트랜잭션 처리
- @Repository, @Transactional -

❖ 트랜잭션 처리의 필요성

- BoardServiceImpl

```
@Override
public void create(Board board) throws Exception {
   int result = dao.insert(board);
   if(result == 1) {
      throw new RuntimeException("test");
   }

   for(Attachment attachment : board.getAttachList()) {
      attachment.setAttachmentId(43); // 무조건 발생
      attachment.setBoardId(board.getBoardId());
      attachmentDao.insert(attachment);
   }
}
```

트랜잭션 처리

❖ @Tansactional

- 데이터베이스 쿼리 실행시
  - 성공하면 commit 처리
  - 예외 발생시 rollback 처리
- 주의 사항
  - 체크예외 발생시 처리하지 않음
  - RuntimeException 예외 발생시 처리
  - DB 관련 예외는 Exception의 자식 클래스
  - <tx:annotation-driven> 설정을 servlet-context.xml에서 등록
    - 동일 수준에서 등록된 빈에 대해서만 적용됨

- servlet-context.xml

```
    :
<context:component-scan base-package="edu.iot" />
<tx:annotation-driven transaction-manager="transactionManager" />
```

❖ @Repository 어노테이션

- 자동 빈 등록 처리
- DB 예외를 RuntimeException 기반의 DataAccessException으로 변환
  - @Transactional이 동작할 수 있게끔 해줌
- MyBatis에 의해 자동 생성되는 Dao 구현 클래스에 자동 추가됨
- 직접 Dao 구현 클래스를 정의하는 경우 @Repository 추가

트랜잭션 처리

❖ BoardServiceImpl

- 단일 쿼리 실행은 auto commit 메커니즘으로 처리
- 다중 쿼리 실행 메서드에 @Transactional 추가
  - BoardDao와 AttachmentDao가 같이 사용하는 메서드
  - create
  - update
  - delete

트랜잭션 처리

❖ BoardServiceImpl

```
@Transactional
@Override
public void create(Board board) throws Exception {
   int result = dao.insert(board);

   for(Attachment attachment : board.getAttachList()) {
      attachment.setBoardId(board.getBoardId());
      attachmentDao.insert(attachment);
   }
}
```

트랜잭션 처리

❖ BoardServiceImpl

```
@Transactional
@Override
public void update(Board board) throws Exception {
   int result = dao.update(board);
   if(result ==0) {
      throw new PasswordMissmatchException();
   }

   for(Attachment attachment : board.getAttachList()) {
      attachment.setBoardId(board.getBoardId());
      attachmentDao.insert(attachment);
   }
}
```

트랜잭션 처리

❖ BoardServiceImpl

```
@Transactional
@Override
public void delete(Board board) throws Exception {
   Board b = dao.findById(board.getBoardId());
   if(!b.getPassword().equals(board.getPassword())) {
      throw new PasswordMissmatchException();
   }

   attachmentDao.deleteByBoardId(board.getBoardId());
   dao.delete(board.getBoardId());
}
```

❖ 체크 예외 발생시 처리

- rollbackFor 속성
  - Rollback을 할 클래스 지정
  - @Transactional(rollbackFor=예외_클래스.class)
- noRollbackFor 속성
  - Rollback 제외 예외 클래스 지정
  - @Transactional(noRollbackFor=예외_클래스.class)