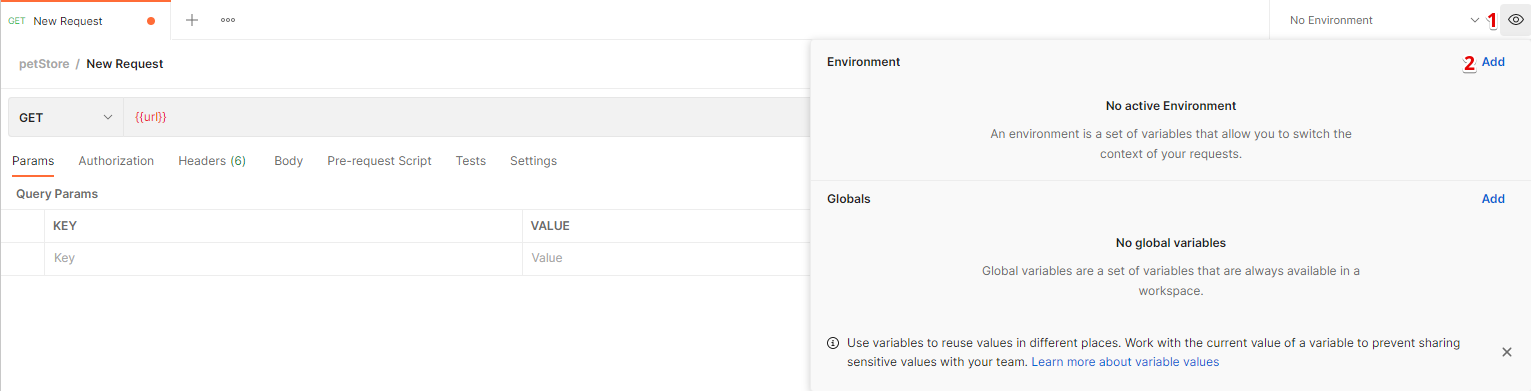
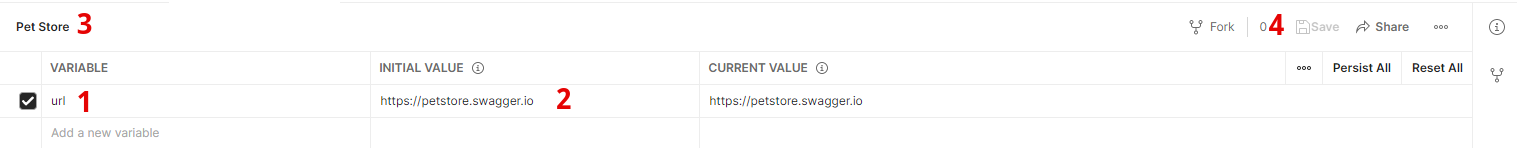
**Переменные** - в них добавляем то что потом будем использовать. Наприер можно создать разное окружение (прод, тестинг, релиз,..)

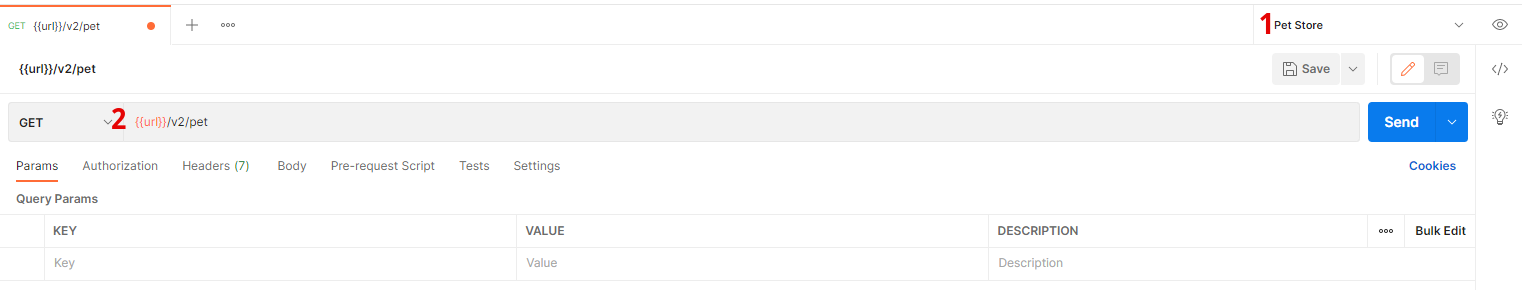
**Окружения**



Пишем в VARIABLE – название нашей переменной. Вставляем в INITIAL VALUE – url нашего API. Вводим название окружения и сохраняем



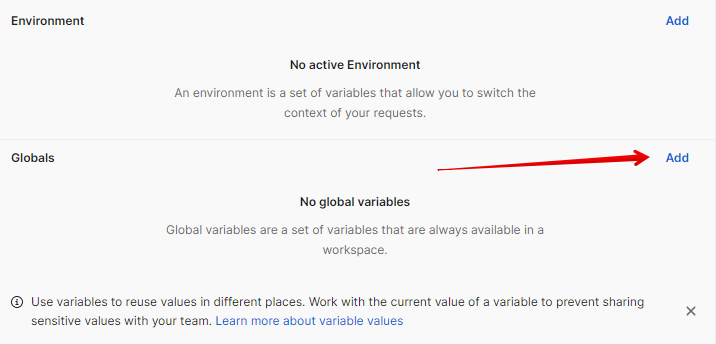
Теперь через двойные фигурные скобки можно указывать переменную. Только заранее нужно выбрать окружение

****

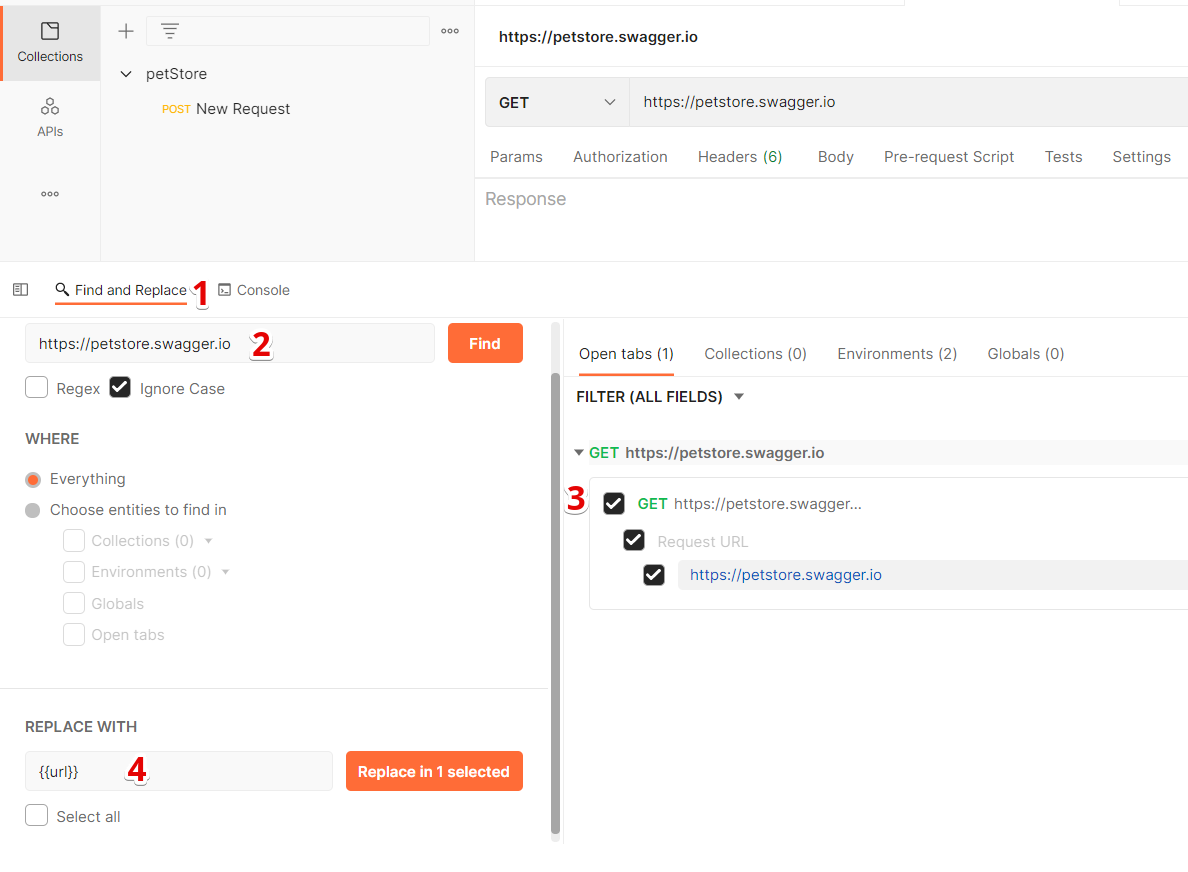
**Переменные можно создавать так же на уровне коллекций**

Collections > 3точки > Edit > Variables

**Переменные можно созадавать глобально. Они доступны для всего WorkSpace**



Если у нас уже создана куча запросов, то можем конвертировать их значения в переменные



**DYNAMIC VARIABLES**

­{{$variableName}} – с помощью такой структуры можно использовать динамические переменные, которые встроены в postman

<https://learning.postman.com/docs/writing-scripts/script-references/variables-list/>

**How scopes work**

When you store a variable in a collection or an environment, Postman will give that variable a scope that corresponds to where you have stored it. When you're trying to resolve a variable, Postman will look for it in a defined order in each of the scopes. Scopes allow a variable to be used in broader or narrower parts of the application. If you define a variable in the global scope, you can use it in any request anywhere in the application, but if you define it at the narrowest scope, it will only be available during one particular iteration of a test. So, what are the available variable scopes in Postman? In order from the broadest to the narrowest scopes, they are as follows:

* Global
* Collection
* Environment
* Data
* Local

When resolving a variable, Postman will use the narrow scope in which that variable exists.

**Global scope**

The first scope we will discuss is the Global scope, which defines variables that are accessible globally or anywhere in the application.

You may want to use global variables when experimenting with sharing data between different tests or collections, but in general, you want to try and avoid using global variables. They are, by definition, available everywhere and if you use them, it is likely that you will end up eventually giving a variable in another scope the same name as the global variable. Postman will, of course, use the narrower scope, but this can still lead to confusion, especially when you're trying to debug failures.

**Collection scope**

Variables defined in the collection are available for all the requests in that collection. They are not available in other collections, but any request in that collection can use them. You want to create collection variables when you need to share the data between multiple requests in the collection, and that data will stay the same. An example of this is an application URL. Every request will need it, and it should stay consistent. Another example would be doing some setup work with a pre-request script and then storing that data in a collection variable.

**Environment scope**

Environments allow you to define a set of variables that belong to a certain context. For example, many companies will have testing sites and production sites. You might also have a staging environment or want to run tests on a local build. Each of these environments might require some minor changes needing to be made to the variables being used, so in these cases, you would want to create variables that live in the different environments.

For example, if you had staging, testing, and local sites that you wanted to test against, you might define an environment in Postman for each of those sites, and then in each of those environments create a URL variable that points to the correct endpoint for each of those environments. It is important to note that in these cases, you should try not to also have the same variable defined in the collection. This will lead to confusion and is not a good practice.

Another time you might store variables in an environment is if you want to try some API endpoints as different users. You might want to try as an administrator and a regular user, for example, and so you could create environments for each of those users. These will store the login information for them and maybe some specific variables that store expected results for those different users.

If you find yourself with only one environment for a collection, you are probably better off just creating variables directly in the collection instead of adding the additional complexity of an environment.

**Data scope**

Most variables will be defined in collections and environments. Using **Data-Driven Testing**. This kind of testing imports data into Postman from a CSV or JSON file. Postman will create variables in the data scope from this imported data. You cannot create these kinds of variables yourself within Postman, but you should be aware of the kinds of variables that you have defined in your collections or environments when you are creating your input data files. The scope of data variables is narrower than that of collections or environments, so any variables defined in this scope will overwrite values defined in those broader scopes

**Local scope**

The narrowest scope you can define variables in in Postman is the local scope. You can only create variables in this scope using request scripts. They are temporary variables that do not persist between sessions, but since they are the narrowest scope, they do allow you to override values that have been set in collections or environments. Sometimes, this is desirable as it lets you override a variable in one request while still using it in all the other requests in a collection, but you should be careful that you are not accidentally overriding them either.