08 - JPA in Job App

Setting up JPA in a Job Application:
To implement JPA with PostgreSQL in a Job Management Application, the following steps involve configuring the project dependencies, repository, service, and REST API endpoints to handle CRUD operations for job postings.

1. Adding Dependencies:

Include JPA and PostgreSQL dependencies in the project's pom.xml file.

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
   <groupId>org.postgresql</groupId>
   <artifactId>postgresql</artifactId>
   <scope>runtime</scope>
</dependency>
```

2. Job Repository Interface:

Create a repository interface JobRepo that extends JpaRepository. This interface will automatically provide CRUD operations for JobPost entities.

```
@Repository
public interface JobRepo extends JpaRepository<JobPost, Integer> {

}
```

TELUSKO

3. JobRestController for Handling HTTP Requests:

- JobRestController leverages JPA through JobRepo, a repository extending JpaRepository, to handle CRUD operations without SQL.
- JPA automatically translates method calls (like save(), findById(), findAll(), and deleteById()) into SQL queries, making database interactions straightforward without needing to write SQL manually.
- JPA’s built-in CRUD methods (findAll(), findById(), save(), deleteById()) allow for straightforward data handling and reduce boilerplate code.
- JPA manages the persistence context, when JobRepo.save() is called in JobRestController, JPA decides whether to insert a new record or update an existing one based on the primary key.

```
@RestController
@CrossOrigin
public class JobRestController {
	@Autowired
	private JobService service;

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

	@GetMapping("/jobPost/{postId}")
	public JobPost getJob(@PathVariable 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";
	}
```

```
        @GetMapping("load")
        public String loadData() {
                service.load();
                return "success";
        }
}
```

4. JobService Layer:

- The JobService class handles the core business logic and communicates with JobRepo for database interactions.
- Each method in JobService corresponds to a controller method in JobRestController, such as fetching all jobs, getting a job by ID, adding, updating, or deleting job posts.

```
@Service
public class JobService {

    @Autowired
    public JobRepo repo;

    public List<JobPost> getAllJobs() {
        return repo.findAll();
    }

     public void addJob(JobPost jobPost) {
        repo.save(jobPost);
    }

    public JobPost getJob(int postId) {
        return repo.findById(postId).orElse(new JobPost());
    }

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

    public void deleteJob(int postId) {
        repo.deleteById(postId);
    }
    public void load() {
        List<JobPost> jobs = new ArrayList<>(List.of(
            new JobPost(1, "Software Engineer", "Exciting opportunity for a skilled software
            engineer.", 3, List.of("Java", "Spring", "SQL")),
           new JobPost(2, "Data Scientist", "Join our data science team and work on cutting-edge
           projects.", 5, List.of("Python", "Machine Learning", "TensorFlow")),
           new JobPost(3, "Frontend Developer", "Create amazing user interfaces with our talented
           frontend team.", 2, List.of("JavaScript", "React", "CSS")),
           new JobPost(4, "Network Engineer", "Design and maintain our robust network
           infrastructure.", 4, List.of("Cisco", "Routing", "Firewalls")),
           new JobPost(5, "UX Designer", "Shape the user experience with your creative design
           skills.", 3, List.of("UI/UX Design", "Adobe XD", "Prototyping"))

      ));
     repo.saveAll(jobs);
    }
}
```

Code Link :
https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/9%20Spring%20DataJPA/9.8%20Jpa%20In%20Job%20App