

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**

Possiamo 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

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/gestioneBiglietteria/clienti`. The **Send** button is highlighted with a green arrow. Below the request bar, the **Body** tab is selected, and the **Body** section is highlighted with a green arrow. The response status is **Status: 200 OK**, with **Time: 575 ms** and **Size: 467 B**. The **Save Response** button is also visible. The response body is displayed in **JSON** format, showing an array of three client objects. A green arrow points to the JSON data.

Request:

- Method: GET
- URL: `http://localhost:8080/gestioneBiglietteria/clienti`

Response:

```
[
  {
    "codCliente": "CL01",
    "nome": "Pippo",
    "cognome": "Franco",
    "message": null
  },
  {
    "codCliente": "CL02",
    "nome": "Mario",
    "cognome": "Rossi",
    "message": null
  },
  {
    "codCliente": "CL03",
    "nome": "Laura",
    "message": null
  }
]
```


PICCOLO RECAP DAL CODICE

```
21
22 @RestController
23 @RequestMapping ("gestioneBiglietteria")
24 @CrossOrigin
25 public class ClienteController {
26
27     @Autowired
28     private IClienteService cSrv;
29
30     // all clienti
31     @GetMapping(value= "clienti", produces= {MediaType.APPLICATION_JSON_VALUE})
32     public ResponseEntity<List <ClienteInfo>> getAllClienti(){
33         List <ClienteInfo> responseClienteList=new ArrayList<>();
34         List<Cliente> clienteList= cSrv.getAllClienti();
35         for(int i=0; i< clienteList.size(); i++) {
36             ClienteInfo ob= new ClienteInfo();
37             BeanUtils.copyProperties(clienteList.get(i), ob);
38             responseClienteList.add(ob);
39         }
40         return new ResponseEntity<List<ClienteInfo>>(responseClienteList, HttpStatus.OK);
41     }
42 }
```



**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>
```

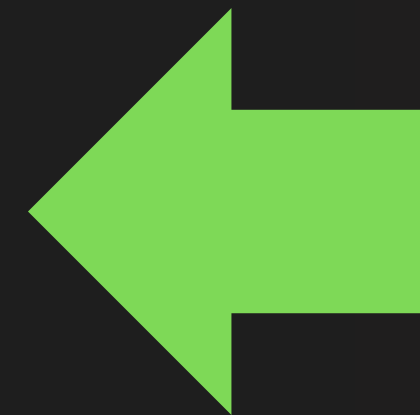
```
      `;
```

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**