Best Practices

Mastering @Transactional in Spring Boot

Palmurugan C
@palmuruganc

What is @Transactional?

@Transactional is a Spring annotation used to manage database transactions. It ensures that a group of operations either complete successfully or are rolled back on failure.

✅ All-or-Nothing

✅ Reduces boilerplate

✅ Declarative transaction management

Example

```
@Service
public class OrderService {

    @Transactional
    public void placeOrder(Order order) {
        saveOrder(order);
        updateInventory(order);
    }
}
```

If any exception occurs, the entire placeOrder transaction will roll back.

@palmuruganc

Use on Public Methods Only

@Transactional only works on public methods.
Spring uses proxies that don’t apply to private or protected methods.

❌ **Incorrect**

```
@Transactional
private void internalMethod() {}
```

✅ **Correct**

```
@Transactional
public void processTransaction() {}
```

Avoid Self-invocation

❌ Calling a ***@Transactional*** method from within the same class won't trigger transaction behavior.

```
public void outer() {
    inner(); // Won't be transactional
}

@Transactional
public void inner() {}
```

✅ Solution

Move the transactional method to another bean/service.

Don't Use on Read-Only Methods

❌ Adding ***@Transactional*** on read-only methods adds unnecessary overhead.

```
@Transactional
public List<Order> getAllOrders() {}
```

✅ Use @Transactional(readOnly = true) if needed

```
@Transactional(readOnly = true)
public List<Order> getOrders() {}
```

Choose the Right Propagation

Use appropriate propagation types depending on the use case.

- REQUIRED (default): Join current or create new
- REQUIRES_NEW: Always start new transaction
- NESTED: Savepoint mechanism

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void auditLog() {}
```

Rollback Rules

By default, Spring only rolls back on unchecked exceptions (RuntimeException).

💡 To roll back on checked exceptions, specify explicitly:

```
@Transactional(rollbackFor = IOException.class)
public void readFileAndSave() throws IOException {}
```

Use with Caution in Async Methods

🚫 ***@Transactional*** doesn't work well with **@Async** out of the box.

Why? Async runs in a different thread → proxy-based transaction lost.

✅ Use **TransactionTemplate** if async transactions are needed.

Summary

- 📌 Use on public methods only
- 📌 Avoid self-invocation
- 📌 Use correct propagation
- 📌 Declare rollback for checked exceptions
- 📌 Avoid on read-only unless necessary
- 📌 Be cautious with *@Async*

Follow Me

"Stay ahead with cutting-edge technical tips and best practices—follow me on LinkedIn today!"

@palmuruganc