BREVI CENNI SULLE CHIAMATE HTTP E PARTE BACKEND (Lato server)


Pronto, pronto

Manco er telefono
sanno usà!

Chiamate http

Esistono vari tipi di chiamate http:

- **Get - permette di recuperare dei dati**
- **Post - permetti di postare dei dati, attraverso un raw type**
- **Put, permette di postare dei dati, attraverso un raw type.**
- **Delete, permette di eliminare un'istanza dell'oggetto passato alla chiamata**

Posssiamo testare le nostre chiamate http, attraverso lo strumento Postman. https://www.postman.com/

Già in questa fase siamo in grado di capire cosa ci sta effettivamente restituendo la nostra applicazione.

Dove?

Se non abbiamo dato altre specifica la porta di default è http://localhost:8080/

A livello di applicazione lato server, possiamo mappare a nostro piacimento, lo start effettivo della nostra applicazione, ovvero il percorso dal quale partirà lo start e tutte le nostre chiamate HTTP

La mappatura la inseriamo a livello di controller della nostra applicazione, nella classe che annotiamo come controller

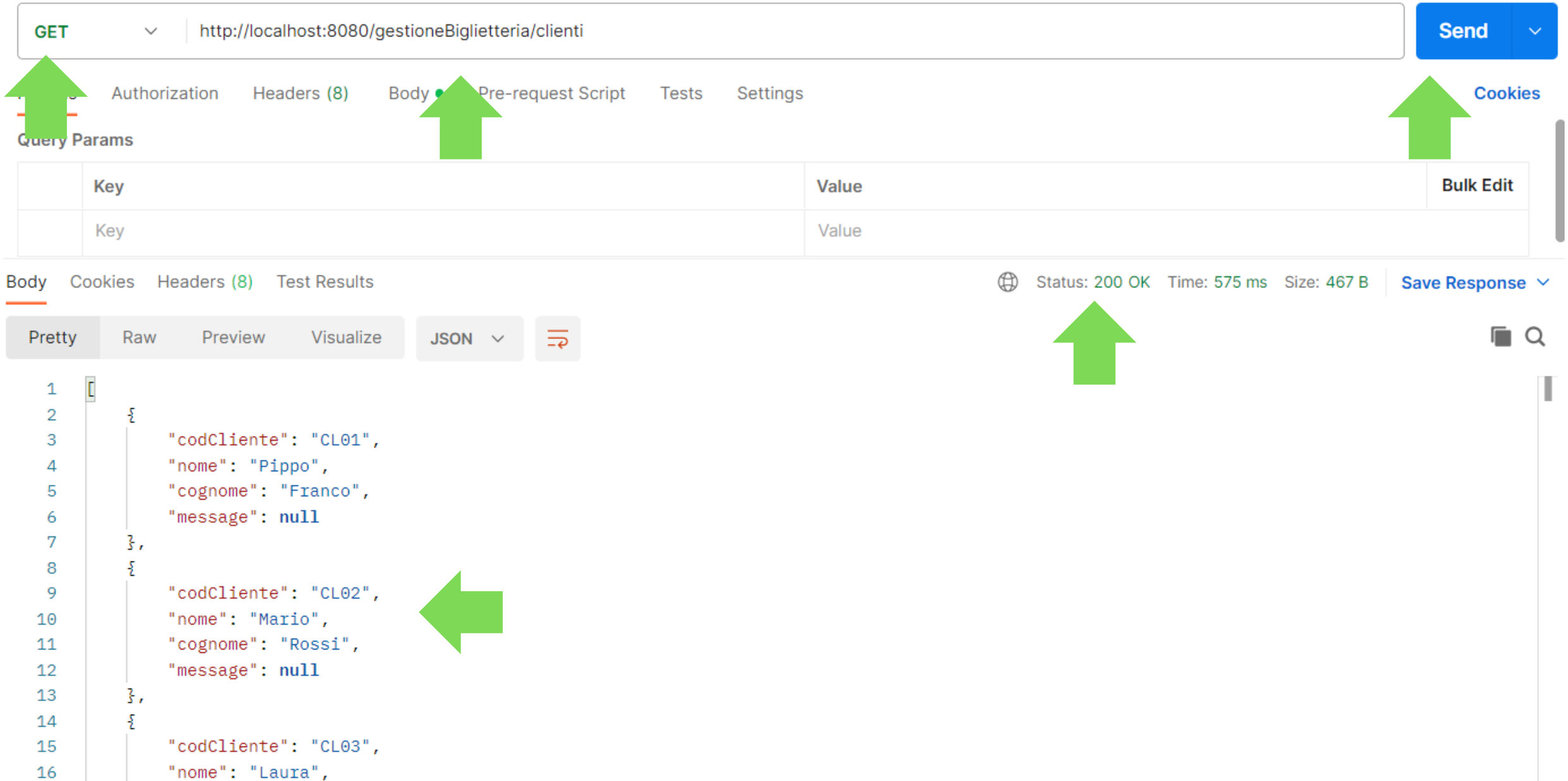
@RestController
@RequestMapping ("nomeMappatura")

Con l'annotazione @RequestMapping, dichiaro a Spring, quando viene fatta una chiamata con questa Stringa, posizionati sulla classe che ti ho identificato

Attenzione

Quando si fanno chiamate http attraverso il localhost, bisogna bypassare la cors policy, che non permette di recuperare dati dal localhost.
Lato server, quindi Springboot, basta annotare le classi controller con l'annotazione:
@CrossOrigin

TEST CON POSTMAN



PICCOLO RECAP DAL CODICE

```
@RestController
@RequestMapping ("gestioneBiglietteria")
@CrossOrigin
public class ClienteController {

    @Autowired
    private IClienteService cSrv;

    // all clienti
    @GetMapping(value= "clienti", produces= {MediaType.APPLICATION_JSON_VALUE})
    public ResponseEntity<List <ClienteInfo>> getAllClienti(){
        List <ClienteInfo> responseClienteList=new ArrayList<>();
        List<Cliente> clienteList= cSrv.getAllClienti();
        for(int i=0; i< clienteList.size(); i++) {
        ClienteInfo ob= new ClienteInfo();
        BeanUtils.copyProperties(clienteList.get(i), ob);
        responseClienteList.add(ob);
        }
        return new ResponseEntity<List<ClienteInfo>>(responseClienteList, HttpStatus.OK);
    }
```

UNO SGUARDO
AL CODICE JAVASCRIPT

```
const btnElencoVoli = document.getElementById('btnVoli');

function mostraTabellaVoli(){
fetch('http://localhost:8080/gestioneBiglietteria/voli')
.then(response => response.json())
.then(data => {
    let table = `
        <tr>
            <th>Nome volo</th>
            <th>Partenza</th>
            <th>Destinazione</th>
            <th>Data volo</th>
            <th>Orario volo</th>
        </tr>
    `;
```

RICETTA PER UNA BUONA CHIAMATA

- **Fetch: ovvero un metodo in grado di ricevere un risultato ricevuto lato server**
- **1 then: dichiarazione di cosa si vuole fare con il risultato ottenuto dalla fetch**
- **2 then: come vogliamo manipolare l'oggetto ottenuto**

```
const btnElencoVoli = document.getElementById('btnVoli');

function mostraTabellaVoli(){
fetch('http://localhost:8080/gestioneBiglietteria/voli')
.then(response => response.json())
.then(data => {
    let table = `
        <tr>
            <th>Nome volo</th>
            <th>Partenza</th>
            <th>Destinazione</th>
            <th>Data volo</th>
            <th>Orario volo</th>
        </tr>
    `;

    for(let i = 0; i < data.length; i++) {

        table +=`
            <tr>
                <td>${data[i].nomeVolo}</td>
                <td>${data[i].partenza}</td>
                <td>${data[i].destinazione}</td>
                <td>${data[i].dataVolo}</td>
                <td>${data[i].orarioVolo}</td>
            </tr>
        `;
    }
```

BENEDETTA APPROVED

GRAZIE PER L'ATTENZIONE