Repository구현

약간의 수정

```
public enum FeatureType {
    SUBSTRATE, FERTILIZER, PH_CORRECTOR
}
```

```
public enum Type {
    PLASTIC, WOOD
}
```

약간의 수정

```
INSERT INTO SEED_STARTER(covered,date_planted,type) VALUES(0,'2002-04-05 10:10:10','PLASTIC');
INSERT INTO SEED_STARTER(covered,date_planted,type) VALUES(0,'2001-04-05 10:10:10','WOOD');
INSERT INTO SEED_STARTER(covered,date_planted,type) VALUES(1,'2003-04-05 10:10:10', 'WOOD');

INSERT INTO FEATURE(name,SEED_STARTER_ID) VALUES('SUBSTRATE',1);
INSERT INTO FEATURE(name,SEED_STARTER_ID) VALUES('PH_CORRECTOR',1);
INSERT INTO FEATURE(name,SEED_STARTER_ID) VALUES('FERTILIZER',2);
INSERT INTO FEATURE(name,SEED_STARTER_ID) VALUES('PH_CORRECTOR',3);

INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(1,10,'Thymus vularis',1);
INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(2,15,'Thymus pseudolaginosus',1);
INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(3,20,'x citriodorus',1);
INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(1,5,'Thymus herba-barona',2);
INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(2,5,'Thymus serpyllum',2);
INSERT INTO DETAIL(ROW_NUM,SEED_PER_CELL, VARIETY, SEED_STARTER_ID) VALUES(1,20,'New Variety',3);
```

Repository

- **SeedStarterRepository**

```
public interface SeedStarterRepository extends JpaRepository<SeedStarter,Long> {

}
```

- **SeedStarterService**

```
@RequiredArgsConstructor
@Service
public class SeedStarterService {
    private final SeedStarterRepository seedStaterRepository;
    public List<SeedStarter> findAll(){
        return this.seedStaterRepository.findAll();
    }
}
```

Repository

- **SeedStarterMngController**

```
@RequiredArgsConstructor
@RestController
public class SeedStarterMngController {

    private final SeedStarterService seedStarterService;

    private final ObjectMapper mapper;
    @RequestMapping({"/","/seedstartermng"})
    public String showSeedstarters(final SeedStarter seedStarter, Model model)
    {
        List<SeedStarter> all = seedStarterService.findAll();
        all.stream().forEach(v-> System.out.println("v.getId() = " + v.getId()));
        return "hi";
        return "hello world";
    }
}
```

```
ddl-auto: update
```

Repository

- **SeedStarterMngController**
  - **JSON으로 변환하여 데이터 보내기**

```
@RequiredArgsConstructor
@RestController
public class SeedStarterMngController {

    private final SeedStarterService seedStarterService;

    private final ObjectMapper mapper;
    @RequestMapping({"/","/seedstartermng"})
    public String showSeedstarters(final SeedStarter seedStarter, Model model)throws JsonProcessingException
    {
        List<SeedStarter> all = seedStarterService.findAll();
        all.stream().forEach(v-> System.out.println("v.getId() = " + v.getId()));
        return mapper.writeValueAsString(all);
    }
}
```

Repository

- **@JsonManagedReference, @JsonBackReference**
  - 양방향 참조(Bidirectional Relationship)인 인스턴스를 JSON으로 변환하면 무한 재귀가 발생

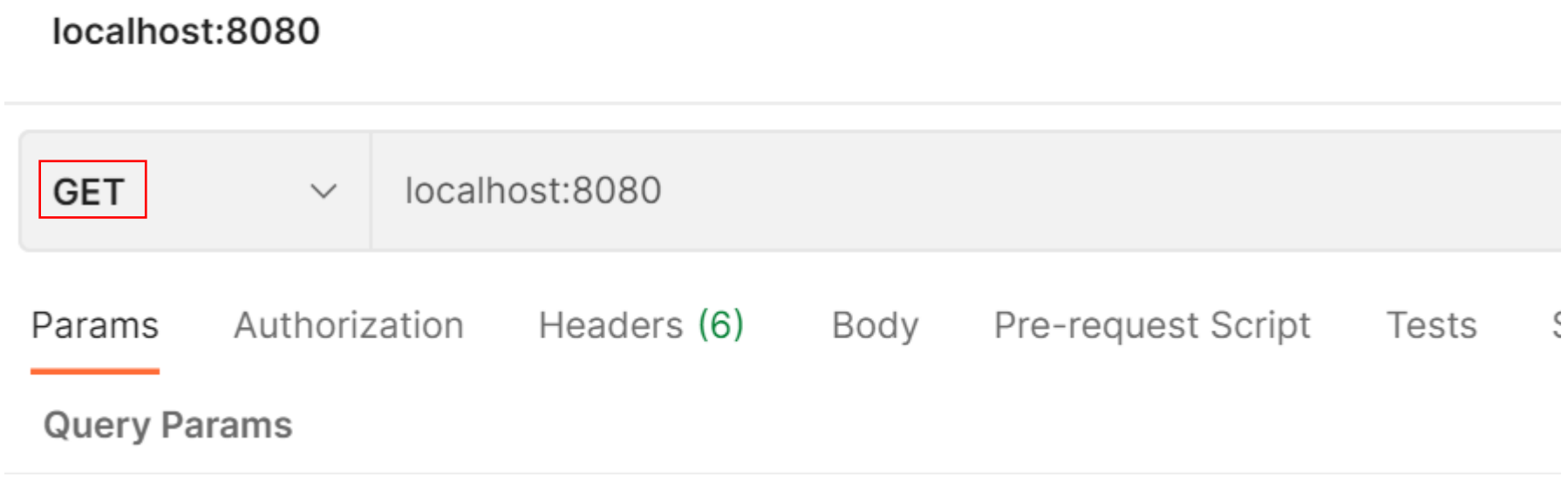
```
public class SeedStarter {

    @OneToMany(mappedBy = "seedStarter",cascade = CascadeType.PERSIST, orphanRemoval = true)
    @JsonManagedReference
    private List<Feature> features = new ArrayList<>();

    @OneToMany(mappedBy = "seedStarter",cascade = CascadeType.PERSIST, orphanRemoval = true)
    @JsonManagedReference
    private List<Detail> details = new ArrayList<>();
```

```
public class Feature {

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="SEED_STARTER_ID")
    @JsonBackReference
    private SeedStarter seedStarter;
```

```
public class Detail {

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "SEED_STARTER_ID")
    @JsonBackReference
    private SeedStarter seedStarter;
```

Repository

- **Postman으로 확인하기**
  - 응답으로 수신한 JSON을 크롬 확장 프로그램을 이용하여 가독성이 좋은 형태로 볼 수 있지만 Postman을 이용하여 수신한 JSON을 확인할 수 있음



  - JSON으로 변환하기 위해 연관관계 엔티티를 찾으면서 N+1문제가 발생
    - 쿼리를 보면 SeedStater를 찾을 때 조인이 발생하지 않음

Repository

- **EntityGraph**
  - 응답으로 수신한 JSON을 크롬 확장 프로그램을 이용하여 가독성이 좋은 형태로 볼 수 있지만

```
@NamedEntityGraph(name = "SeedStarter.all", attributeNodes = {
        @NamedAttributeNode("features")
})
```

```
@Repository
public interface SeedStarterRepository extends JpaRepository<SeedStarter,Long> {
    @EntityGraph(value = "SeedStarter.all", type = EntityGraphType.LOAD)
    @Query("SELECT DISTINCT s FROM SeedStarter s")
    List<SeedStarter> findWithFeatureAndDetail();
}
```

```
@NamedEntityGraph(name = "SeedStarter.all", attributeNodes = {
        @NamedAttributeNode("features"),
        @NamedAttributeNode("details")
})
```

MultipleBagFetchException

Repository

- **NamedEntityGraphs**
  - 쿼리 두 개로 분리해서 각각 조회

```
@Getter
@Setter
@NamedEntityGraphs(
	{
		@NamedEntityGraph(name = "SeedStarter.withFeature", attributeNodes = {
			@NamedAttributeNode("features")
		}),
		@NamedEntityGraph(name = "SeedStarter.withDetail", attributeNodes = {
			@NamedAttributeNode("details")
		})
	}
)

@Entity
public class SeedStarter {
```

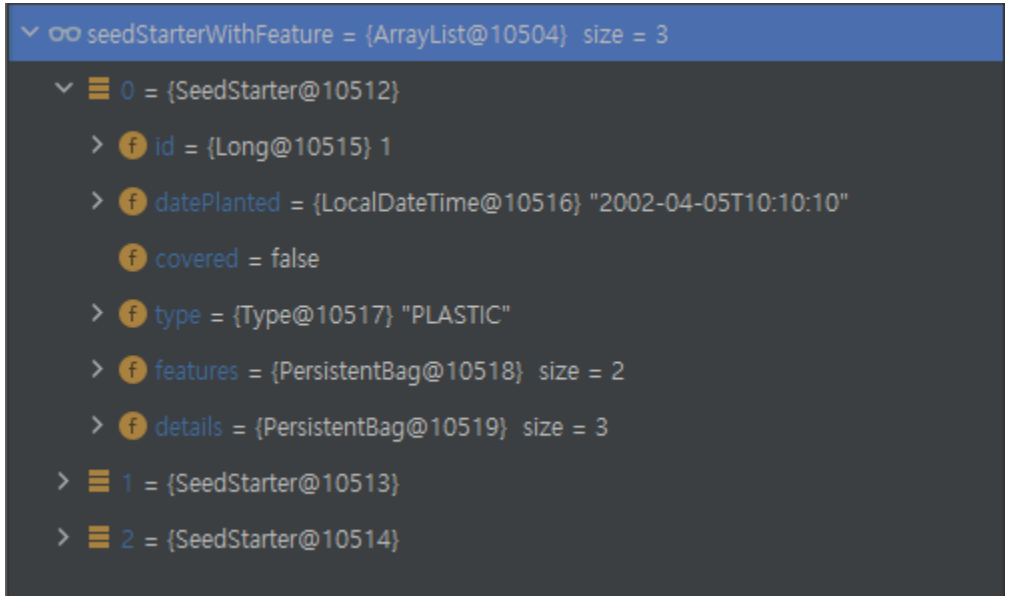
Repository

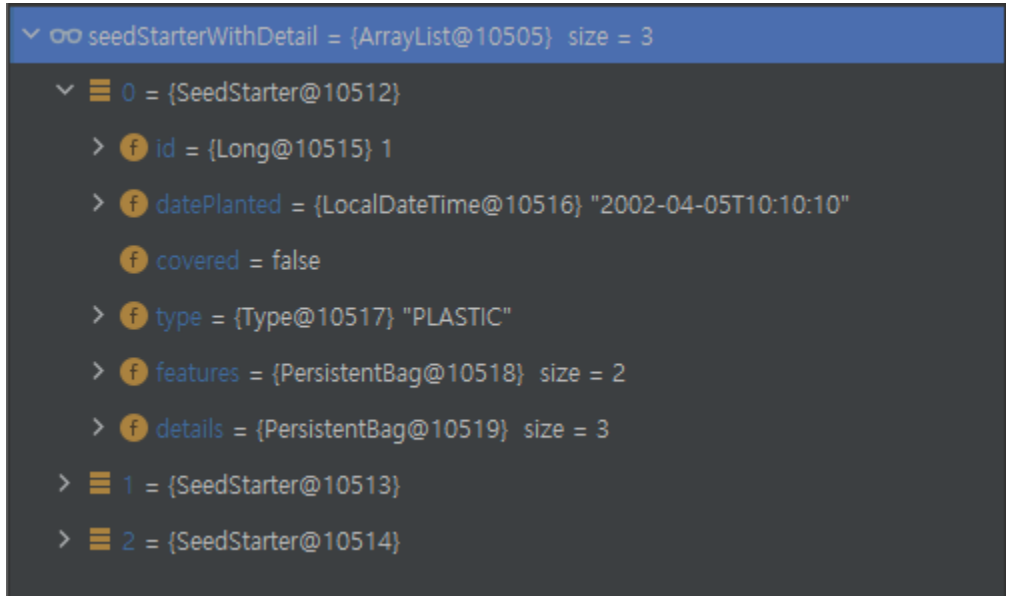
- **NamedEntityGraphs**
  - 쿼리 두 개로 분리해서 각각 조회

```
@Repository
@RequiredArgsConstructor
public class SeedStarterService {
    private final SeedStarterRepository seedStaterRepository;
    public List<SeedStarter> findWithFeature(){
        return this.seedStaterRepository.findWithFeature();
    }
    public List<SeedStarter> findWithDetail(){
        return this.seedStaterRepository.findWithDetail();
    }
}
```

Repository

- **NamedEntityGraphs**
  - 디버깅으로 값 확인해보기





Repository

- **EntityGraphType**
  - EntityGraph.EntityGraphType.FETCH
    - entity graph에 명시한 attribute는 EAGER로 패치
    - 나머지 attribute는 LAZY로 패치
  - EntityGraph.EntityGraphType.LOAD
    - entity graph에 명시한 attribute는 EAGER로 패치
    - 나머지 attribute는 entity에 명시한 fetch type이나 디폴트 FetchType으로 패치

모델에 데이터 담기

- **SeedStarterMngController**

```
public class SeedStarterMngController {

    private final SeedStarterService seedStarterService;

    private final ObjectMapper mapper;
    @RequestMapping({"/","/seedstartermng"})
    public String showSeedstarters(final SeedStarter seedStarter, Model model)throws JsonProcessingException
    {
        List<SeedStarter> seedStarterWithFeature = seedStarterService.findWithFeature();
        List<SeedStarter> seedStarterWithDetail = seedStarterService.findWithDetail();
        model.addAttribute("seedStarterWithFeature",seedStarterWithFeature);
        model.addAttribute("seedStarterWithDetail",seedStarterWithDetail);
        return "seedstartermng";
    }
}
```