Spring JDBC

polymorphism에 의해서 1번 Servlet을 사용하거나 2번 Servlet을 사용하던지 WAS가 하는 일은 항상 같다. 부모타입을 불러와서 실행 객체는 자식타입

DBConfig.java를 생성

DataSource를 생성해서 Bean에 등록해야 한다.

RowMapper

* 일치하지 않는 필드명을 매칭 시켜주는 작업

RootApplicationContextConfig

* Business Login 에 대한 설정만 해준다.
* dao, service에 대한 정보를 알려준다.
* ComponentScan할 패키지를 명시 해준다.
* DB연결에 대한 설정을 알려 주어야 Dao를 사용 할 수 있다.

DBConfig

* value들을 다른 property에서 읽어와서 넣어주기 때문에 propertySource만 변경해주면 된다.
* Bean들을 등록
* Property객체 구조
  1. key=value
  2. java에 property객체가 존재한다.
* return new DataSourceTransactionManager(dataSource())
  1. dataSource는 xml에서 Bean을 등록할 때 ‘ref’를 사용하는 것과 같은 방식이다.

@ContextConfiguration(classes = RootApplicationContextConfig.class)

* 해당 annotation을 통해서 Factory를 위의 class를 통해서 생성해달라는 요청을 한다.

Dao를 구현할 때, Dao로 접근하는 Interface를 추가 할 것인지 아닌지는 프로젝트 팀별로 합의해서 결정한대로 진행한다.

DataSourceTransactionManager를 통해서 Database에서 해 주어야 할 Transaction을 대신 해준다.

참고 URL: <http://wikibook.co.kr/article/transaction-management-using-spring/>

* 통합 테스트 과정에서 테스트를 실행할 때 마다 Database Table에 데이터가 1건씩 늘어나는데 이를 DB의 트랜잭션을 이용해서 해결 할 수 있다.
* 트랜잭션을 시작한 후에 실행되는 쿼리는 해당 트랜잭션이 롤백될 때는 모두 취소되는데 테스트를 시작하기 전에 트랜잭션을 시작하고, 종료 전에 롤백을 시킨다면 테스트 도중에 만든 데이터는 없어진다.
* Spring에서는 이와 같은 트랜잭션을 이용해서 테스트 실행 중에 이뤄진 DB조작을 취소한느 기능을 제공한다.

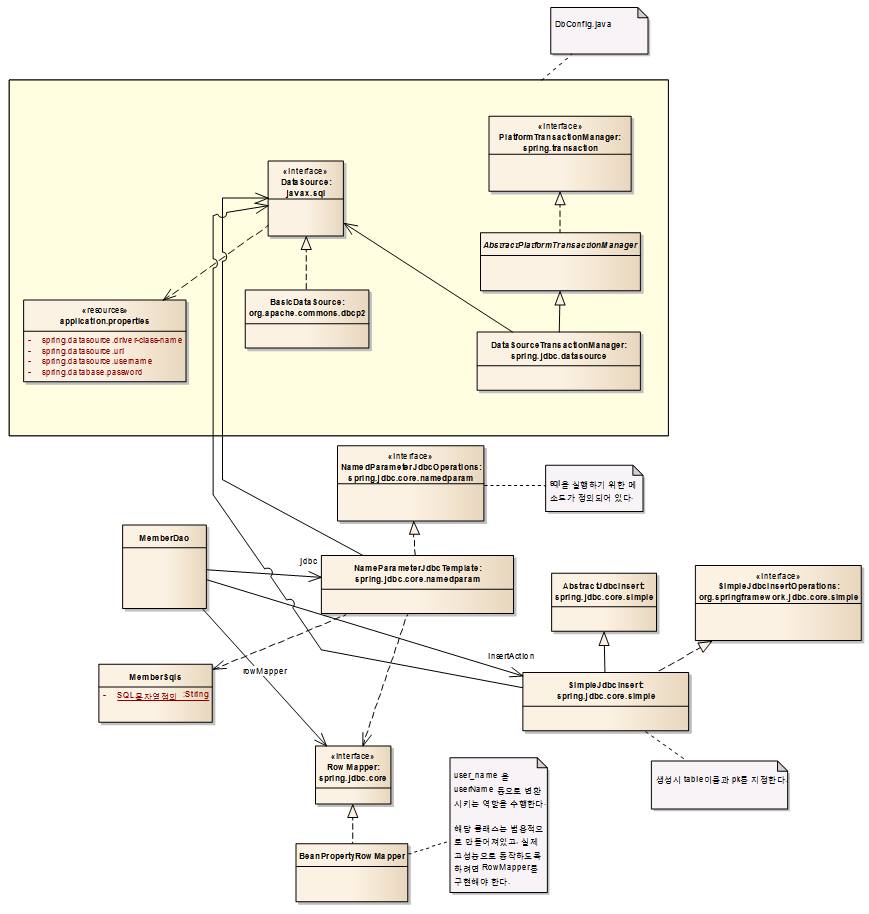
Member Class는 domain 패키지에 넣어서 주로 사용한다.

Dao Class에서 @Repository를 입력해서 Bean에 등록시켜주도록 붙여준다. 자동 객체 생성

public MemberDao 생성자에서 현재 객체를 생성해서 만들어 주는 방법을 사용하는데 Bean을 이용해서 등록 할 수 있다. (직접 Bean에 등록해서 사용해볼 것) -> @Autowired

queryForObject -> row 하나 가져올 때 사용

queryForList -> 여러 row를 가져올 때 사용



* 위의 그림 중 아래쪽의 MemberDao, MemberSqls를 제외하고 나머지 부분은 Spring에서 제공하는 기능을 사용한다.

addResourceHandlers

* 해당 함수를 통해서 resource에 대한 설정을 해주어야 된다.
* 해당 설정을 하면 Dispatcher Servlet이 처리하지 않고 따로 처리된다.
* 이미지 불러오는 요청에서 사용

Transaction을 Service에 처리해 주어야 한다.

jsp -> el표기법

redirect뒤에는 요청 URL을 적어야 한다.

스크립트릿 대신 el표기법을 사용하면 Java 코드가 직접 노출되지 않기 때문에 유리하다.

custom tag

jstl을 사용 -> 자바코드를 숨겨서 사용한다는 의미

MVC의 취지가 분리하자 이기 때문에 가급적 자바 코드가 나오지 않는 것이 좋다.

@Autowired

* 하나의 필드에 하나씩 정의해주어야 한다.

@Transactional

* Transaction을 Connection객체에서 처리한다.
* read only = true -> 커밋하지 않는다. (Database에 write하지 않는다.)
* Dao가 아닌 Service에서 처리해준다.
  1. 하나의 Service에서 여러 database transaction을 실행 할 수 있기 때문에 오류가 나는 경우 모두 롤백을 해야 하기 때문에 Service에서 처리해주어야 한다.

Database Table과 매핑되는 DTO는 domain이라는 용어로 사용한다.

dto나 vo는 값을 담아서 다니는 그릇으로 비유 될 수 있다. 용도에 알맞게 선언해서 사용 가능하다.

* MemberFormParam.java

@Configuration

* 이 파일은 설정파일이다. 라고 알려준다.

@ComponentScan

* 이 설정파일 내에 Bean을 등록해줘 라는 의미이다.

명일 수행

* REST API 구현 및 테스트
* 논리 모델링 & DDL작성
  1. Reservation System확인
* 카테고리 등록/수정/삭제
  1. github -> Fork
  2. 프로젝트 수행

Pull Request에 내용 작성 + Pull Request에 학습노트 URL 링크 작성

예약 메인 페이지, 관리자 페이지를 대체하도록 query insert문 작성해서 진행

kr.or.connect.reservation

Todo

* Sqls
  1. select\_by\_id
  2. select\_by\_comp
  3. update\_by\_id
  4. delete\_by\_id
  5. select\_count
  6. select\_all
  7. delete\_by\_comp
* todo Table
  1. id
  2. todo
  3. completed
  4. date