[문제 1] MyBatis 애노테이션 기반 Mapper 를 활용한 사원 등록 및 목록 조회

구현하려고 한다. Emp Table

요구사항

- emp 테이블에 사원명을 등록할 수 있어야 한다.
  - INSERT 작업은 @Insert 애노테이션을 사용하여 구현할 것
- 등록된 사원명을 모두 조회할 수 있어야 한다.
  - SELECT 작업은 @Select 애노테이션을 사용하여 구현할 것
- XML 매퍼 파일은 사용하지 않는다.
  - 오직 **애노테이션 기반 Mapper 인터페이스만 활용**
- 콘솔에서 main() 메서드를 통해 테스트할 수 있어야 한다.
  - 등록 → 조회 → 결과 출력까지 하나의 실행 흐름에서 확인

프로젝트 구성 가이드

```
SpringWorkORM05/
├── pom.xml
├── src/
│   └── main/
│       ├── java/
│       │   └── com/
│       │       └── test01/
│       │           ├── EmpMapper.java          ← MyBatis Mapper 인터페이스
│       │           └── EmpMapperTest.java      ← 테스트 클래스
│       └── resources/
│           ├── config/
│           │   ├── applicationContext.xml      ← Spring 설정
│           │   └── mybatis-config.xml          ← MyBatis 설정
```

주요파일

파일명	위치	설명
EmpMapper.java	com.test01	@Insert, @Select 애노테이션을 사용하는 MyBatis Mapper 인터페이스
EmpMapperTest.java	com.test01	main() 메서드를 통해 Mapper 테스트 수행
applicationContext.xml	resources/config/	데이터소스, SqlSessionFactory, Mapper 스캔 설정 포함
mybatis-config.xml	resources/config/	MyBatis 설정 (소문자 ↔ 카멜케이스 매핑 등)
pom.xml	루트	MyBatis + Spring + MySQL 의존성 관리

실행결과

```
사원 등록 완료!
사원 목록:
홍길동
김철수
```

[문제 2] SqlSessionFactory 를 직접 이용한 수동 트랜잭션 제어를 구현하시오

요구사항

- MyBatis 의 SqlSessionFactory 를 이용하여 SqlSession 을 직접 생성하고 DB 작업을 수행하시오.
  - 자동 트랜잭션이 아닌 **수동 트랜잭션 제어(commit/rollback)** 방식으로 insert 를 구현할 것
- EmpMapper.java 를 통해 사원 이름을 등록할 수 있어야 한다.
  - insert 후 예외가 발생하면 rollback, 그렇지 않으면 commit 되도록 구현할 것
- try-with-resources 문법을 사용하여 SqlSession 을 안전하게 관리할 것
- 등록된 사원 목록을 조회하여 콘솔에 출력할 수 있어야 한다.

프로젝트 구성가이드

```
SpringWorkORM05/
├─── src/
|    └─── main/
|         ├─── java/
|         |    └─── com/
|         |         └─── test02/
|         |              ├─── EmpService.java          ← SqlSession 수동 트랜잭션 서비스
|         |              └─── EmpServiceTest.java      ← 테스트 클래스
```

주요파일

파일명	위치	설명
EmpService.java	com.test02	SqlSessionFactory 로 수동 트랜잭션 제어, insert 구현
EmpServiceTest.java	com.test02	예외 발생 여부에 따라 commit 또는 rollback 동작 테스트

기존 EmpMapper.java	com.test01 또는 공통	그대로 재사용 가능 (애노테이션 기반)

실행 예시 흐름

```
service.insertEmp("박영희"); // 정상 → commit
service.insertEmpWithError("에러유발"); // RuntimeException 발생 → rollback
```

실행결과

```
[INSERT 시도] 이름: 박영희
[COMMIT 완료]

[INSERT 시도] 이름: 에러유발
[ROLLBACK 발생] org.mybatis.MyBatisSystemException: ...
```

트랜잭션 흐름 요약

```
SqlSession session = factory.openSession(false); // autoCommit: false
try {
    mapper.insertEmp("홍길동");
    session.commit();
} catch (Exception e) {
    session.rollback();
}
```

[문제 3] @Transactional 을 이용한 선언적 트랜잭션 처리하시오

요구사항

- Spring 의 @Transactional 어노테이션을 사용하여 **선언적 트랜잭션 처리**를 구현하시오.
  - 트랜잭션 설정은 applicationContext.xml 에서 <tx:annotation-driven />을 통해 활성화할 것
- EmpService 클래스의 insertThenFail() 메서드는 다음을 수행해야 한다.
  - 사원 이름을 insert 한 후 **강제로 예외를 발생시켜 롤백되는지 확인**할 것
- 예외 발생 시 DB 에 해당 사원이 저장되지 않아야 한다.
- 콘솔에서 트랜잭션 동작 여부(rollback 포함)를 확인할 수 있도록 로그 출력 포함
- SqlSessionTemplate 을 사용하여 EmpMapper 인터페이스를 호출할 것

디렉토리 구조

```
SpringWorkORM05/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/
│       │       └── test03/
│       │           ├── EmpService.java          ← @Transactional 적용된 서비스
│       │           └── EmpServiceTest.java      ← 예외 발생 테스트 클래스
```

주요 파일

파일명	위치	설명
EmpService.java	com.test03	@Transactional 로 트랜잭션 관리
EmpServiceTest.java	com.test03	정상 호출 → 예외 발생 → 롤백 테스트
applicationContext.xml	config/	<tx:annotation-driven /> 포함 필수

EmpService 핵심 구현 예시

```
@Transactional
public void insertThenFail(String name) {
    empMapper.insertEmp(name);
    System.out.println("INSERT 성공 후 예외 발생 예정");
    throw new RuntimeException("강제 예외");
}
```

실행 흐름 예시

```
service.insertThenFail("실패될이름");
```

실행결과

```
INSERT 성공 후 예외 발생 예정
Exception in thread "main" java.lang.RuntimeException: 강제 예외
→ DB 에 '실패될이름'은 저장되지 않음 (rollback)
```