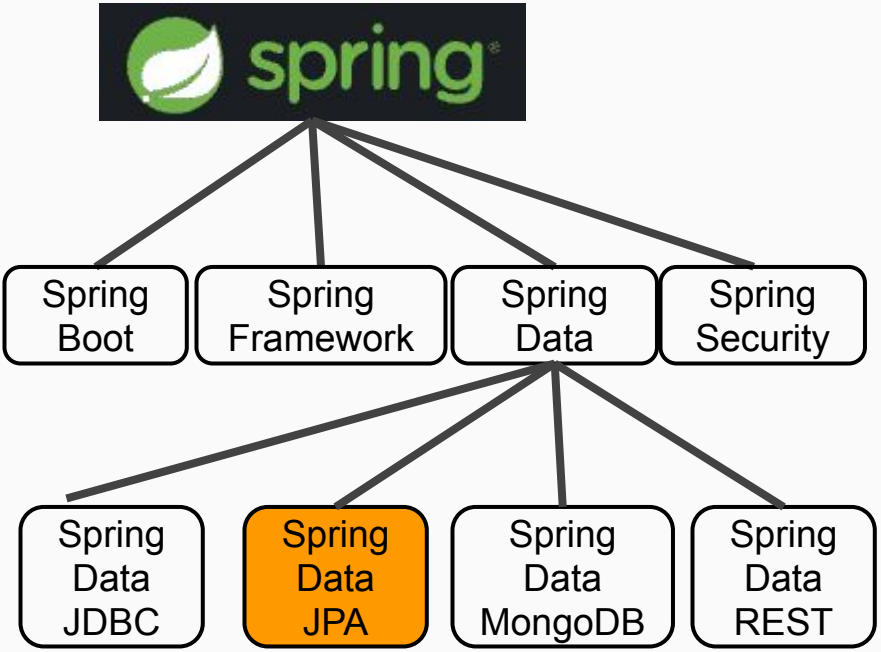
Chap12. 스프링 데이터 JPA

-꼬막조림

스프링 데이터 JPA



스프링 데이터 JPA

스프링 프레임워크에서 JPA를 편리하게 사용할 수 있도록 지원하는 프로젝트.

단순반복되는 CRUD 코딩작업을 없애줌.

데이터 접근 계층을 개발할 때 구현 클래스 없이 인터페이스만으로 작성.

프로젝트 설정

pom.xml

```
<dependencies>
   <dependency>
      <groupId>
         org.springframework.data
      </groupId>
      <artifactId>
         spring-data-jpa
      </artifactId>
      <version>1.8.2.RELEASE</version>
   </dependency>
</dependencies>
```

src/main/resources/appConfig.xml

```
<beans>
  <jpa:repositories base-package=
       "jpabook.jpashop.repository" />
</beans>
```

공통 인터페이스 기능

간단한 CRUD는 소스 작성없이 인터페이스 상속만으로 즉시 사용.

MemberRepository.java

```
public interface MemberRepository
  Extends JpaRepository<Member, Long> {
}
```

MemberService.java

```
public class MemberService {
  MemberRepository memberRepo;
  ...
  memberRepo.save(member);
  memberRepo.delete(member);
  memberRepo.findOne(id);
  memberRepo.findAll();
  ...
}
```

쿼리 메소드 기능

1. 메소드 이름으로 쿼리 생성.

MemberRepository.java

```
public interface MemberRepository
  Extends JpaRepository<Member, Long> {

  List<Member> findByEmailAndName(
    String email, String name);
}
```

MemberService.java

```
public class MemberService {
  …
  memberRepo.findByEmailAndName(
   “abc@gmail.com”, “홍길동”);
  …
}
```

```
select m from Member m
where m.email = “abc@gmail.com”
and m.name = ”홍길동”
```

쿼리 메소드 기능

2. JPA NamedQuery

MemberRepository.java

```
public interface MemberRepository
  Extends JpaRepository<Member, Long> {

  List<Member> findByUsername(
    @Param("username") String username);
}
```

Member.java

```
@Entity
@NamedQuery(
  name="Member.findByUsername",
  query="select m from Member m where
    m.username = :username")
public class Member {
  …
}
```

쿼리 메소드 기능

3. @Query

MemberRepository.java

```
public interface MemberRepository extends JpaRepository<Member, Long> {

  …

  @Query(“select m from Member m where M.username = ?1”)
  Member findByUsername(String username);

  ...
}
```

명세(SPECIFICATION)

참과 거짓으로 표현될수있는 조건을 서술한 것.
컴포지트 패턴으로 구성되어 있어 여러 명세를 조합하여 쓸수있음.

OrderService.java

```
Import static jpabook…..domain.OrderSpec.*;
public List<Order> findOrders(String name) {

  List<Order> result = orderRepository.findAll(
    where(memberName(name))
    .and(isOrderStatus())
    );
    return result;
}
```

domain/OrderSpec.java

```
public class OrderSpec {

  public static Specification<Order>
    memberName(final String memberName) {
      return true or false;
    }

  public static Specification<Order>
    isOrderStatus() {
      return true or false;
}
```

사용자 정의 리포지토리

스프링 데이터 JPA로 리포지토리를 개발하면 구현체를 만들필요가 없지만 필요에 의해서 구현체를 만들고 메소드를 직접 작성 할 수도있음.

MemberRepositoryCustom.java(인터페이스)

```
public interface MemberRepositoryCumstom {
  Public List<Member> findMemberCustom();
}
```

MemberRepositoryImpl.java(구현클래스)

```
public class MemberRepositoryImpl
  Implements MemberRepositoryCustom {
  @Override
  public List<Member> findMemberCustom() {...}
}
```

MemberRepository.java(인터페이스)

```
public interface MemberRepository
  extends JpaRepository<Member, Long>,
  MemberRepositoryCustom {
  ...
}
```

```
MemberRepository memberRepo;
member = memberRepo.findMemberCustom();
```

Web 확장(도메인 클래스 컨버터 기능)

URL: /member/memberUpdateFrom?id=1

MemberController.java

```
@RequestMapping(“member/memberUpdateForm”)
public String memberUpdateForm(@RequestParam(“id”) Long id, Model model) {
    Member member = memberRepository.findOne(id);
    model.addAttribute(“member”, member);
...}
```

```
@RequestMapping(“member/memberUpdateForm”)
Public String memberUpdateForm(@RequestParam(“id”) Member member, Model model) {
    model.addAttribute(“member”, member);
...}
```

Web 확장(페이징과 정렬)

URL: /members? page=0 & size=20 & sort=name,desc & sort=address.city

MemberController.java

```
@RequestMapping(“members”)
public String list(Pageable pageable, Model model) {

    ...
    Page<Member> page = memberservice.findMembers(pageable);
    model.addAttribute(“members”, page.getContent());
    return “members/memberList”;

}
```

감사합니다.