Sarthak Pathak

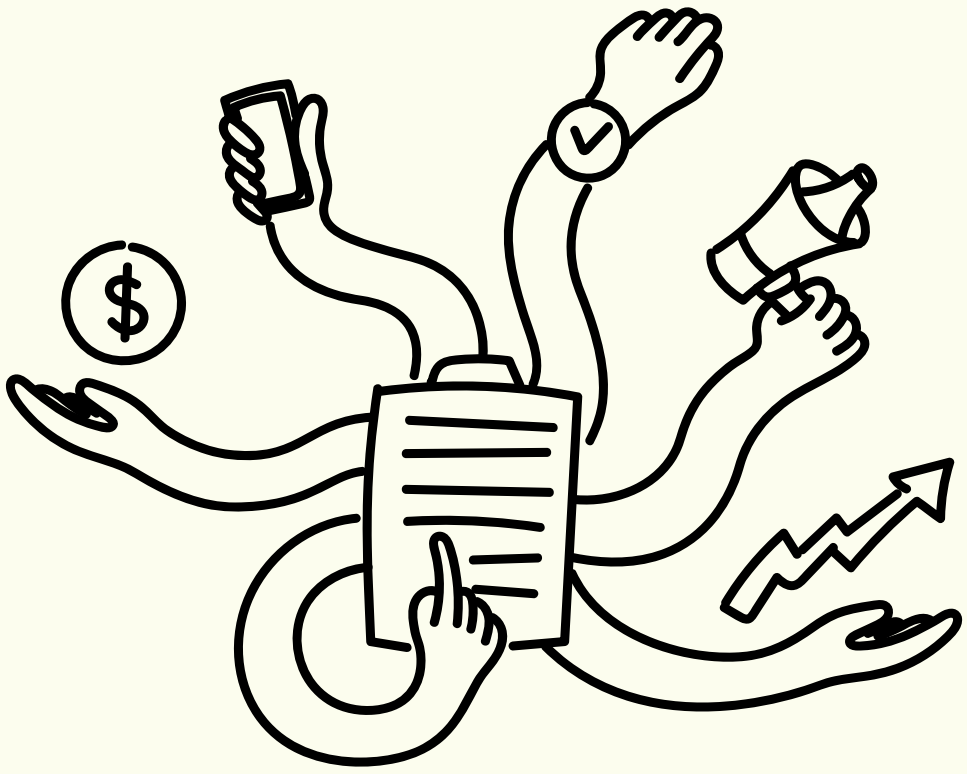
Mastering Spring Boot

Inheritance in JPA

Swipe for more

Sarthak Pathak

Inheritance is a fundamental OOP concept, and JPA provides powerful ways to map it to your database. Understanding JPA inheritance strategies is crucial for designing efficient and maintainable data models in Spring Boot applications. Let's explore the key concepts!



Swipe for more

Sarthak Pathak

JPA offers three main inheritance strategies:

1. **Single Table**
2. **Joined Table**
3. **Table Per Class**

Single Table is often the most performant but can lead to unused columns.

Joined Table provides the cleanest data model but may require joins.

Table Per Class offers simplicity but can complicate queries.

Swipe for more

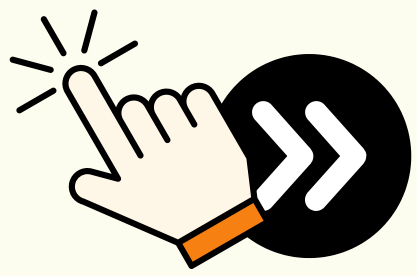
Sarthak Pathak

```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "employee_type")
public abstract class Employee {
    @Id
    private Long id;
    private String name;
}

@Entity
@DiscriminatorValue("manager")
public class Manager extends Employee {
    private BigDecimal bonus;
}
```

This code demonstrates the Single Table strategy. All subclasses are stored in one table, distinguished by a discriminator column. It's simple and performs well for most scenarios.

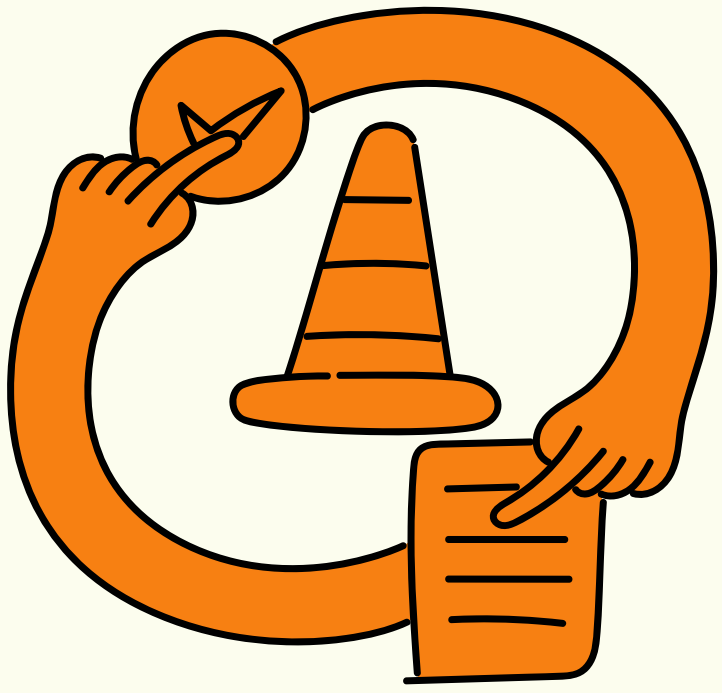
Swipe for more



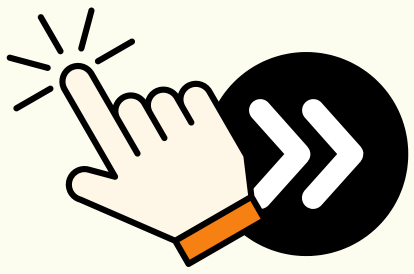
Sarthak Pathak

Choose your inheritance strategy based on your specific use case and performance requirements. Consider future scalability!

What's your preferred JPA inheritance strategy and why? Share your thoughts in the comments! 💬



Swipe for more



Sarthak Pathak

Helpful? Share and spread knowledge! Your repost grows our community.

Not Reels