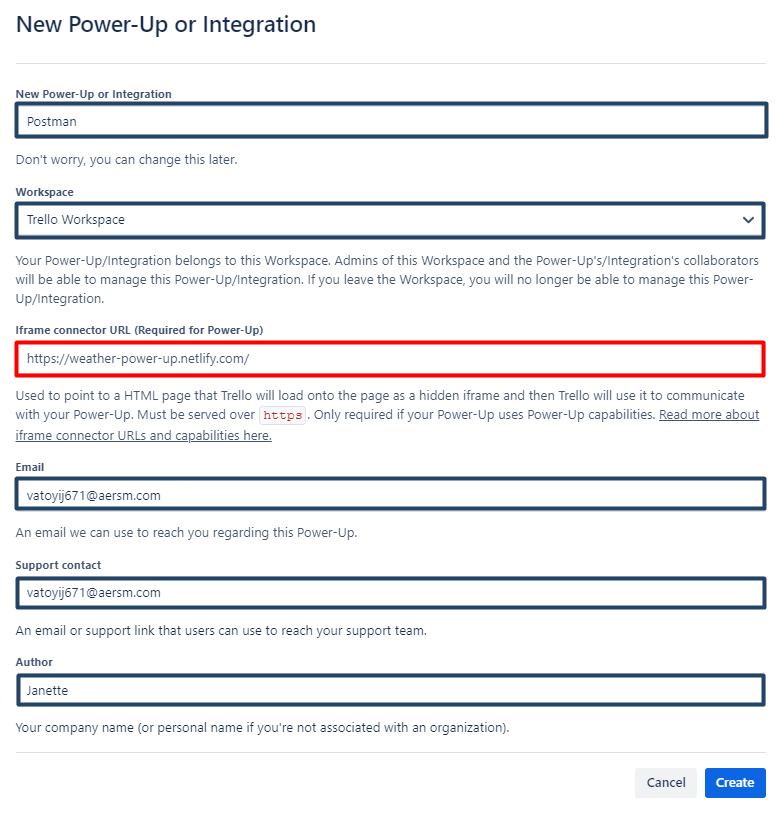
* Go to <https://trello.com/power-ups/admin>
* Agree to the [Trello Developer Terms](https://developer.atlassian.com/platform/marketplace/atlassian-developer-terms/)



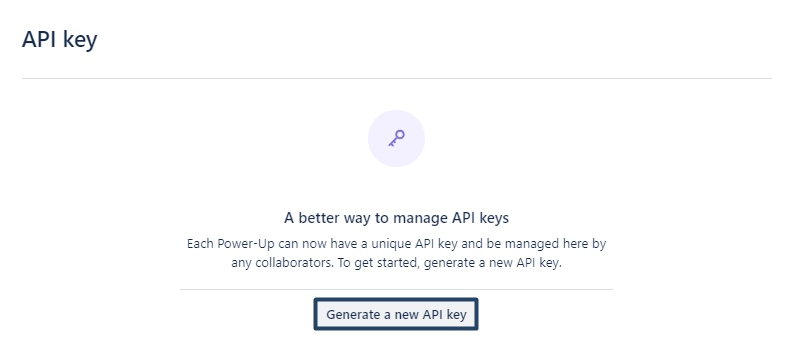
* Create a New Power-Up



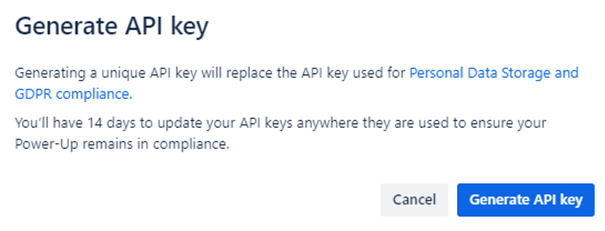
* **Fill in some data**. **Leave the red one as it is**. We **named our Power-Up "Postman"**, but feel free to   
  choose otherwise.



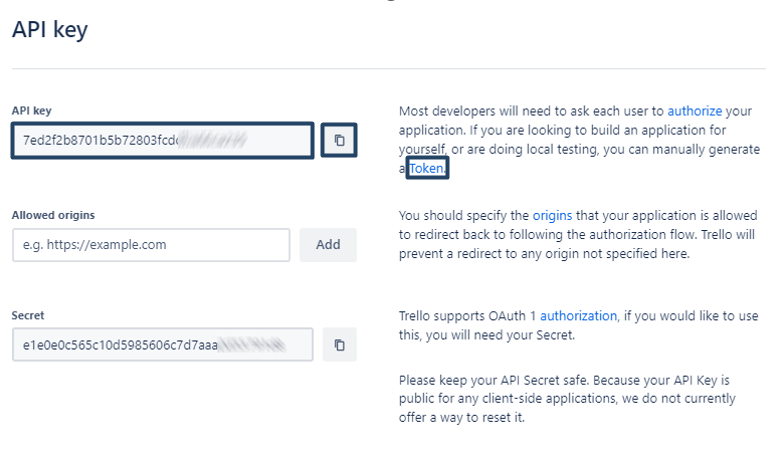
* Click on **"Generate a new API Key button"**



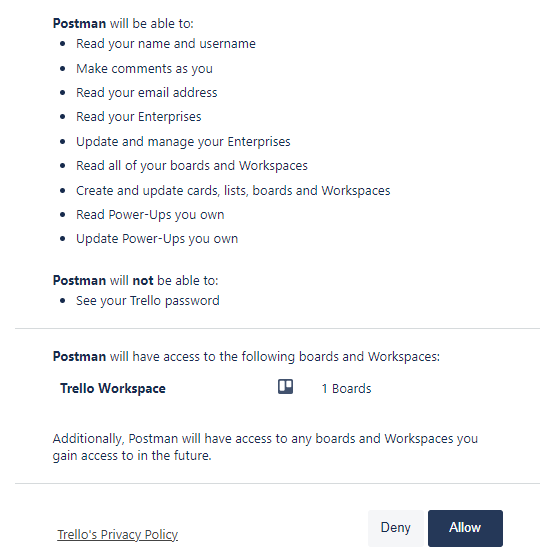
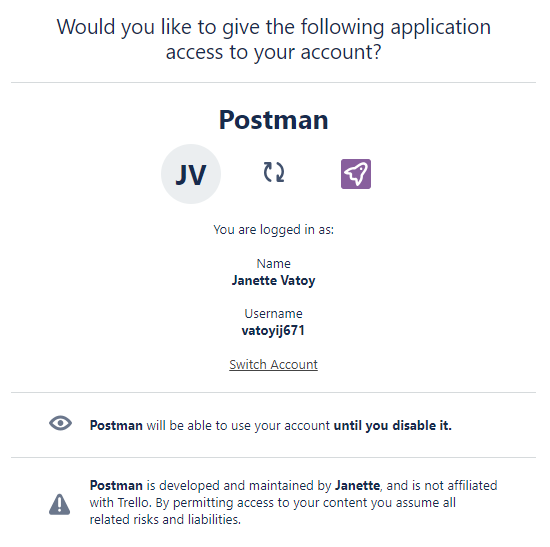
* And then **"Generate API key"**



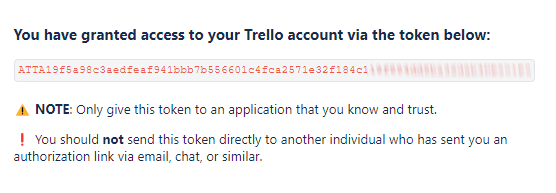
* We need the **API key**. **Forget the other options**. **Copy and paste your API Key somewhere safe**, because we will need it in order to make requests and click on **Token**.



* On the page that will **open in a new tab**, you have to **give permission** to **your Power-Up** (in our case, we named our Power-Up "Postman", remember?) to **use your Trello Account**.



* And it will **generate your Token**. **Copy and paste this Token somewhere safe**, because you will need it in your Postman requests.



## First Request

Now that **you have you API Key** and **your Token**, (you can think of your API Key as a username and Token as a password) let's try our first API call. We will get all of the boards from our workspace. Considering you have at least one.

https://api.trello.com**/**1/members/me/boards?key={yourKey}&token={yourToken}

So, what we see here is the **endpoint**, which is: **https://api.trello.com/1/members/me/boards**And two query parameters – **key and token**. **This is the syntax that this API uses.** You need to **replace "yourKey"** with the value of your **API key** and **"yourToken"** with the value of your **Token**. Remember that these **{ }** are just placeholders, so remove them.

* So, send your first request!
* In the response returned you should see the board that we created via Trello website or any other board that you have in the Trello Workspace.



## Create a Collection for your Requests

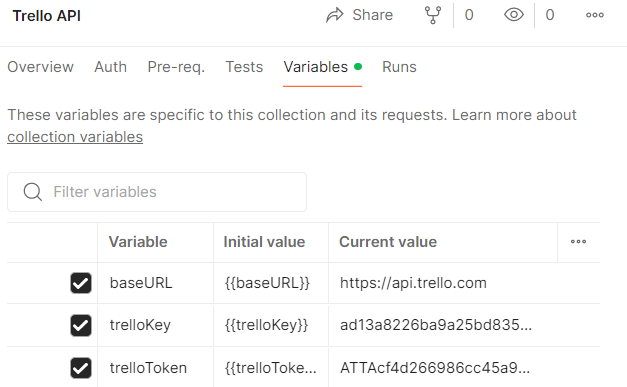
We've already created a collection that you need to import in Postman.

* Give it a **proper name**. We called ours "**Trello API**" (not very creative)
* **Move** the above **request** in the **"Trello API" Collection** and **rename it to "Get All Boards"**

## Collection Variables

Since we will **use** [**https://api.trello.com**](https://api.trello.com) **in all of our requests**, also **each request requires authentication** (**API key and Token**), let's turn those into **collection variables**.

* Add new **collection variable {{baseURL}}** variable with value **https://api.trello.com**
* Add new **collection variable {{trelloKey}}**
* Add new **collection variable {{trelloToken}}**



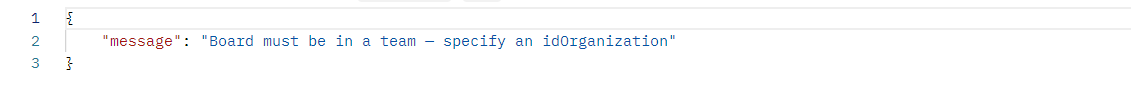
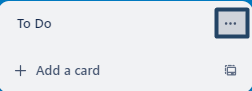
* At the end the **URL** of **"Get All Boards" request** should look like this:



## Trello API Documentation

Whenever you're trying to use a new API, you need to **find and study the API documentation**. It will give you an idea about **everything you need to know** about the API:  
<https://developer.atlassian.com/cloud/trello/rest/api-group-actions/#api-group-actions>

## Writing requests

1. **As you create your requests:**
   * Feel free to **check Trello's Website** to **observe what's happening;**
2. **Get all boards**You've already done this one
3. **Create a new board**  
   You can find the URL and the required parameters here:   
   <https://developer.atlassian.com/cloud/trello/rest/api-group-boards/#api-boards-post>   
   **Assert that the response code is 200  
   Note: If you ever get the following message as a response, it means that you have exceeded the maximum of boards that you can have and need to close some of them.**  
   
4. **Get a single board**You can find the URL and the required parameters here:<https://developer.atlassian.com/cloud/trello/rest/api-group-boards/#api-boards-id-get>   
   **Hint:** You can **find the id of the board** from your **"Get All Boards" request**  
   **Assert that the response code is 200**
5. **Create a TO DO List** **on your Board**  
   **Note: Since Trello creates three sample lists named 'To Do,' 'Doing,' and 'Done' for each new board, please go to Trello.com and archive these lists to avoid any confusion.** **🡪** 

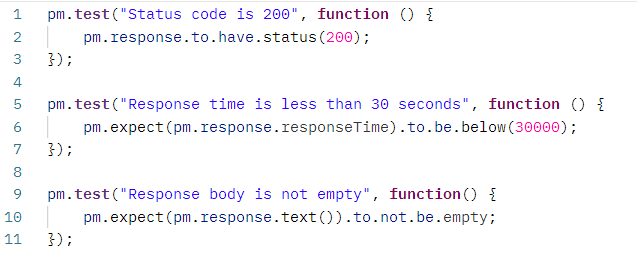
You can find the URL and the required parameters here:  
<https://developer.atlassian.com/cloud/trello/rest/api-group-boards/#api-boards-id-lists-post>   
**Assert that the response code is 200**

1. **Create a DONE List** **on your Board**  
   You can find a **different way** to create a list here:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-lists/#api-lists-post> **Assert that the response code is 200**
2. **Get All Lists from a Board**  
   You can find the URL and the required parameters here:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-boards/#api-boards-id-lists-get>   
   **Assert that the response code is 200**
3. **Create a Card in the TO DO List  
   Create a "Sign-up for Trello" card in TO DO list**  
   You can find the URL and the required parameters here:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-cards/#api-cards-post>   
   **Assert that the response code is 200  
   Hint:** Get the **id of the TO DO list from the previous request**.
4. **Move Card** **to DONE list**  
   This is done via PUT request:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-cards/#api-cards-id-put>   
   **Hint:** Think what query parameter you need to add in order to change the list  
   **Assert that the response code is 200**
5. **Delete the card**  
   You can find the URL and the required parameters here:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-cards/#api-cards-id-delete>   
   **Assert that the response code is 200**
6. **Delete the board**  
   You can find the URL and the required parameters here:  
   <https://developer.atlassian.com/cloud/trello/rest/api-group-boards/#api-boards-id-delete>   
   **Assert that the response code is 200**  
     
   **We only have tests asserting the status code is 200, but let's try to run our tests once more one by one. Most of the tests are failing, right? Why is that?**

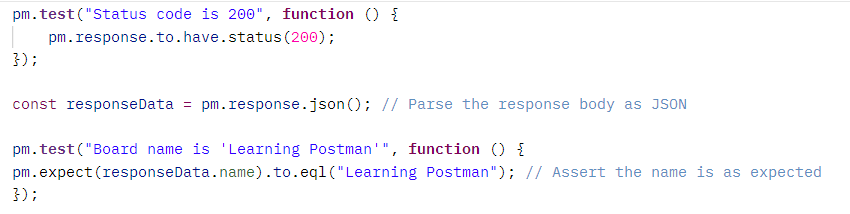
## Writing Scripts and Tests

Let's take a closer look at our requests

1. **Get All Boards**Nothing wrong here. Although, we can add some more tests:
   * **Assert that the API response body is not empty**
   * **Assert that API response time is less than 30 seconds**

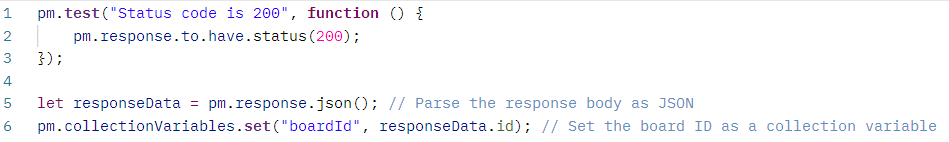


1. **Create a new board**Nothing wrong here either. **Each time** we're **creating** a new board, with the **same name** (in our case) "Learning Postman", but since **we're deleting the board at the last request of the collection**, we won't end up with multiple boards with the same name.
   * **Assert that the board created has the expected name**



1. **Get a single board**This **test** will **pass once** and then **fail each time**. Why is that? **Each time** we're **creating a board**, no matter its name, the **API is giving it a different id**. So, when we try to **Get the board** from the previous request the test will pass just once, because at the end of our collection we are deleting the board and creating a new one with a different id. In order for **this test to pass** we have to **get the id of the board** from the **previous request each time**. How to do that?

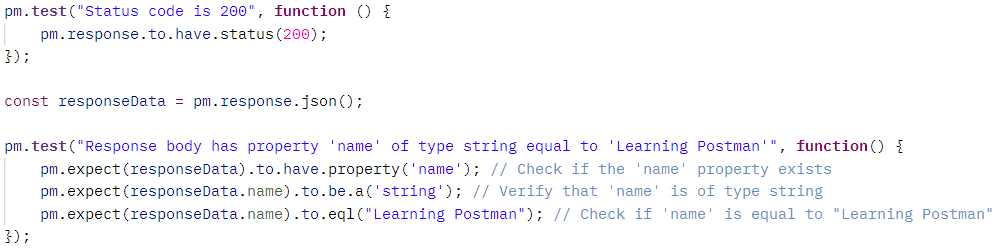
* In the **Tests tab** of the **previous request (Create a new board)**, we parse the response body and **extract the board ID**, then we **set this ID as a collection variable** within Postman.



* Then in the **current request** **(Get a single board)** use **{{boardId}}** in the URL **as a path variable**.



* **Assert that the response body has property 'name' of type string and it is equal to 'Learning Postman'**



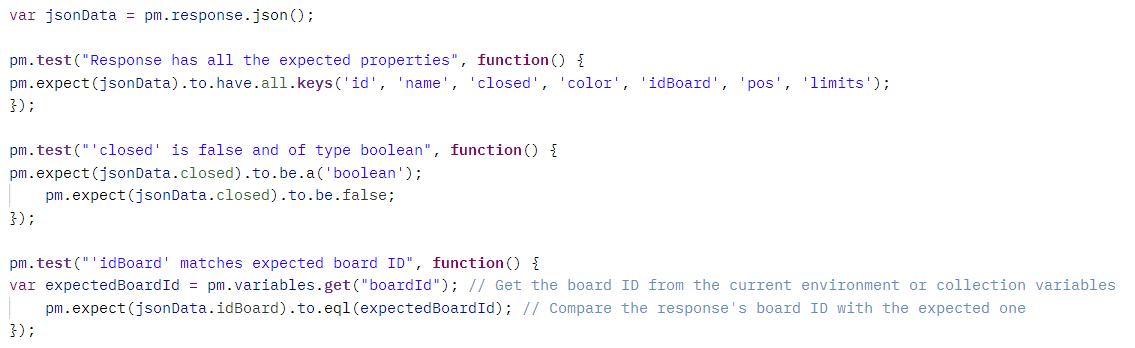
1. **Create a TO DO List**

You should **again use {{boardId}}** in the URL, but this time **as a query parameter.**



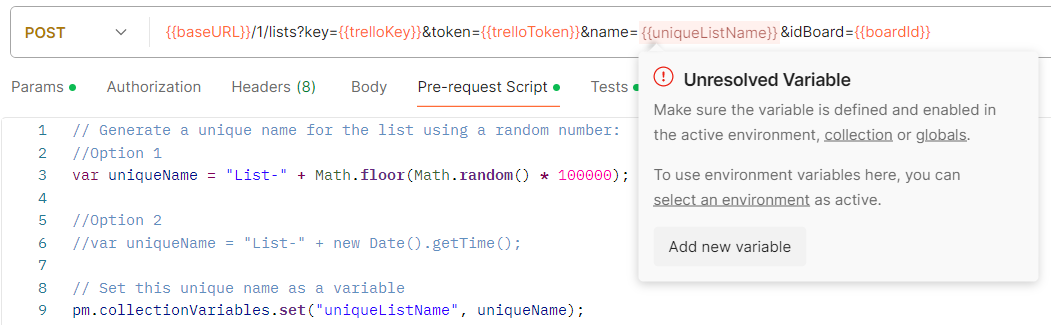
And let's write some more tests:

* + **Assert that the response contains all the expected fields**
  + **Assert that the 'closed' field is false and is of type Boolean**
  + **Assert that the idBoard in the response matches the expected board ID**



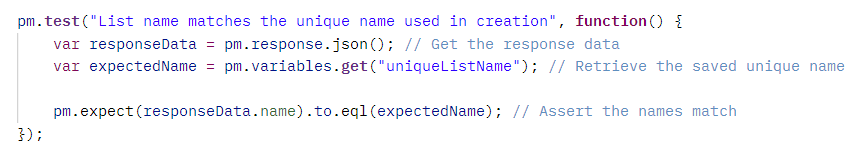
1. **Create a DONE List**You should **again use {{boardId}}** in the URL, **as a query parameter.** But how about **this time we are not creating a DONE list, but we create a list with a different name** every single time?!
   * **In the Pre-Request Tab generate a random number and use it as part of the list name**
   * **Sets this unique name as a Postman variable named uniqueListName**
   * **Replace the static list name with the variable {{uniqueListName}} you've just set in the Pre-request Script**

**Note: You can use Math.random or add the current date and time to create a unique name**

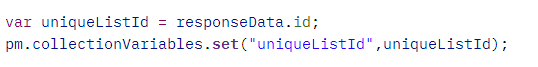


At first **{{iniqueListName}}** will light as **Unresolved Variable**, this is because we have to execute the request in order to run the pre-request script and to set the variable.

* **Assert that that the list created matches the unique name used in creation**

  
**Note:** Don't forget to rename the request from "**Create a DONE List" to something like "Move Card to List with Unique Name"**

* **Add the following in the Tests tab as well. Believe me you will need it.** 😉

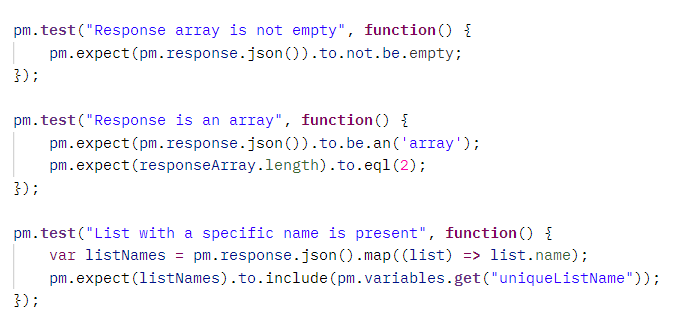


1. **Get All Lists from a Board**Nothing special here. Just replace the path variable {id} with the {{boardId}} variable



Think what you can assert here?!

* **Response is an Array**
* **Array is not empty**
* **Presence of a List with a Specific Name**



1. **Create a Card in the TO DO List**

So, what do we need here?



**First, we need the id of the TO DO list and** maybe, if you want to practice creating unique names each time you create a card, you could do that, but it's up to you. We'll not showing it again. You have the information needed in the previous requests. So, we'll create "Sign-up for Trello" card each time.

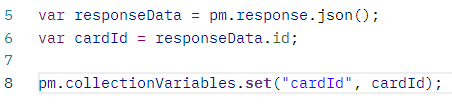
* **So, TO DO list id, we head back to "Create TO DO List" request and add the following script into   
  the Tests tab:**



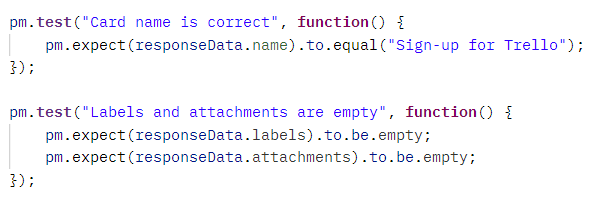
* + **Then we use the variable as a query parameter in Create a Card in the TO DO List**



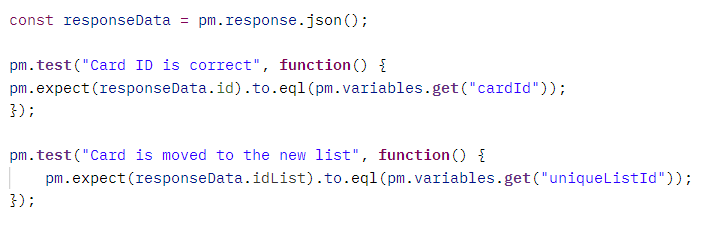
* We will need the **id of the card in the next two requests**, so it is advisable to write a script to get the id.

Since all of the boards, cards, lists have id, be really careful about the names of your variables in order not to duplicate it. Call this one **{{cardId}}.** The script should be placed in the Tests tab.  


* Write some asserts here. What can you assert? Many things in the response. **Be creative!**

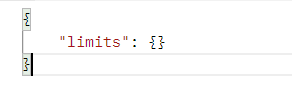


1. **Move Card** **to List with Unique Name (ex-DONE list)**You need to **replace the {id}** with the **{{cardId}}**. And also **replace**, the **hardcoded value** of the **query parameter idList**. Where can you get this one? You already have it. We took it when we created **List with Unique Name**  
   
   * **Assert that the card is moved**

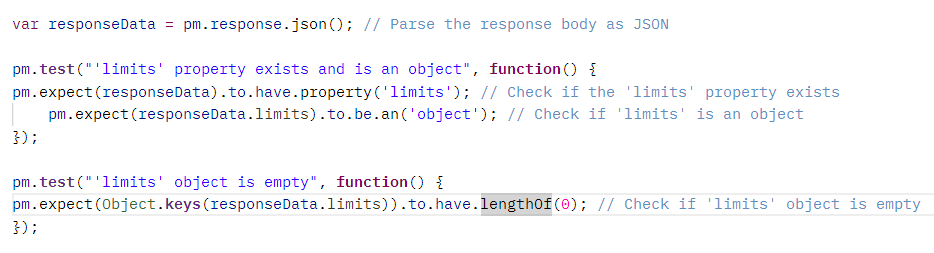


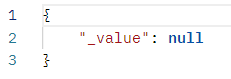
1. **Delete Card**Same here. Replace the **{{id}}** with the **{{cardId}}**

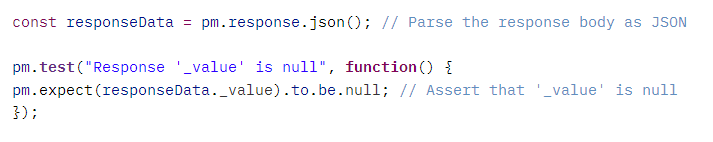


The response here is:  


What can you assert?



1. **Delete Board**You need the board id for this one.  
   The response is:  
   
   * **Assert that it is null.**



## Running all the tests

At this point we advise you to go to your Trello.com board and clear everything that might prevent you from observing the results of each request. Close any boards that may have been left open from the creation of requests and tests.

Run each test one by one. Make sure that each pass. After you've run all the tests, you should see your Trello workspace looking just like it did at the start. This is because our first request creates a board, and our final request deletes that same board.

In the next exercise, we will proceed with automation...