



#### Request URL

**https://petstore.swagger.io/v2/store/order**

Применим технику «Таблица принятия решений»

Таблица 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| key | value | value | value | value | value | value | value |
| id | t | t | t | t | t | t | f |
| petId | t | t | t | t | t | f | f |
| quantity | t | t | t | t | f | f | f |
| shipDate | t | t | t | f | f | f | f |
| status | t | t | f | f | f | f | f |
| complete | t | f | f | f | f | f | f |
| **result** | pet is ordered | Error | Error | Error | Error | Error | Error |

T – Корректное значение, соответствующего типа

F – Некорректное значение, тип данных не соответствует заявленному

Если хотя бы одно значение не будет отвечать завяленному типу, то должна вернуться ошибка

Таблица 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| key | value | value | value | value | value | value | value |  |
| id | T | F | T | T | T | t | t | f |
| petId | T | T | F | T | T | t | t | f |
| quantity | t | T | T | F | T | t | t | f |
| shipDate | t | t | t | t | F | t | t | f |
| status | t | t | t | t | T | F | t | f |
| complete | t | t | t | t | T | T | F | f |
| **result** | pet is ordered | Error | Error | Error | Error | Error | Error | Error |

Так же применим технику попарного тестирования

«Отправляем» таблицу 2 в <https://pairwise.teremokgames.com/>

и получаем 10 вариантов проверок +вариант где всё T+ вариант всё F

То есть получаем всего 12 проверок

Таблица 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| key | value | value | value | value | value | value | value | value | value | value | value | value |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| id | ***t*** | t | t | t | t | t | f | f | f | f | f | ***f*** |
| petId | ***t*** | t | f | t | f | f | t | t | f | t | t | ***f*** |
| quantity | ***t*** | t | f | f | t | t | t | t | t | f | f | ***f*** |
| shipDate | ***t*** | t | f | t | t | f | f | t | t | t | t | ***f*** |
| status | ***t*** | t | t | t | t | t | t | t | t | t | t | ***f*** |
| complete | ***t*** | f | t | f | t | t | t | t | t | f | t | ***f*** |
| **Expected**  **result** | Pet is ordered | Error | Error | Error | Error | Error | Error | Error | Error | Error | Error | Error |

Дальше проверим вводимые данные согласно технике тест-дизайна «Эквивалентное разделение на классы». Только класс эквивалентности, который отвечает типу данных, должен давать правильный результат. В ином случае ожидается ошибка.

Однако, в некоторых значениях у нас указан тип integer ($int64), который может принимать  целочисленные значения от – 9 223 372 0368 54 775 808 до + 9 223 372 036 854 775 807 (у integer ($int32) чуть поменьше)))

С логической точки зрения, наши поля с этим типом данных (id, petId, quantity) не должны быть отрицательны, т.е. тут у нас будет еще 2 класса эквивалентности, и нам как-то в тестах надо будет проверять положительное число или нет.

Так же для полей «id», «petId», «quantity» можно проверить граничные значения

0 – вроде бы как не может быть в качестве id, но в программировании именно он первый )))

«1» и «9 223 372 036 854 775 807»

и можем проверить негативные значения «-1» и «-9 223 372 036 854 775 807»

Таблица 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **key** | **type** | **value** | **ER** | **notes** |
| id | integer  (number) | 11 | OK | Value must be positive |
| integer  (number) | -11 | Error??? | Should be error, but can be OK, because type is integer |
| float  (number) | 11.5 | Error | We can use ‘.’ or ‘,’ |
| string | abcd | Error | also we can try any wildcard |
| NULL/ empty | field empty | Error |  |
| boolean | TRUE | Error |  |
| petId | integer (number) | 22 | OK | Value must be positive |
| integer (number) | -22 | Error??? | Should be error, but can be OK, because type is integer |
| float (number) | 22.22 | Error | We can use ‘.’ or ‘,’ |
| string | efij% | Error | also we can try any wildcard |
| NULL/ empty | field empty | Error |  |
| boolean | FALSE | Error |  |
| quantity | integer (number) | 33 | OK | Value must be positive |
| integer  (number) | -33 | Error??? | Should be error, but can be OK, because type is integer |
| float (number) | 33.333 | Error | We can use ‘.’ or ‘,’ |
| string | qwery | Error | also we can try any wildcard |
| NULL/ empty | field empty | Error |  |
| boolean | TRUE | Error |  |
| shipDate | string  ($date-time) | "2020-02-14T11:23:37.775+0000" | OK | Кроме проверки типа, надо было бы проверить и формат даты |
| number (integer/ float) | 23 | Error |  |
| NULL/ empty | leave field empty | Error |  |
| boolean | TRUE | Error |  |
| status | string  [placed, approved, delivered] | placed | OK | тут хорошо бы проверить не только тип данных, а и вхождение в массив значений |
| number (integer/ float) | 45,67 | Error | We can use ‘.’ or ‘,’ |
| NULL/ empty | field empty | Error |  |
| boolean | FALSE | Error |  |
| complete | boolean | TRUE | OK |  |
|  | string | as%df | Error |  |
|  | number (integer/ float) | 78.66 | Error |  |
|  | NULL/ empty | field empty | Error |  |

Возьмем 2 последние таблицы для построения тестового сценария в Postman. Из таблицы 3 «схему» тестирования, из таблицы 4 значения корректных и некорректных значений. Причем, будем миксовать некорректные значения различным образом, хотя тут наверное тоже можно выработать схему дейцствий.