Put and Delete Mapping

👉 PutMapping

Updating Job with PUT Mapping

- When we want to update an existing job post (like changing its profile or required experience), we use the HTTP PUT method.
  - We send the complete updated job information to the server
  - The server finds the matching job by ID and updates all its fields
  - The updated job is returned as confirmation

- Create a controller method with @PutMapping

```
@PutMapping("jobPost")
public JobPost updateJob(@RequestBody JobPost jobPost){
    service.updateJob(jobPost);
    return service.getJob(jobPost.getPostId());
}
```

- Add a service method

```
public void updateJob(JobPost jobPost){
    repo.updateJob(jobPost);
}
```

- Implement repository logic

```
public void updateJob(JobPost jobPost){
    for(JobPost job:jobs){
      if(job.getPostId()==jobPost.getPostId()){
          job.setPostProfile(jobPost.getPostProfile());
          job.setPostDesc(jobPost.getPostDesc());
          job.setReqExperience(jobPost.getReqExperience());
          job.setPostTechStack(jobPost.getPostTechStack());
      }
    }
}
```

👉 DeleteMapping

Deleting Job with DELETE Mapping

- When we want to remove a job post completely, we use the HTTP DELETE method.
  - We specify which job to delete using its ID in the URL path
  - The server removes that job from the list
  - A confirmation message is returned

- Create a controller method with @DeleteMapping

```
@DeleteMapping("jobPost/{postId}")
public String deleteJob(@PathVariable int postId){
    service.deleteJob(postId);
    return "Deleted";
}
```

- Add a service method

```
public void deleteJob(int postId) {
    repo.deleteJob(postId);
}
```

- Implement repository logic

```
public void deleteJob(int postId) {
    for(JobPost jobPost : jobs){
        if(jobPost.getPostId() == postId){
            jobs.remove(jobPost);
        }
    }
}
```

Example:

👉 Controller

```java
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());
    }

    @PutMapping("jobPost")
    public JobPost updateJob(@RequestBody JobPost jobPost){
        service.updateJob(jobPost);

        return service.getJob(jobPost.getPostId());
    }

    @DeleteMapping("jobPost/{postId}")
    public String deleteJob(@PathVariable int postId){
        service.deleteJob(postId);

        return "Deleted";
    }
}
```

👉 Service

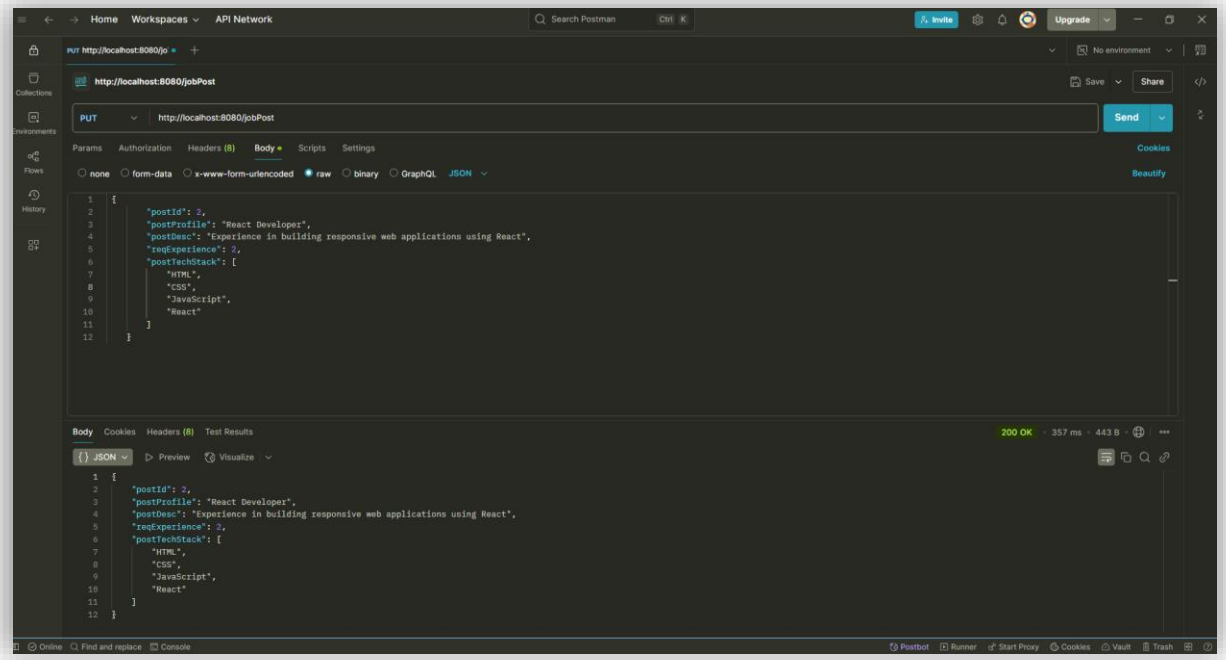
```java
package com.telusko.spring_boot_rest.service;

import com.telusko.spring_boot_rest.model.JobPost;
import com.telusko.spring_boot_rest.repo.JobRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class JobService {
    @Autowired
    public JobRepo repo;


    // method to add a jobPost
    public void addJob(JobPost jobPost) {
        repo.addJob(jobPost);

    }


    //method to return all JobPosts
    public List<JobPost> getAllJobs() {
        return repo.getAllJobs();

    }

    // method to return single JobPost by postId
    public JobPost getJob(int i){
        return repo.getJob(i);
    }

    // method to update a job post object
    public void updateJob(JobPost jobPost){
        repo.updateJob(jobPost);
    }

    // method to delete a job post object
    public void deleteJob(int postId) {
        repo.deleteJob(postId);
    }
}
```

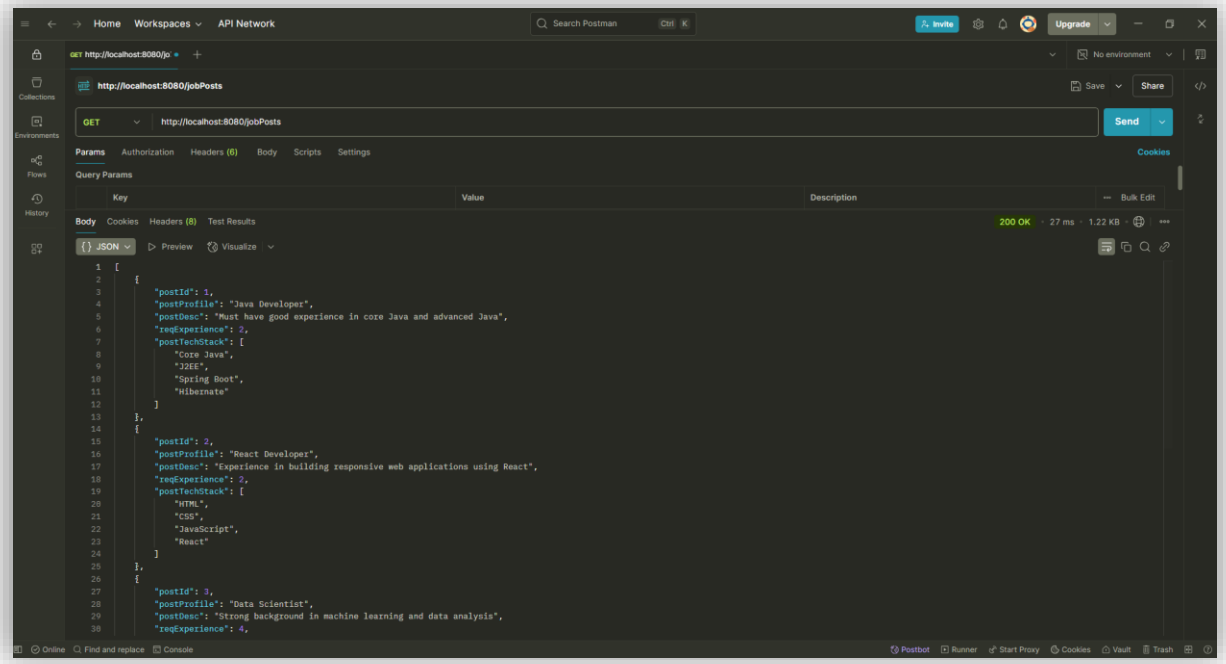
👉 Repository

```java
package com.telusko.spring_boot_rest.repo;

import com.telusko.spring_boot_rest.model.JobPost;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

@Repository
public class JobRepo {


  // ArrayList to store JobPost objects
  List<JobPost> jobs = new ArrayList<>(Arrays.asList(

    new JobPost(1, "Java Developer", "Must have good experience in core Java and advanced Java", 2,
            List.of("Core Java", "J2EE", "Spring Boot", "Hibernate")),


    new JobPost(2, "Frontend Developer", "Experience in building responsive web applications using React", 3,
            List.of("HTML", "CSS", "JavaScript", "React")),

    new JobPost(3, "Data Scientist", "Strong background in machine learning and data analysis", 4,
            List.of("Python", "Machine Learning", "Data Analysis")),


    new JobPost(4, "Network Engineer", "Design and implement computer networks for efficient data communication", 5,
            List.of("Networking", "Cisco", "Routing", "Switching")),


    new JobPost(5, "Mobile App Developer", "Experience in mobile app development for iOS and Android", 3,
            List.of("iOS Development", "Android Development", "Mobile App"))
  ));

  // method to return all JobPosts
  public List<JobPost> getAllJobs() {
       return jobs;
  }

  // method to save a job post object into arrayList
  public void addJob(JobPost job) {
       jobs.add(job);
       System.out.println(jobs);

  }

  // method to return single JobPost by postId
  public JobPost getJob(int i){

      for(JobPost job: jobs){

          if(job.getPostId()==i)
               return job;
      }

      return null;
  }

  // method to update a job post object
  public void updateJob(JobPost jobPost){

      for(JobPost job:jobs){
        if(job.getPostId()==jobPost.getPostId()){
             job.setPostProfile(jobPost.getPostProfile());
             job.setPostDesc(jobPost.getPostDesc());
             job.setReqExperience(jobPost.getReqExperience());
             job.setPostTechStack(jobPost.getPostTechStack());
         }
      }
  }

  // method to delete a job post object
  public void deleteJob(int postId) {
      for(JobPost jobPost : jobs){
          if(jobPost.getPostId() == postId){
               jobs.remove(jobPost);
          }
      }
  }
}
```

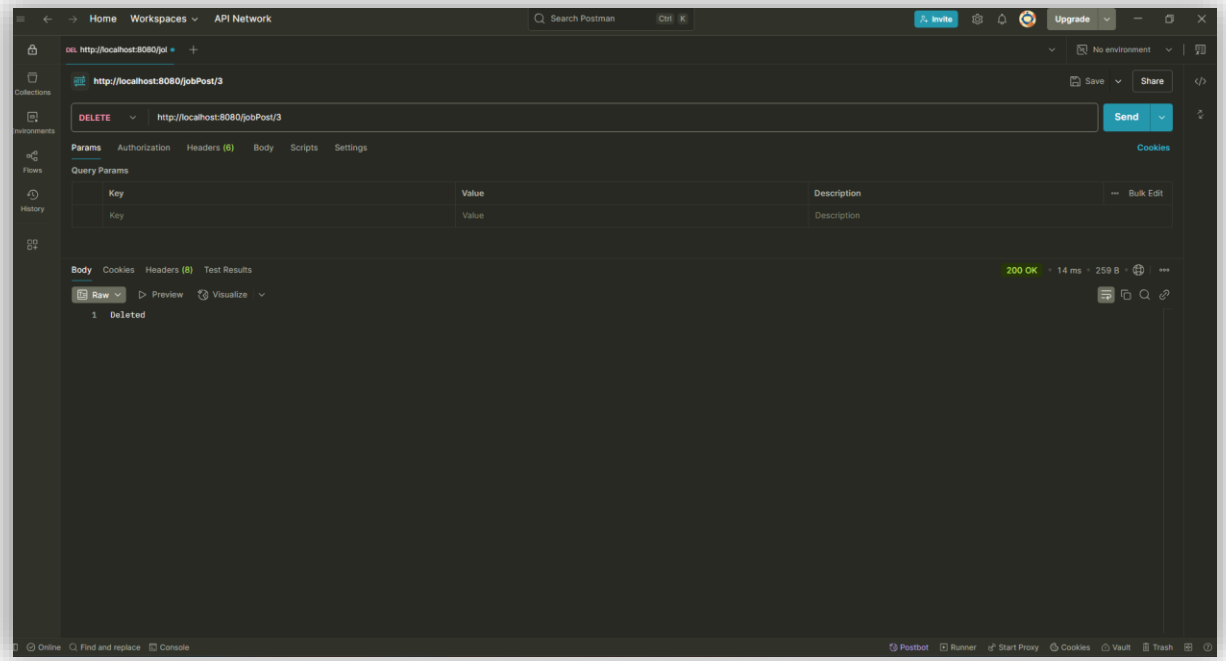
Output:

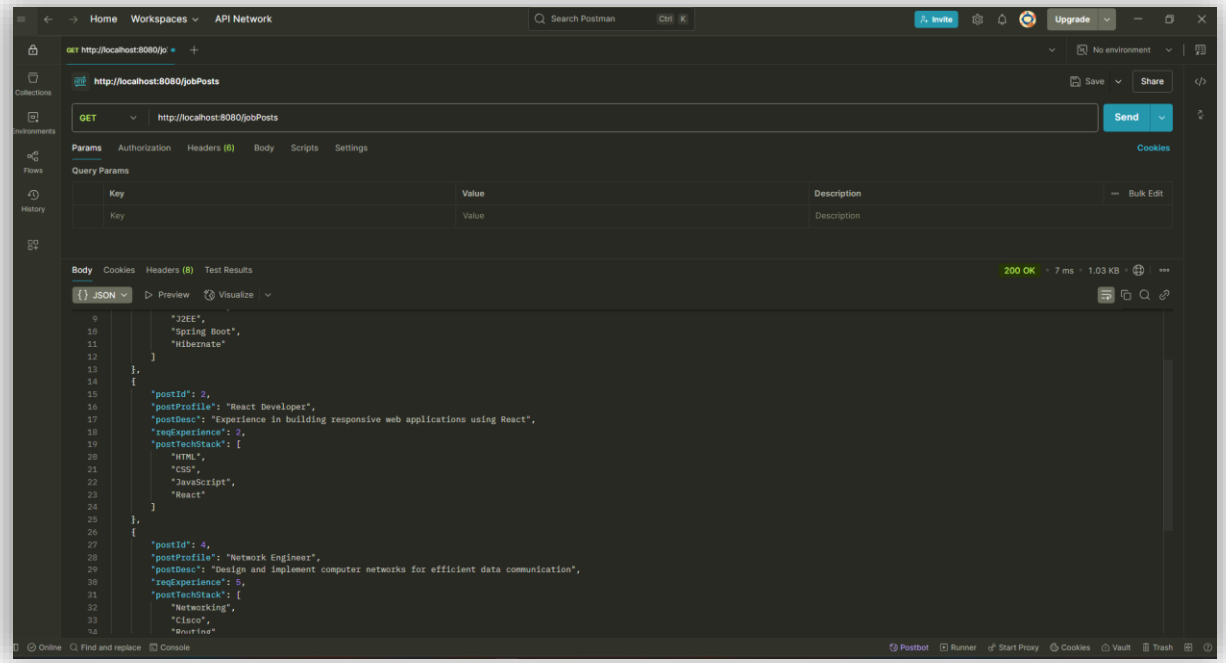
👉 **For Update (PUT):**





👉 **For Delete (DELETE):**





👉 Remember

- **PUT** is used for updating existing resources
- **DELETE** is used for removing resources
- With PUT, we send the complete object with all updated fields
- With DELETE, we typically just need the ID of the resource to remove
- Both operations follow the same pattern: Controller → Service → Repository