

- Implement Put method

```
@PutMapping(value =("/{id}")
public ResponseEntity<ClientDTO> update(@PathVariable Long id, @RequestBody ClientDTO dto){
    dto = service.update(id, dto);
    return ResponseEntity.ok().body(dto);
}
```

- Implement update service
- If we put the findById in service, we will access the database twice (To find and to save). To not access twice the database we use the findOne method.
- Be careful, we have a mistake in the name of the exception

```
@Transactional
public ClientDTO update(Long id, ClientDTO dto) {
    try {
        Client entity = repository.findOne(id);
        entity.setName(dto.getName());
        entity.setCpf(dto.getCpf());
        entity.setIncome(dto.getIncome());
        entity.setBirthDate(dto.getBirthDate());
        entity.setChildren(dto.getChildren());
        entity = repository.save(entity);
        return new ClientDTO(entity);
    }
    catch(EntityNotFoundException e){
        throw new ResourceNotFoundException("Id not found " + id);
    }
}
```

- Lets Test in Postman

The screenshot shows the Postman interface. At the top, a collection named 'Client' is expanded, listing four requests: 'All Clients' (GET), 'Client by id' (GET), 'New Client' (POST), and 'Update Client' (PUT). The 'Update Client' request is selected, and its details are shown below. The request is a PUT to the URL `{{host}}/clients/1`. The body is a JSON object representing a client: `{ "name": "Maria Silvaa", "cpf": "12345678901", "income": 6500.0, "birthDate": "1994-07-20T10:30:00Z", "children": 2 }`.

Body Cookies Headers (5) Test Results

ⓘ Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Maria Silvaa",
4   "cpf": "12345678901",
5   "income": 6500.0,
6   "birthDate": "1994-07-20T10:30:00Z",
7   "children": 2
8 }
```

- Give the 200 response. Is ok