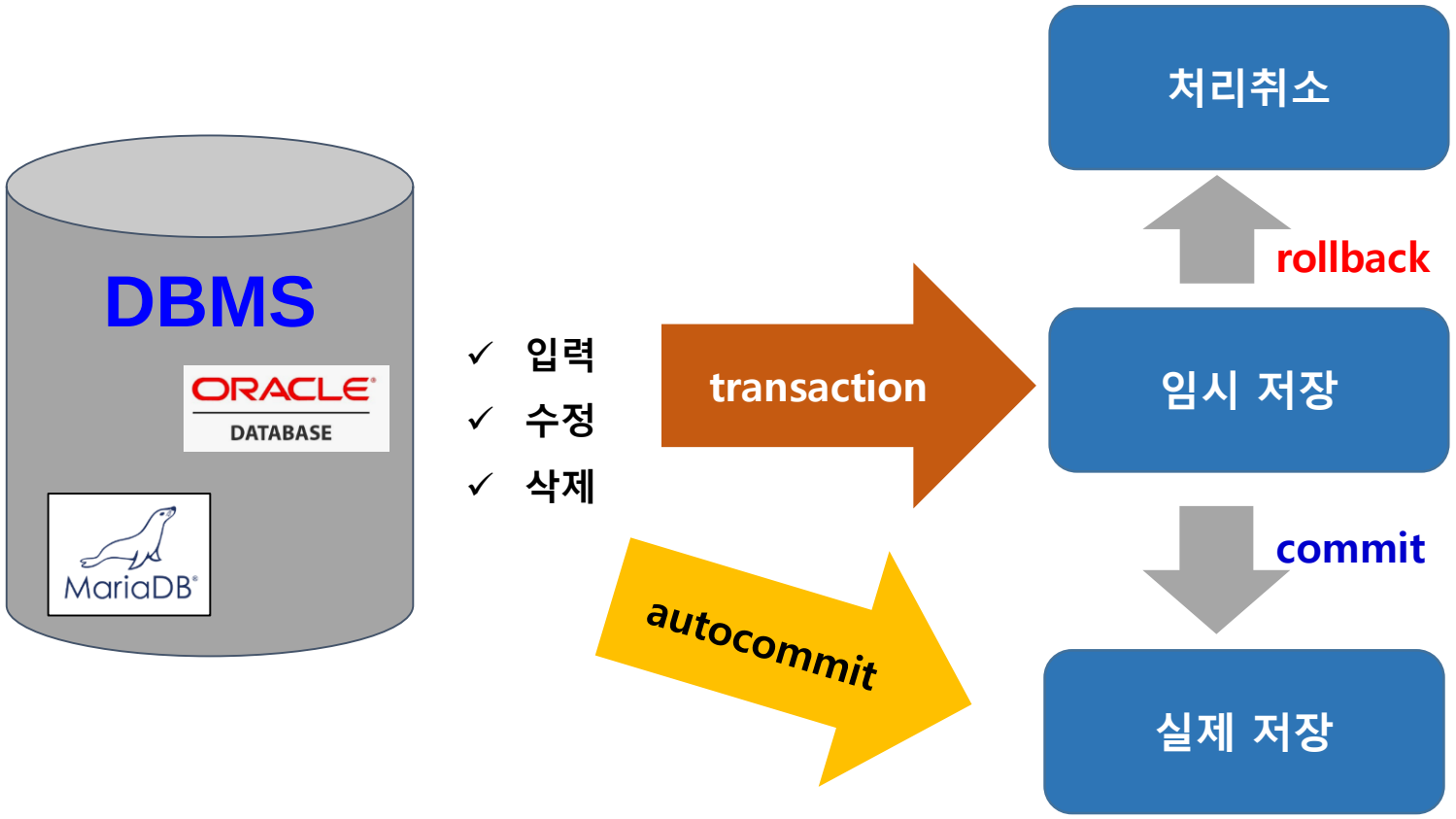

Spring boot 항해를 같이 떠나요 ~
front
back
Html, Js
Java
Spring boot
made by daughter

섹션11-1. DB 트랜잭션에 대한 이해

[Transaction – 거래 , 매매 (deal)]



Why? Transaction

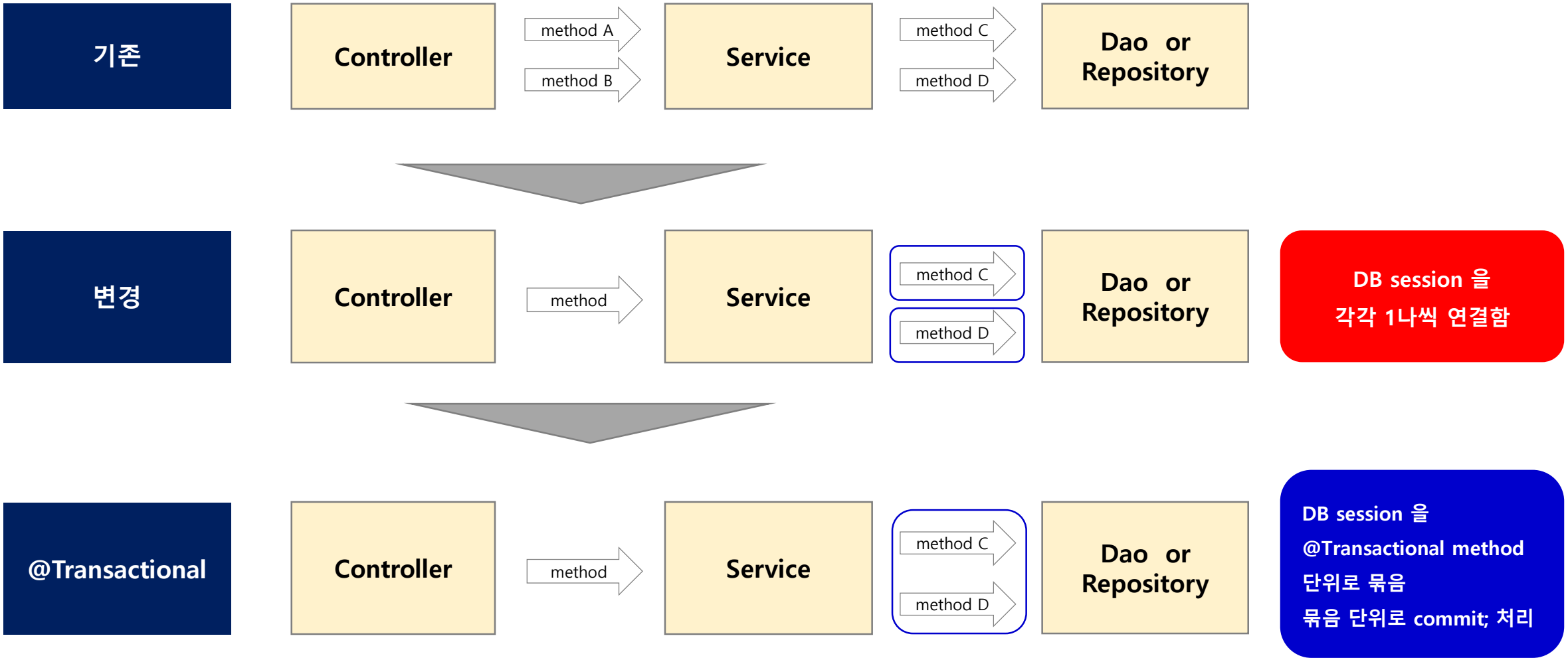
- ✓ 입금은 했는데, 통장에 기재가 안됨
- ✓ 온라인 입금하다가 인터넷이 끊김
- ✓ ATM 에서 카드로 현금을 찾는데
  현금은 없고 돈만 지출됨
- ✓ 당근거래 나갔는데 물건만 받고 돈을 안줌
- ✓ 시장에 가서 물건을 사는데 돈만 받고 물건을 안줌

Mariadb (기본 autocommit - Y) : **show variables LIKE 'autocommit%';**

Oracle (기본 autocommit - N) : **show autocommit;**

섹션12-1. @Transactional 의 역할

@Transactional - https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/annotation/Transactional.html



섹션12-2. @Transactional 에서 Exception (Checked vs Unchecked)

@Transactional 적용 시 RuntimeException or Error 인 경우만 Rollback 처리를 한다.

Checked Exception

```
File file = new File("not_existing_file.txt");
    FileInputStream stream
        = new FileInputStream(file);
```

예측 가능한 오류로
컴파일단계에서 예외처리를 강제함

Unchecked Exception

```
int numerator = 1;
int denominator = 0;
int result = numerator / denominator;
-> ArithmeticException
```

개발자의 실수로 발생하는
예상치 못한 예외 발생

컴파일러가 예외처리를 확인하지 못함

참고 사이트 : https://www.baeldung.com/java-checked-unchecked-exceptions

섹션12-3. Transactional 에서 Exception 처리 (CustomException)

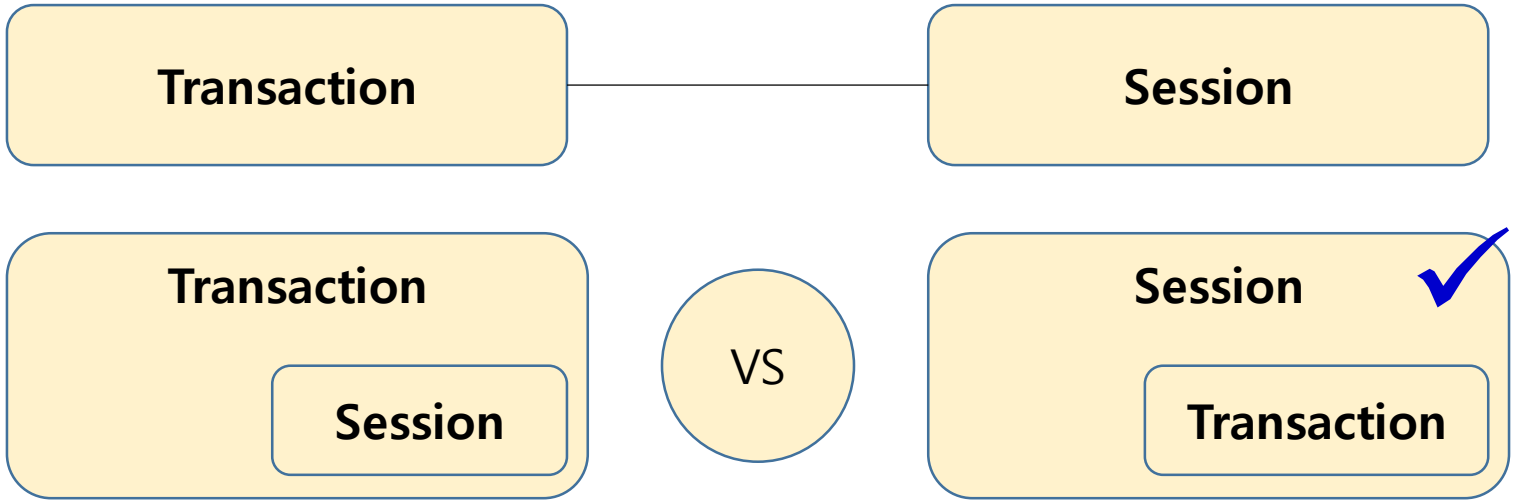
※. CustomException

```
/**
 * CustomException 생성, Exception 처리는 나에게..
 */
@Log4j2
public class MyExceptionRuntime extends RuntimeException{

    public MyExceptionRuntime(String s){
        super("My Exception 처리 =>"+s);
    }

    public MyExceptionRuntime(String s, String sClass){
        super("My Exception 처리 =>"+s);
        log.info("MyException Class 오류발생 =>"+sClass);
    }
}
```

@Transactional(propagation = Propagation.REQUIRED)

REQUIRED – Default : 부모 트랜잭션에 의지, 없다면 새로운 트랜잭션을 생성

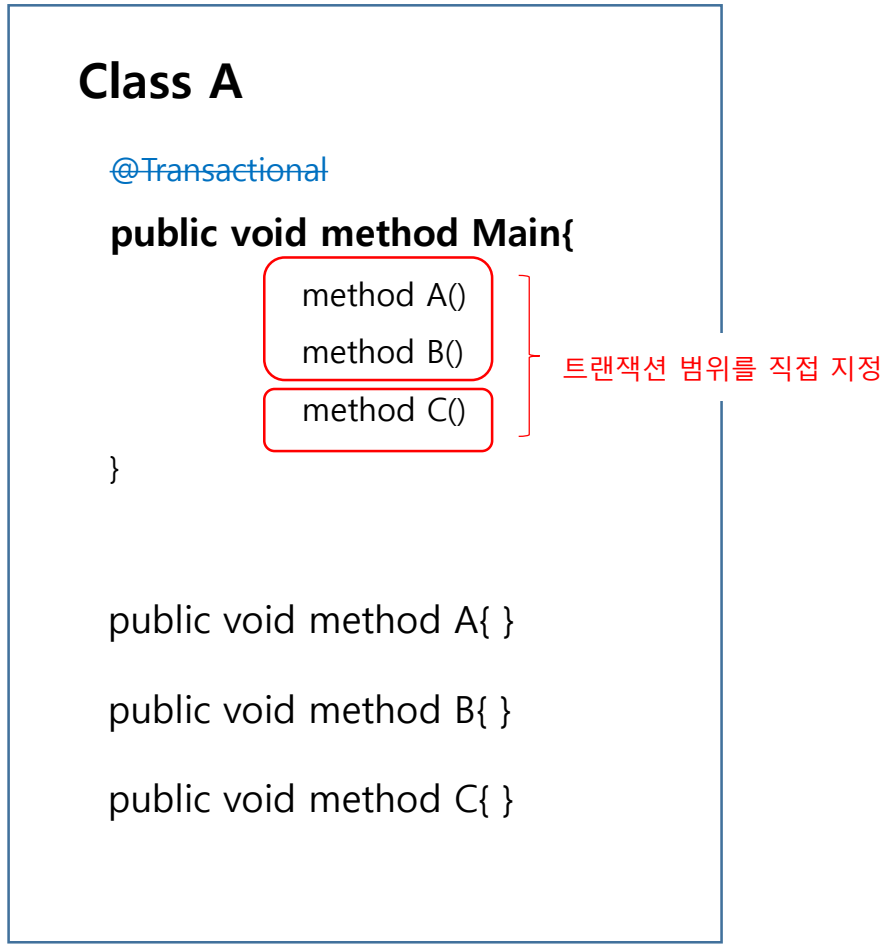
REQUIRES_NEW : 부모 트랜잭션에 상관없이 새로운 트랜잭션을 생성한다.



섹션12-5. PlatformTransactionManager 트랜잭션을 수동으로 직접 관리하자

트랜잭션 수동처리하기 PlatformTransactionManager

- commit , rollback 은 내가 직접 처리한다.



사용구문

```
// 트랜잭션매니저 객체를 불러온다.
@Autowired
PlatformTransactionManager transactionManager;

// 트랜잭션에 사용할 속성을 불러온다.
@Autowired
TransactionDefinition definition;

// 트랜잭션을 얻어와 상태를 보관한다.
TransactionStatus status = transactionManager.getTransaction(definition);

// commit or rollback 처리를 하고 세션을 반환한다.
transactionManager.commit(status);
```

TransactionTemplate

(feat. PlatformTransactionManager)

참고 사이트 Baeldung - https://www.baeldung.com/spring-programmatic-transaction-management