

Sending Data and RequestBody

Now let's see how to create an API that accepts data from the client. We already have `addJob()` methods in our repository and service layers, so we just need to add an endpoint in our controller.

👉 `@RequestBody`

The `@RequestBody` annotation is crucial for receiving data in REST APIs:

- It converts JSON data from the request body into a Java object.
- Spring automatically maps JSON fields to matching properties in your Java class.
- No need to manually parse JSON - Spring does it for you.
- Data comes in as the exact object type you need.
- Implementing the Add Job API

```
● ● ●  
 @PostMapping("/jobPost")  
 public JobPost addJob(@RequestBody JobPost jobPost){  
     service.addJob(jobPost);  
     return service.getJob(jobPost.getId());  
 }
```

- `@PostMapping("/jobPost")` creates an endpoint that accepts POST requests
- `@RequestBody JobPost jobPost` converts the incoming JSON to a JobPost object
- The method adds the job and then returns the newly added job

Example:

```
● ● ●

package com.telusko.spring_boot_rest.controller;
import java.util.List;
import com.telusko.spring_boot_rest.model.JobPost;
import com.telusko.spring_boot_rest.service.JobService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

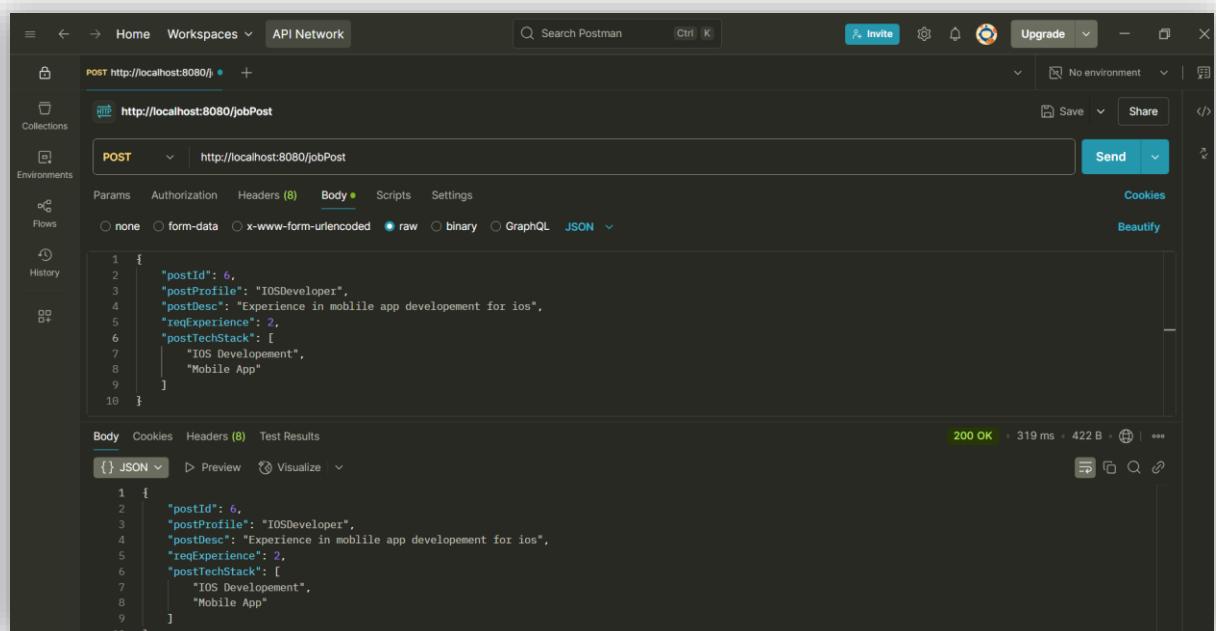
@RestController
@CrossOrigin(origins = "http://localhost:3000")
public class JobRestController {
    @Autowired
    private JobService service;

    @GetMapping("jobPosts")
    public List<JobPost> getAllJobs() {
        return service.getAllJobs();
    }

    @GetMapping("jobPost/{postId}")
    public JobPost getJob(@PathVariable("postId") int postId){
        return service.getJob(postId);
    }

    @PostMapping("/jobPost")
    public JobPost addJob(@RequestBody JobPost jobPost){
        service.addJob(jobPost);
        return service.getJob(jobPost.getPostId());
    }
}
```

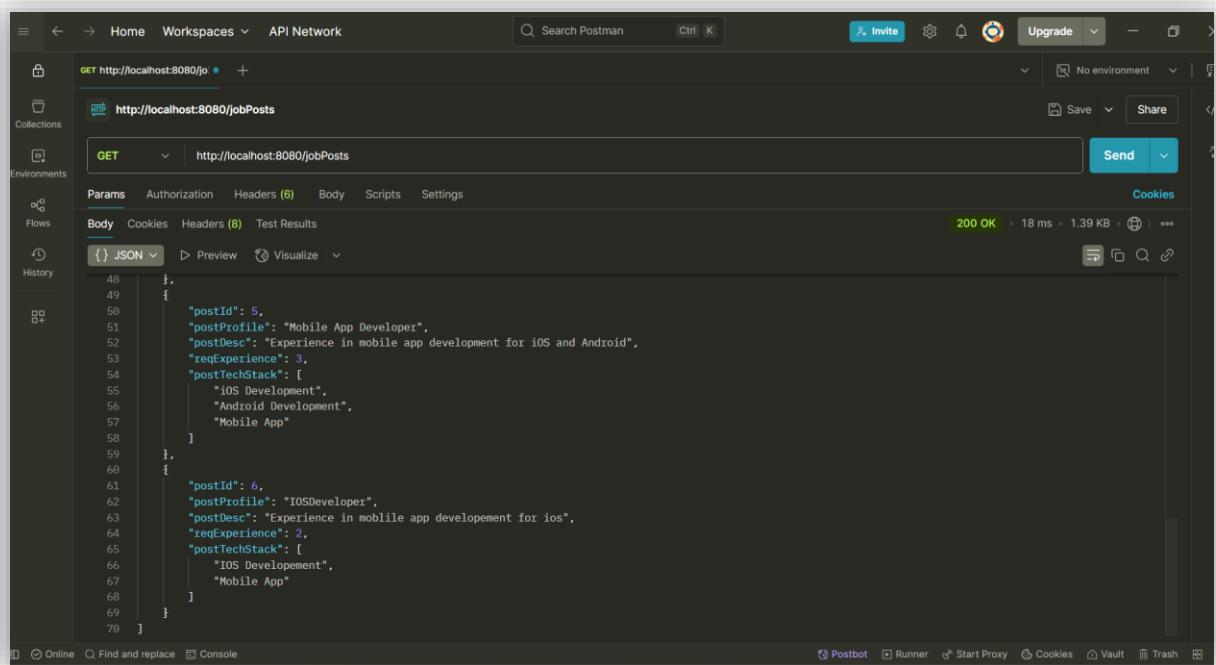
Output:



The screenshot shows the Postman interface with a successful POST request to `http://localhost:8080/jobPost`. The request body is a JSON object:

```
1 {
2     "postId": 6,
3     "postProfile": "IOSDeveloper",
4     "postDesc": "Experience in mobile app developement for ios",
5     "reqExperience": 2,
6     "postTechStack": [
7         "IOS Developement",
8         "Mobile App"
9     ]
10 }
```

The response status is 200 OK, with a response time of 319 ms and a response size of 422 B. The response body is identical to the request body.



The screenshot shows the Postman interface with a successful GET request to `http://localhost:8080/jobPosts`. The response body is a JSON array containing two objects, representing the job posts:

```
48 [
49     {
50         "postId": 5,
51         "postProfile": "Mobile App Developer",
52         "postDesc": "Experience in mobile app development for iOS and Android",
53         "reqExperience": 3,
54         "postTechStack": [
55             "IOS Development",
56             "Android Development",
57             "Mobile App"
58         ]
59     },
60     {
61         "postId": 6,
62         "postProfile": "IOSDeveloper",
63         "postDesc": "Experience in mobile app developement for ios",
64         "reqExperience": 2,
65         "postTechStack": [
66             "IOS Developement",
67             "Mobile App"
68         ]
69     }
70 ]
```