Heameelega selgitan nende erinevate kihtide ja komponentide seoseid Spring rakenduses, eriti Service Layer Pattern kontekstis.

**Arhitektuurilised kihid ja nende seosed**

1. **Model (Mudel)** - Üksused, mis kajastavad andmebaasi tabeleid
   * Sisaldab andmeväljasid ja seoseid teiste mudelitega
   * Kasutab JPA annotatsioone (@Entity, @Table, jne)
   * Näide: Product, Category, User, Order
2. **DTO (Data Transfer Object)** - Andmete edastamiseks mõeldud objektid
   * Määratleb, millised andmed liiguvad teenuse ja kliendi vahel
   * Sageli sisaldab valideerimisannotatsioone
   * Erineb mudelitest, kuna:
     + Võib sisaldada vähem välju (ainult vajalikke)
     + Võib kombineerida mitme mudeli andmeid
     + Võib sisaldada täiendavaid välju
   * Näide: ProductDTO, ProductCreateDTO, ProductUpdateDTO
3. **Repository (Hoidla)** - Andmebaasiga suhtlemise kiht
   * Laiendab JpaRepository liidest
   * Defineerib andmebaasipäringud
   * Töötab otseselt Model objektidega
   * Näide: ProductRepository, CategoryRepository
4. **Mapper** - Teisendab Model ja DTO objekte omavahel
   * Tavaliselt implementeeritud MapStruct'i abil
   * Defineerib reeglid, kuidas andmed teisendatakse
   * Näide: ProductMapper, CategoryMapper
5. **Exception (Erandid)** - Kohandatud veaklassid
   * Võimaldab täpsemalt määratleda veatüüpe
   * Aitab luua ühtseid veaformaate
   * Näide: ResourceNotFoundException, InsufficientStockException
6. **Service (Teenus)** - Äriloogika kiht, mis määratleb toimingud
   * Defineerib liidesena kõik lubatud operatsioonid
   * Eraldab toimingute definitsiooni nende implementatsioonist
   * Näide: ProductService
7. **Service Impl (Teenuse implementatsioon)** - Äriloogika tegelik implementatsioon
   * Implementeerib teenuse liides
   * Sisaldab kogu toimingutega seotud loogikat
   * Kasutab hoidlaid andmebaasiga suhtlemiseks
   * Kasutab mappereid andmete teisendamiseks
   * Näide: ProductServiceImpl

**Kuidas nad koos töötavad**

Vaatame näidet getProductById meetodi voost:

1. **Controller (pole veel implementeeritud)** kutsub välja productService.getProductById(id)
2. **Service liides (ProductService)** defineerib meetodi signatuuri:

java

Copy

ProductDTO getProductById(Long id);

1. **Service implementatsioon (ProductServiceImpl)** teostab loogika:

java

Copy

@Override

@Transactional(readOnly = true)

public ProductDTO getProductById(Long id) {

log.info("Fetching product with id: {}", id);

// Kasutab repository't, et leida Product üksus

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Product not found with id: " + id));

// Kasutab mapper'it, et teisendada Product üksus ProductDTO-ks

return productMapper.toDTO(product);

}

1. **Repository (ProductRepository)** teostab andmebaasipäringu:

java

Copy

// JpaRepository sisaldab juba findById meetodit

Optional<Product> findById(Long id);

1. **Model (Product)** on andmebaasist leitud üksus, mis sisaldab kõiki välju
2. **Mapper (ProductMapper)** teisendab Product üksuse ProductDTO-ks:

java

Copy

@Mapping(source = "category.id", target = "categoryId")

@Mapping(source = "category.name", target = "categoryName")

ProductDTO toDTO(Product product);

1. **DTO (ProductDTO)** on teisendatud objekt, mis tagastatakse kontrollerile
2. **Exception (ResourceNotFoundException)** visatakse, kui toodet ei leita

**Meetodite seosed**

**1. Hoidlate meetodid (Repository)**

* findById - leiab ühe üksuse ID järgi
* findAll - leiab kõik üksused
* save - salvestab uue või uuendab olemasolevat üksust
* deleteById - kustutab üksuse ID järgi
* Kohandatud päringud (nt findByCategoryId, search)

**2. Mapperite meetodid (Mapper)**

* toDTO - teisendab mudeli DTO-ks
* toEntity - teisendab DTO mudeli üksuseks
* updateEntityFromDTO - uuendab olemasolevat üksust DTO andmetega

**3. Teenuste meetodid (Service)**

* CRUD operatsioonid:
  + getById - leiab ühe üksuse
  + getAll - leiab kõik üksused
  + create - loob uue üksuse
  + update - uuendab olemasolevat üksust
  + delete - kustutab üksuse
* Äriloogika operatsioonid:
  + reduceStock - vähendab toote laovarude seisu
  + isInStock - kontrollib, kas toode on saadaval

Kokkuvõttes moodustavad need kihid tugeva arhitektuurilise mustri, mis:

* Eraldab mured (separation of concerns)
* Teeb koodi paremini testitavaks
* Muudab muudatuste tegemise lihtsamaks
* Parandab koodi loetavust ja hooldatavust

Loodan, et see selgitus aitas sul mõista erinevate kihtide ja nende meetodite seoseid ServiceLayer Pattern'i kontekstis.