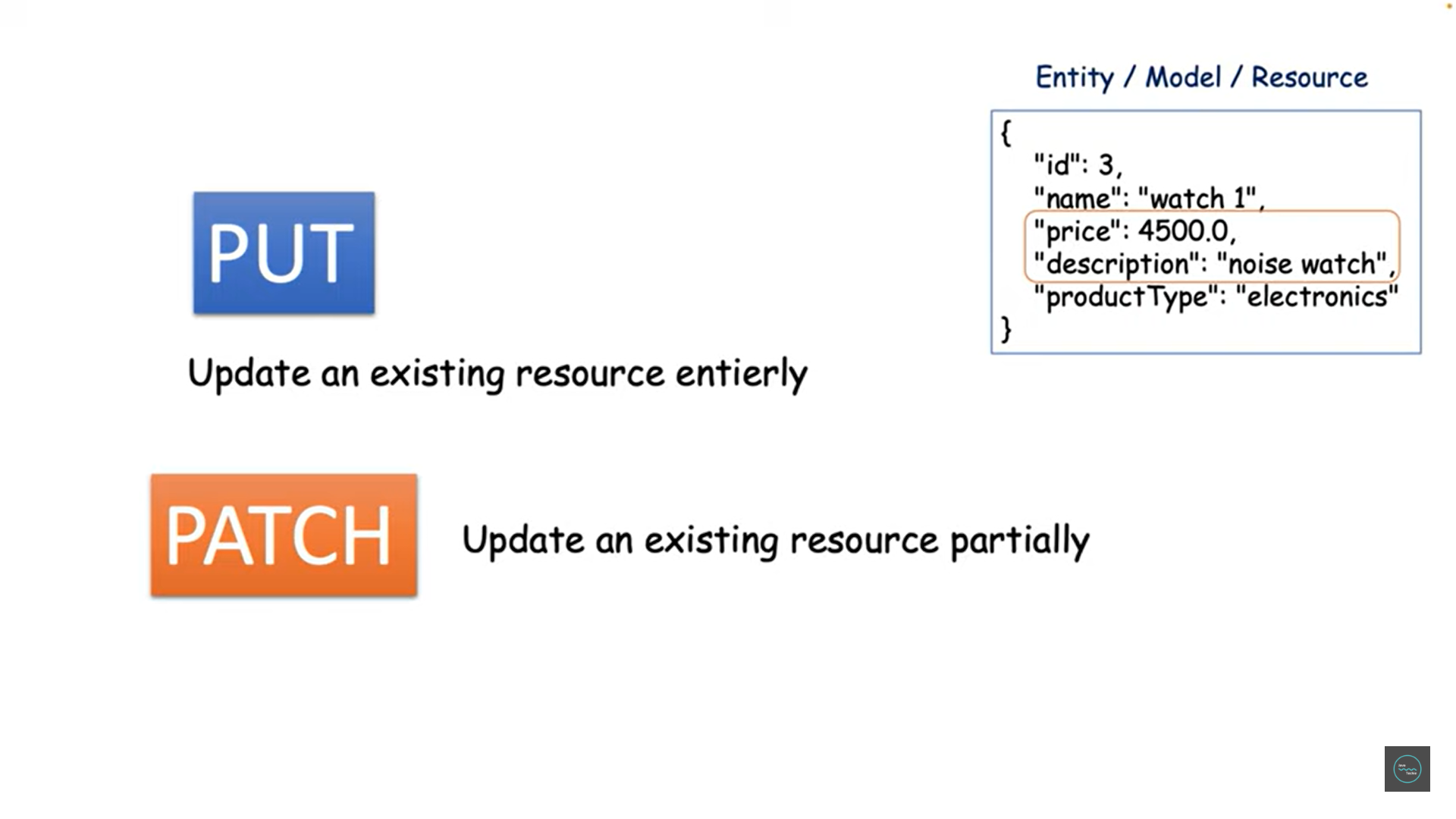
As we know there are 2 types of HTTP methods to update an existing resource **PUT** and **PATCH** method. Now the question is why we are using 2 types of HTTP methods to perform a same update operation.

PUT – you can go for the PUT HTTP method it’s always recommended to use if you want to update all the fields of your model or Entity.

Let’s say we have a product object which we consider as a model or entity and you want to update each and every field of an object, then you can go for the PUT HTTP method.

However, PATCH is recommended to use for Partial Update. Let’s say we have 10 fields or more than that of an entity and you want to update the partial field or few of the fields from object then you can go for the PATCH HTTP Method.



**POST :** [**http://localhost:9191/products**](http://localhost:9191/products)

**Body**:

    {

        "name": "Geysers",

        "price": 3469,

        "description": "Havells",

        "productType": "Kitchen & Home"

    }

**GET :** [**http://localhost:9191/products**](http://localhost:9191/products)

**Response:**

[

    {

        "id": 1,

        "name": "Android Mobile",

        "price": 10000.0,

        "description": "Samsung Galaxy",

        "productType": "Electronics"

    },

    {

        "id": 3,

        "name": "IPhone13",

        "price": 80000.0,

        "description": "Apollo",

        "productType": "Electronics"

    },

    {

        "id": 4,

        "name": "Camera",

        "price": 80000.0,

        "description": "Canon",

        "productType": "Electronics"

    },

    {

        "id": 5,

        "name": "Mixer Grinder",

        "price": 6999.0,

        "description": "Bosch",

        "productType": "Kitchen & Home"

    },

    {

        "id": 6,

        "name": "Geysers",

        "price": 3469.0,

        "description": "Havells",

        "productType": "Kitchen & Home"

    }

]

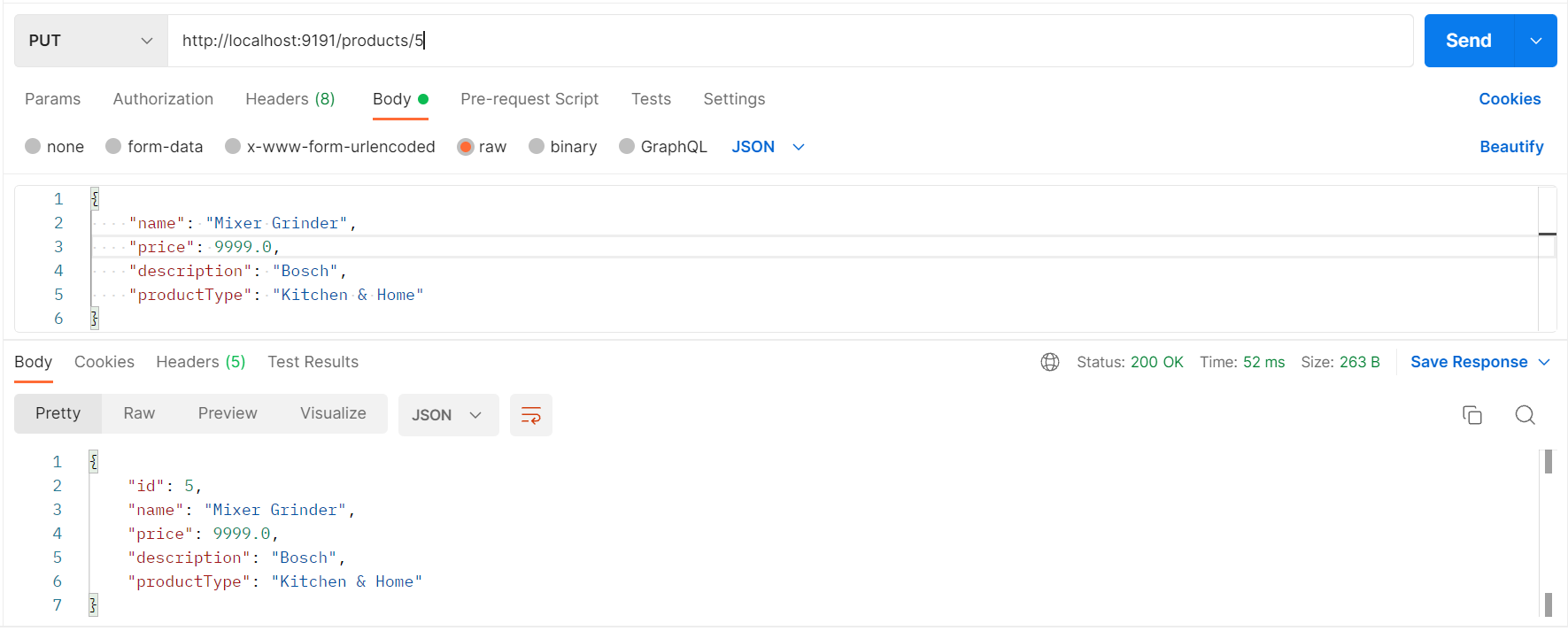
**Using PUT Method**

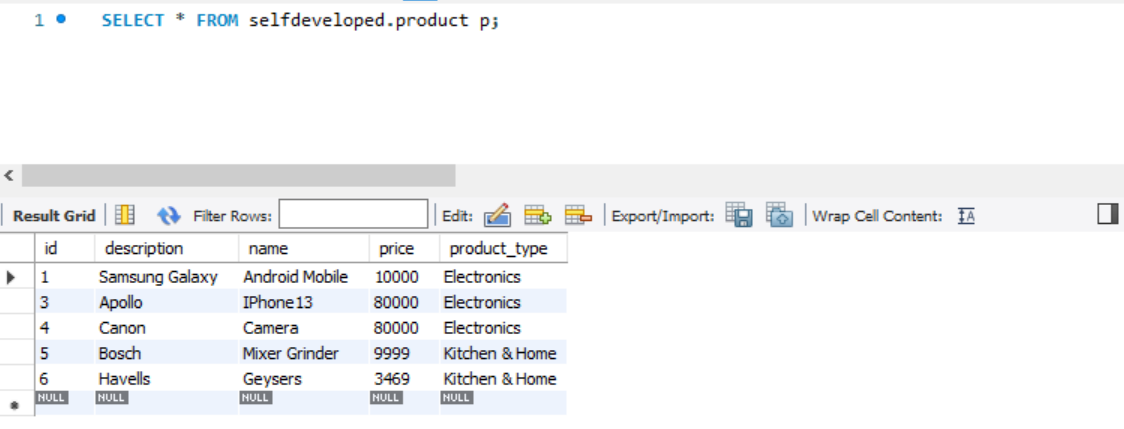
Now currently there are 9 products available in our product database. now let me update one of the product object using **PUT method**. Let’s say we want to update one of the product **Mixer Grinder** which id is 5.

So, id we need to pass in Path variable and rest another field with payload…

Let’s say I just want to update the price to 9999 and make sure I am giving each and every inputs. See here I am not updating the name, I am not updating the description and product type but still I am giving as part of request. Let me hit the endpoint and the value got updated.

**PUT** [**http://localhost:9191/products/5**](http://localhost:9191/products/5)

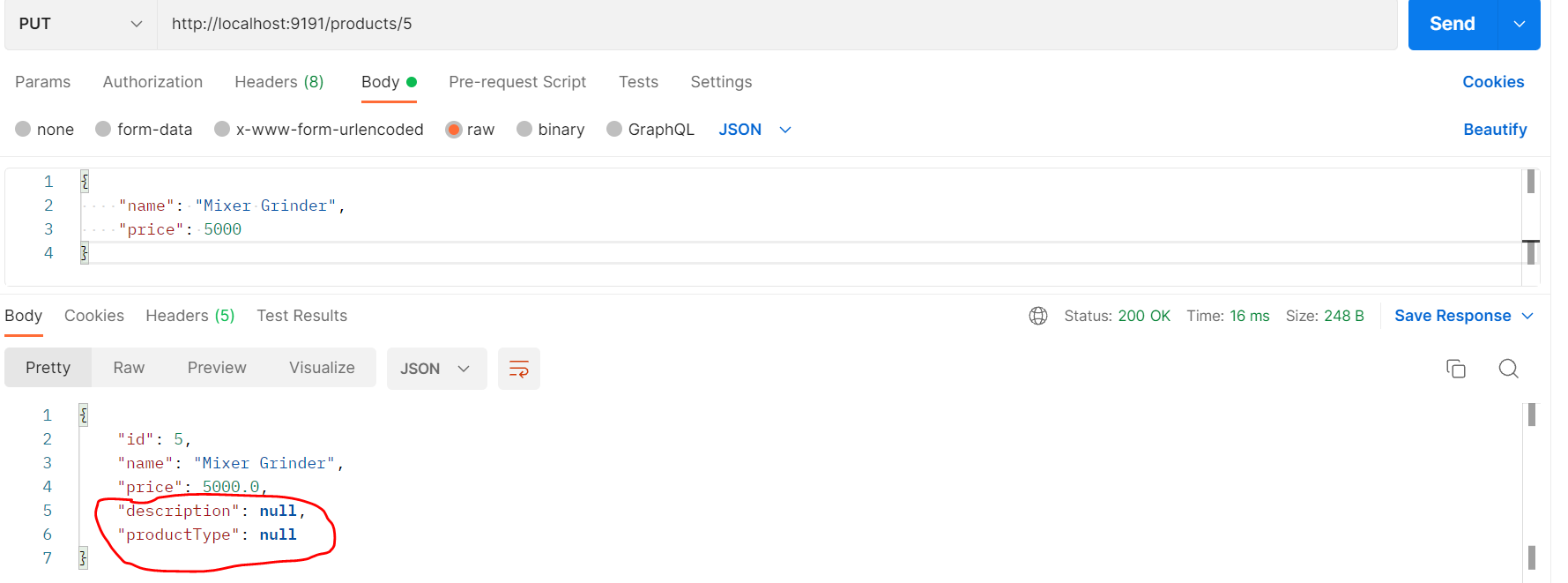




It’s got updated that’s cool but if you observed here even though I am not updating other field still I need to give those information as a part of request.

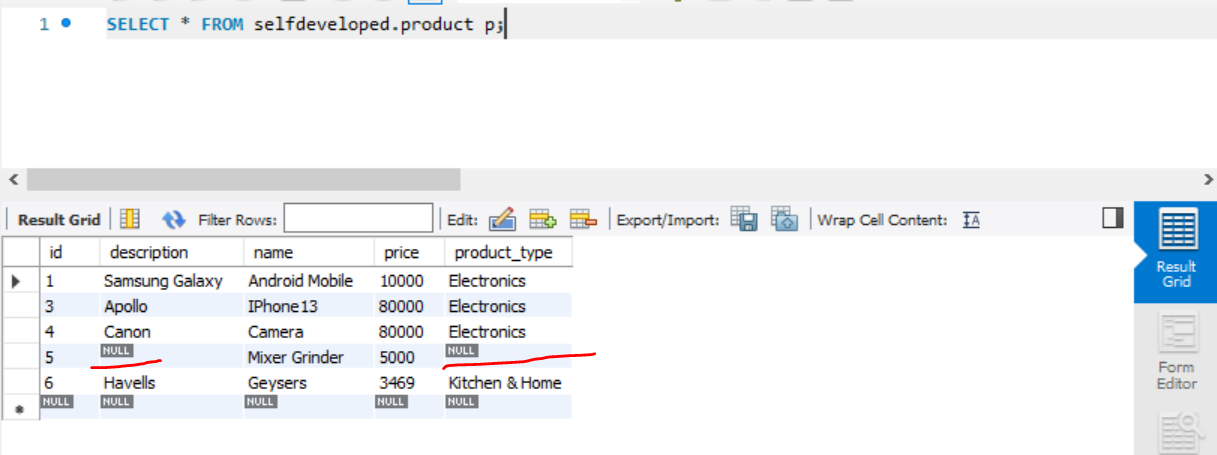
**Which is the wrong Designing.**

Now let’s see if I don’t want to update this product type and description, so I will simply remove it then let’s see how the PUT method will behave now. I will just update something called let’s say 5000. Only price I am updating but I am giving the value of name even though I am not updating but description and product type I am just removing as part of the request itself and I am doing the PUT mapping execution which is the PUT HTTP method and let me send the request.



Can you see here the description and the productType value is null here, why because we are not giving these two field as part of the Request.

So, if you are using **PUT** method then make sure to include all the field as part of the Request even though you are not changing a single value from them so, you need to keep the entire request body while performing the **PUT** method nowlet me check into our DB either values really got updated with null or not.



So, this is one of the drawbacks if you are using **PUT** mapping to update the field or to update the entity then you must need to pass the entire request body as part of your payload if you not give that input then the value will be set to the default which is null.

**So, To overcome that there is another HTTP method that is patch mapping or patch HTTP method you can use that for partial update, so in this case I just want to update the price so I can use the patch mapping without giving all the request input or all the attribute of my object as part of the payload. Now let’s see how we can use the patch mapping.**

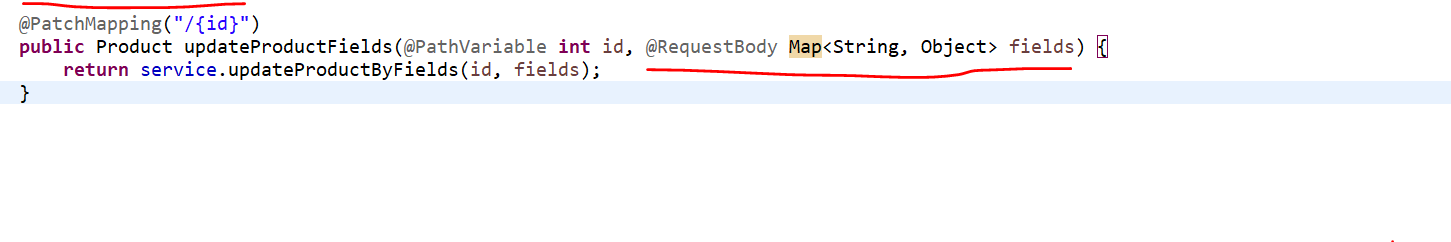
Note:-

Make sure if you want to perform the full field update then go for only the HTTP method **PUT** being a developer I also maximum time use the put mapping not patch mapping but some time you will find the real time usecase where only you need to update either the mobile number or Gmail id or specific unique number then in that case this PUT mapping will give you such kind of behavior. So, in that scenario you need to go for PATCH mapping.

**Using PATCH Method**

Let’s create an endpoint for partial update. You just need to pass only those fields which you want to update. So, I can keep them as a part of keys and value in Map.

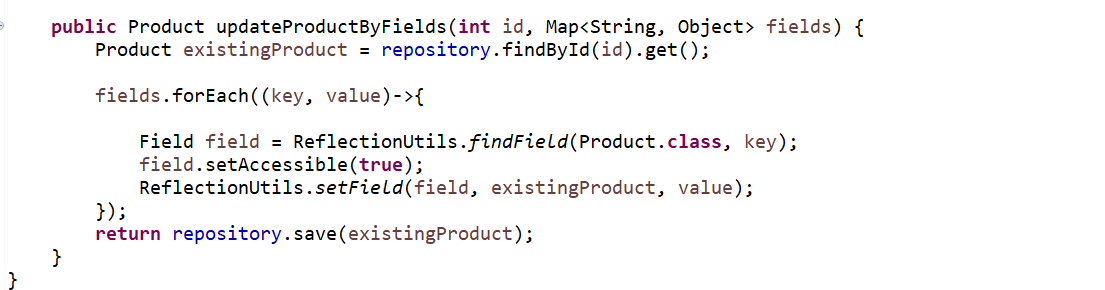
Key will be type of String which is nothing your field name which you want to update, and value is object.

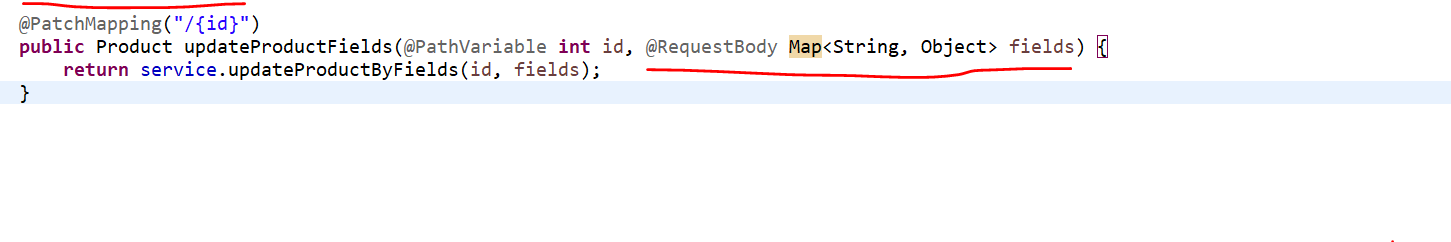


So, let me create this method int the service class.so, first step to update anything first you need to get that from DB. So, I will use the same logic to get the existing product by id. now once I got the existing product, I want to update few of the fields which I will give as part of the request not all the attributes of my object.

**So, what I can do since this is a map, I can iterate it forEach, I will just give key and value. then simply I can iterate each key and value. now here you can use the Reflection Utils class given by the spring framework. To get the field and to set the fields dynamically.**

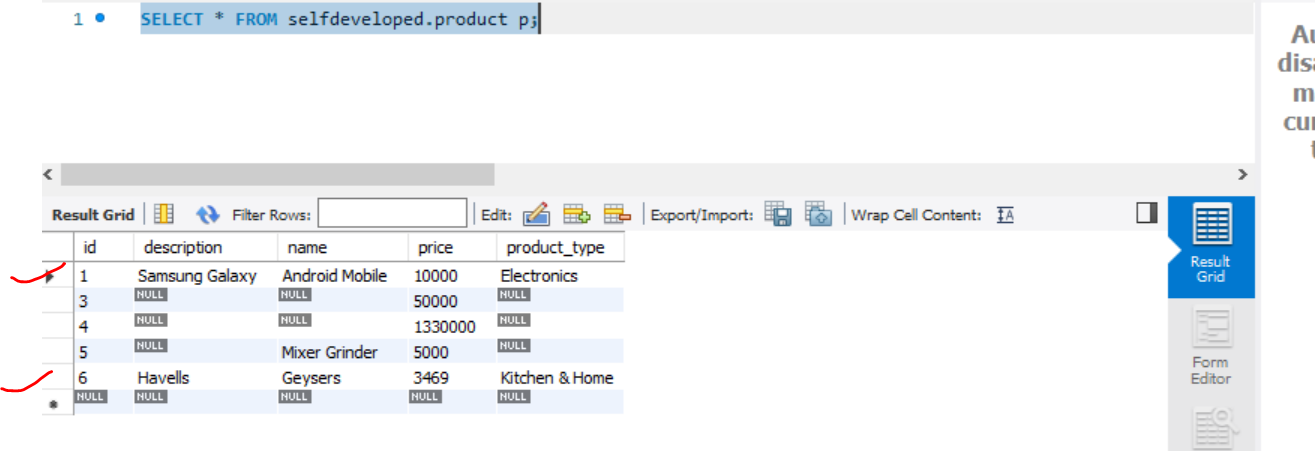
**So, I will show you there is a class called Reflection Utils, make sure to import it from the correct package and then call findField() method. So, I have a Map of key and value and simply looping that map so that key will be nothing my variable name and value will be the input value with that value I want to update the object. So, first I will get the field from Product.class** which exact field I want to update as part of the request then I am just setting that field with the new value, and I am telling this object I want to modify.



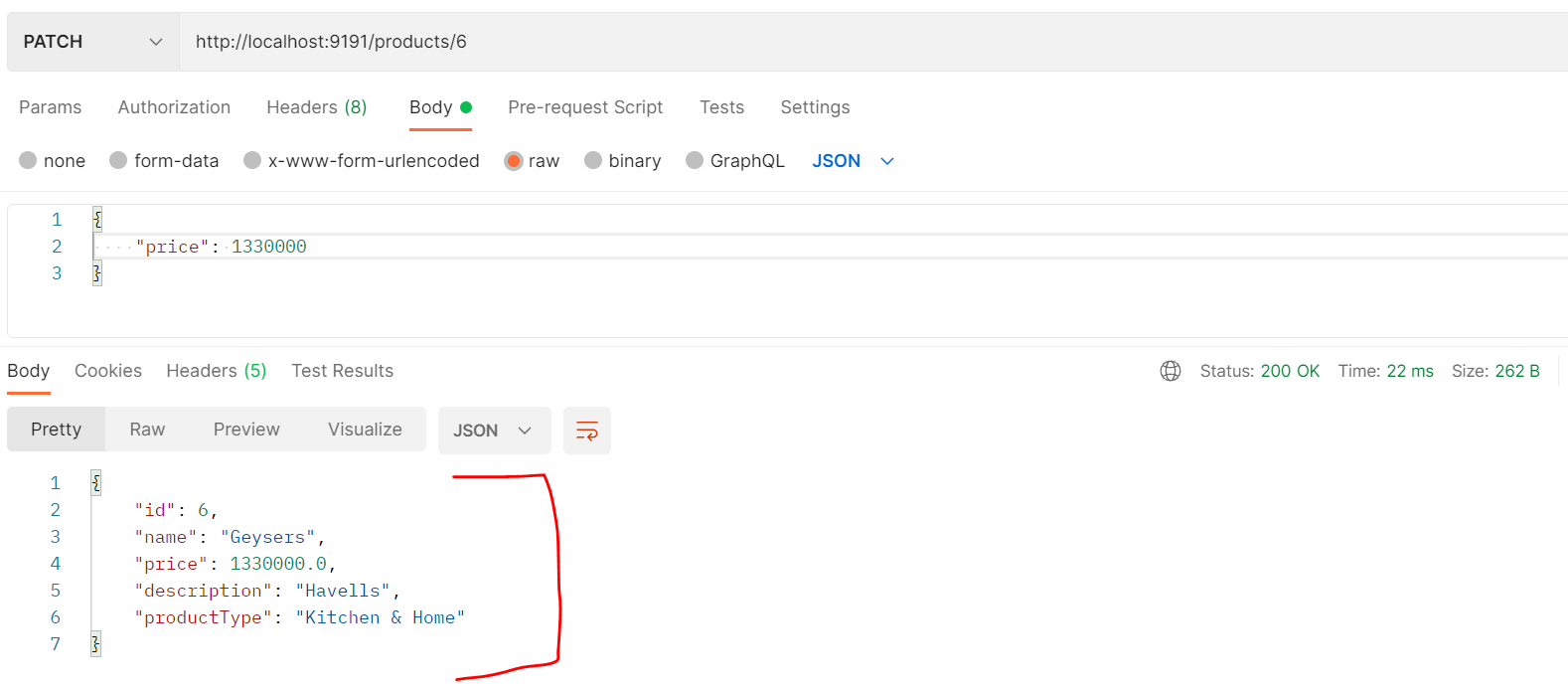


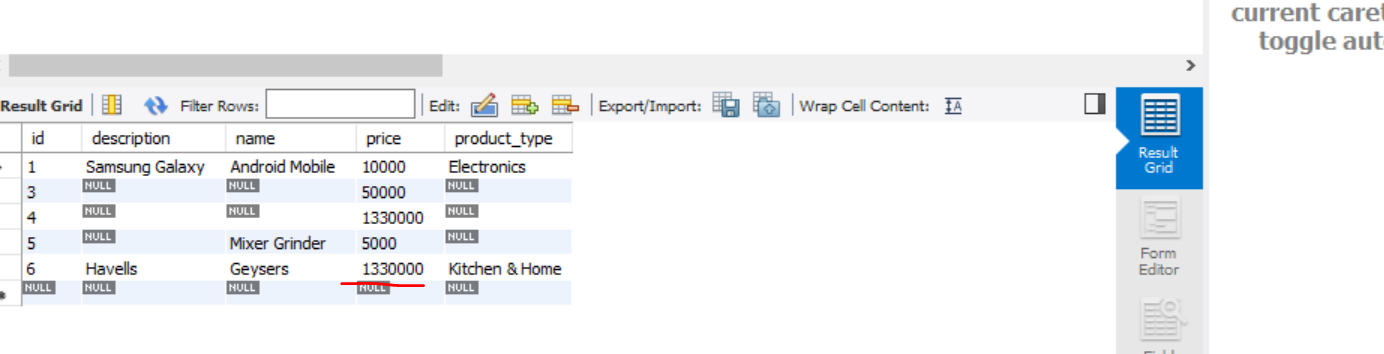
For the id 6 let’s try to use patch method

Before calling let’s see what object is there in DB then we will check after update

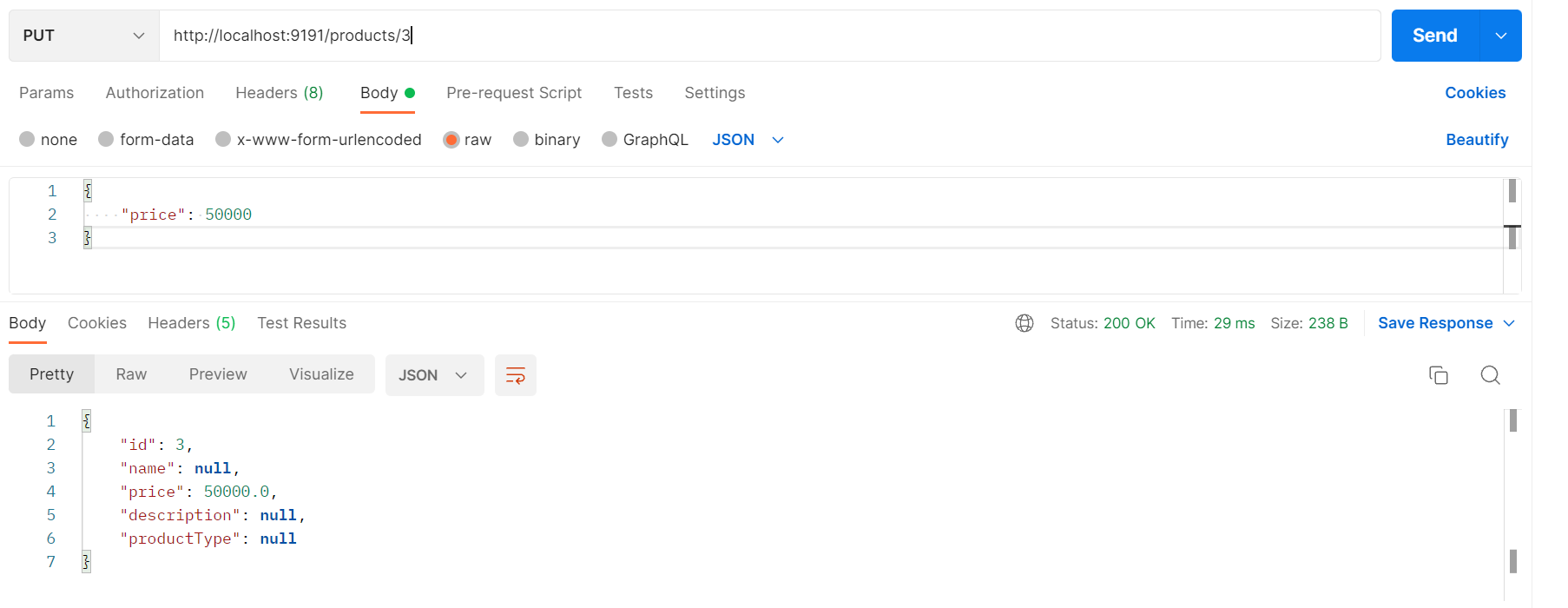


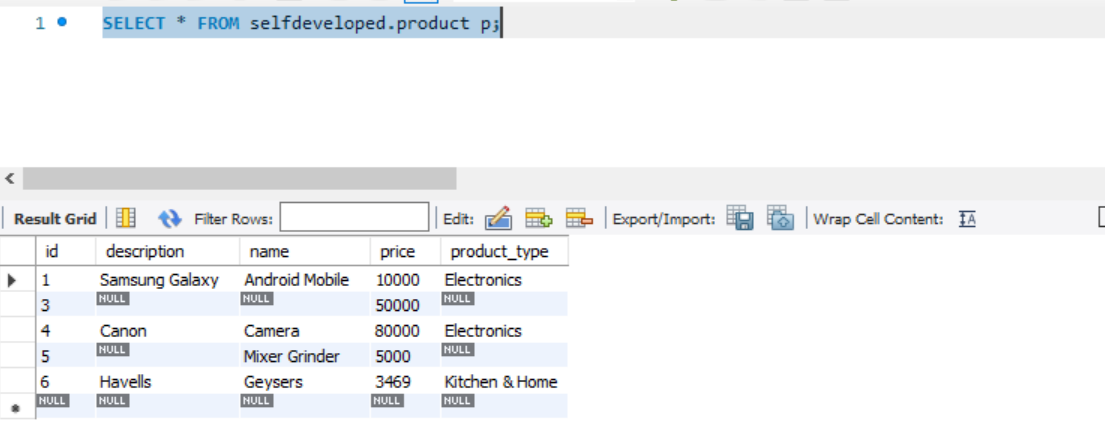
After update let’s see updated values…



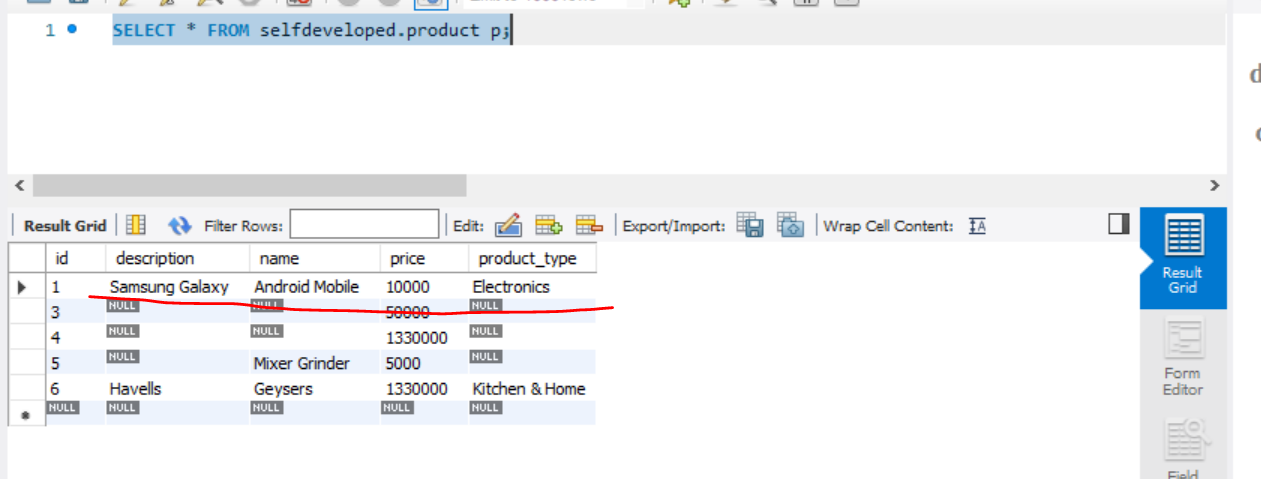


**While using put**





2nd test example Before Update for id = 1



After update

