Domain Driven Design

Вопросы к обсуждению

- Общие сведения
- Rich model vs Anemic model
- Что нам хочется?
- Примеры кода
- Выводы
- Бонус

Общие слова о DDD

- DDD - это методология разработки, фокусирующаяся на логике предметной области

Rich Model vs Anemic Model

- Rich Model содержит бизнес-логику в объектах
- Anemic Model: объекты лишь хранят данные, без логики

  - Мне больше нравится русскоязычные термины
  - Полнокровная модель vs Анемичная модель

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных
  - Сохранить доменный объект в базе данных
  - Извлечь его из базы данных

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных
  - Сохранить доменный объект в базе данных
  - Извлечь его из базы данных
- Преобразовывать доменные объекты в DTO с помощью Mapstruct

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных
  - Сохранить доменный объект в базе данных
  - Извлечь его из базы данных
- Преобразовывать доменные объекты в DTO с помощью Mapstruct
- И обратно - DTO в доменные объекты

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных
  - Сохранить доменный объект в базе данных
  - Извлечь его из базы данных
- Преобразовывать доменные объекты в DTO с помощью Mapstruct
- И обратно - DTO в доменные объекты
- Возможно, потребуется еще ряд преобразований

Что нам хочется?

- Создавать полнокровные доменные модели для сохранения и извлечения данных
  - Сохранить доменный объект в базе данных
  - Извлечь его из базы данных
- Преобразовывать доменные объекты в DTO с помощью Mapstruct
- И обратно - DTO в доменные объекты
- Возможно, потребуется еще ряд преобразований
- И мы не хотим отказываться от преимуществ экосистемы Spring

Давайте сравним решения одной и той же задачи с помощью анемичной и полнокровной доменных моделей.

Давайте сравним решения одной и той же задачи с помощью анемичной и полнокровной доменных моделей.

- Java, Spring, https://github.com/pl99/car-management

Пример кода для анемичной модели

```java
@Entity
@Table(name = "cars")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@FieldDefaults(level = AccessLevel.PRIVATE)
@Builder(toBuilder = true)
public class Car {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(nullable = false, unique = true)
    String registrationNumber;

    @Column(nullable = false, unique = true)
    String vin;

    @Column(nullable = false)
    String make;

    @Column(nullable = false)
    String model;

    @Column(nullable = false)
    LocalDate productionDate;

}
```

Пример кода для анемичной модели

```java
@Entity
@Table(name = "cars")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@FieldDefaults(level = AccessLevel.PRIVATE)
@Builder(toBuilder = true)
public class Car {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(nullable = false, unique = true)
    String registrationNumber;

    @Column(nullable = false, unique = true)
    String vin;

    @Column(nullable = false)
    String make;

    @Column(nullable = false)
    String model;

    @Column(nullable = false)
    LocalDate productionDate;

}
```

```java
@Entity
@Table(name = "owners")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Builder(toBuilder = true)
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Owner {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(nullable = false)
    String name;

    @Column(nullable = false)
    String contactInfo;
}
```

Пример кода для анемичной модели

```
@Entity
@Table(name = "ownerships")
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder(toBuilder = true)
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(name = "car_id", nullable = false)
    Long carId;

    @Column(name = "owner_id", nullable = false)
    Long ownerId;

    @Column(nullable = false)
    LocalDate purchaseDate;

    @Column
    LocalDate saleDate;
}
```

Пример кода для анемичной модели

```
@RestController
@RequestMapping("/rest/ownerships")
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipController {

    OwnershipService service;

    @GetMapping
    public ResponseEntity<List<OwnershipDto>> getList() {
        return ResponseEntity.ok(service.findAll());
    }


    @PostMapping("sell")
    public ResponseEntity<OwnershipDto> sell(@RequestBody SellCarDto dto){
        return ResponseEntity.ok(service.sell(dto));
    }
    @PostMapping("purchase")
    public ResponseEntity<OwnershipDto> purchase(@RequestBody PurchaseCarDto dto){
        return ResponseEntity.ok(service.purchase(dto));
    }

}
```

Пример кода для анемичной модели

```
@Service
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {

    OwnershipRepository ownershipRepository;
    OwnershipMapper mapper;

    @Transactional
    public List<OwnershipDto> findAll() {
        return mapper.toDtos(ownershipRepository.findAll());
    }

    public OwnershipDto sell(SellCarDto dto) {
        Optional<Ownership> os = ownershipRepository.findByCarIdAndOwnerIdAndSaleDateNull(dto.getCarId(),dto.getOwnerId());
        if(os.isEmpty()){
            throw new IllegalArgumentException("can't sale this car");
        }
        Ownership ownership = os.get().toBuilder().saleDate(dto.getSaleDate()).build();
        Ownership saved = ownershipRepository.save(ownership);
        return mapper.toDto(saved);
    }

    public OwnershipDto purchase(PurchaseCarDto dto) {
        return null;
    }
}
```

Пример кода для анемичной модели

```java
@Service
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {

    OwnershipRepository ownershipRepository;
    OwnershipMapper mapper;

    @Transactional
    public List<OwnershipDto> findAll() {
        return mapper.toDtos(ownershipRepository.findAll());
    }

    public OwnershipDto sell(SellCarDto dto) {
        Optional<Ownership> os = ownershipRepository.findByCarIdAndOwnerIdAndSaleDateNull(dto.getCarId(),dto.getOwnerId());
        if(os.isEmpty()){
            throw new IllegalArgumentException("can't sale this car");
        }
        Ownership ownership = os.get().toBuilder().saleDate(dto.getSaleDate()).build();
        Ownership saved = ownershipRepository.save(ownership);
        return mapper.toDto(saved);
    }

    public OwnershipDto purchase(PurchaseCarDto dto) {
        return null;
    }
}
```

Пример кода для полнокровной модели

Пример кода для полнокровной модели

```
@Entity
@Table(name = "ownerships")
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder(toBuilder = true)
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(name = "car_id", nullable = false)
    Long carId;

    @Column(name = "owner_id", nullable = false)
    Long ownerId;

    @Column(nullable = false)
    LocalDate purchaseDate;

    @Column
    LocalDate saleDate;
}
```

Пример кода для полнокровной модели

```
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    Long carId;

    @Column(name = "owner_id", nullable = false)
    Long ownerId;

    @Column(nullable = false)
    LocalDate purchaseDate;

    @Column
    LocalDate saleDate;


    public Ownership save() {
        return null;
    }


    public OwnershipDto toDto() {
        return null;
    }

}
```

Пример кода для полнокровной модели

```
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    Long carId;

    @Column(name = "owner_id", nullable = false)
    Long ownerId;

    @Column(nullable = false)
    LocalDate purchaseDate;

    @Column
    LocalDate saleDate;


    public Ownership save() {        return null;    }


    public OwnershipDto toDto() {        return null;    }


    @Component("dddRepositoryOwnership")
    private static class Repository {
        private static OwnershipRepository repository;

        @Autowired
        void setRepository(OwnershipRepository component) {
            repository = component;
        }
    }
}
```

Пример кода для полнокровной модели

```
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    Long carId;

    @Column(name = "owner_id", nullable = false)
    Long ownerId;

    @Column(nullable = false)
    LocalDate purchaseDate;

    @Column
    LocalDate saleDate;


    public Ownership save() {
        return Repository.repository.saveAndFlush(this);
    }

    public OwnershipDto toDto() {return null;}


    @Component("dddRepositoryOwnership")
    private static class Repository {
        private static OwnershipRepository repository;
        @Autowired
        void setRepository(OwnershipRepository component) {
            repository = component;
        }
    }
}
```

Пример кода для полнокровной модели

```
public class Ownership {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    ...
    @Column
    LocalDate saleDate;


    public Ownership save() {
        return Repository.repository.saveAndFlush(this);
    }

    public OwnershipDto toDto() {
        return Mapper.mapper.toDto(this);
    }

    @Component("dddRepositoryOwnership")
    private static class Repository {
        private static OwnershipRepository repository;

        @Autowired
        void setRepository(OwnershipRepository component) {
            repository = component;
        }
    }
    @Component("dddMapperOwnership")
    private static class Mapper {
        private static OwnershipMapper mapper;

        @Autowired
        void setMapper(OwnershipMapper component) {
            mapper = component;
        }
    }
}
```

Пример кода для полнокровной модели

```
@RestController
@RequestMapping("/rest/ownerships")
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipController {

    OwnershipService service;

    @GetMapping
    public ResponseEntity<List<OwnershipDto>> getList() {
        return ResponseEntity.ok(service.findAll());
    }


    @PostMapping("sell")
    public ResponseEntity<OwnershipDto> sell(@RequestBody SellCarDto dto){
        return ResponseEntity.ok(service.sell(dto));
    }
    @PostMapping("purchase")
    public ResponseEntity<OwnershipDto> purchase(@RequestBody PurchaseCarDto dto){
        return ResponseEntity.ok(service.purchase(dto));
    }

}
```

Пример кода для анемичной модели

```
@Service
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {

    OwnershipRepository ownershipRepository;
    OwnershipMapper mapper;

    @Transactional
    public List<OwnershipDto> findAll() {
        return mapper.toDtos(ownershipRepository.findAll());
    }

    public OwnershipDto sell(SellCarDto dto) {
        Optional<Ownership> os = ownershipRepository.findByCarIdAndOwnerIdAndSaleDateNull(dto.getCarId(),dto.getOwnerId());
        if(os.isEmpty()){
            throw new IllegalArgumentException("can't sale this car");
        }
        Ownership ownership = os.get().toBuilder().saleDate(dto.getSaleDate()).build();
        Ownership saved = ownershipRepository.save(ownership);
        return mapper.toDto(saved);
    }

    public OwnershipDto purchase(PurchaseCarDto dto) {
        return null;
    }
}
```

Пример кода для полнокровной модели

```
@{...}
public class Ownership {
    //<editor-fold>
    public Ownership findForSale(SellCarDto dto) {  1 usage  new *
        return Repository.repository.findAllByOwnerId(dto.getOwnerId()) List<Ownership>
                .stream() Stream<Ownership>
                .filter(it -> it.getCarId().equals(dto.getCarId()))
                .findFirst() Optional<Ownership>
                .orElseThrow(() -> new IllegalArgumentException("car not found in this ownership!"));
    }

    public Ownership save() {  1 usage  new *
        return Repository.repository.saveAndFlush( entity: this);
    }

    //<editor-fold>
}
```

Пример кода для полнокровной модели

```java
@Service
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {

    OwnershipRepository ownershipRepository;
    OwnershipMapper mapper;

    @Transactional
    public List<OwnershipDto> findAll() {
        return mapper.toDtos(ownershipRepository.findAll());
    }

    @Transactional
    public OwnershipDto sell(SellCarDto dto){
        Ownership ownership = Ownership.builder().build().findForSale(dto);
        // Бизнеслогика
        Ownership sell = ownership.toBuilder().saleDate(dto.getSellDate()).build();

        Ownership saved = sell.save();
        return saved.toDto();
    }

    public OwnershipDto purchase(PurchaseCarDto dto) {
        return null;
    }
}
```

Пример кода для полнокровной модели

```
@{...}
public class SellCarDto {
    Long carId;
    Long ownerId;
    LocalDate saleDate;


    public Ownership findForSale() {  no usages  new *
        return SellCarDto.Repository.repository.findAllByOwnerId(this.getOwnerId())  List<Ownership>
                .stream()  Stream<Ownership>
                .filter(it -> it.getCarId().equals(this.getCarId()))
                .findFirst()  Optional<Ownership>
                .orElseThrow(() -> new IllegalArgumentException("car not found in this ownership!"));
    }
    //<editor-fold>
}
```

Пример кода для полнокровной модели

```
@{...}
public class Ownership {
    //<editor-fold>

    public OwnershipDto sale() {  2 usages  new *
        Ownership forSale = this.toBuilder()
                .saleDate(this.saleDate)
                .build();
        return Mapper.mapper.toDto(forSale.save());
    }

    //<editor-fold>

}
```

Пример кода для полнокровной модели

```java
@Service  new *
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {
    //<editor-fold>

    @Transactional  no usages  new *
    public OwnershipDto selfSell(SellCarDto dto){
        Ownership ownership = dto.findForSale();
        // Бизнеслогика
        return ownership.sale();
    }

    //<editor-fold>
}
```

Пример кода для полнокровной модели

```java
@Service  new *
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
public class OwnershipService {
    //<editor-fold>

    @Transactional  no usages  new *
    public OwnershipDto selfSell(SellCarDto dto){
        return dto.findForSale().sale();
    }

    //<editor-fold>
}
```

Выводы

- Это не DDD!!!
- Снижение когнитивной нагрузки. Читаем код, верхнеуровнево понимаем, что происходит.

- Это, на мой взгляд, полезные приемы для реализации конкретных действий.

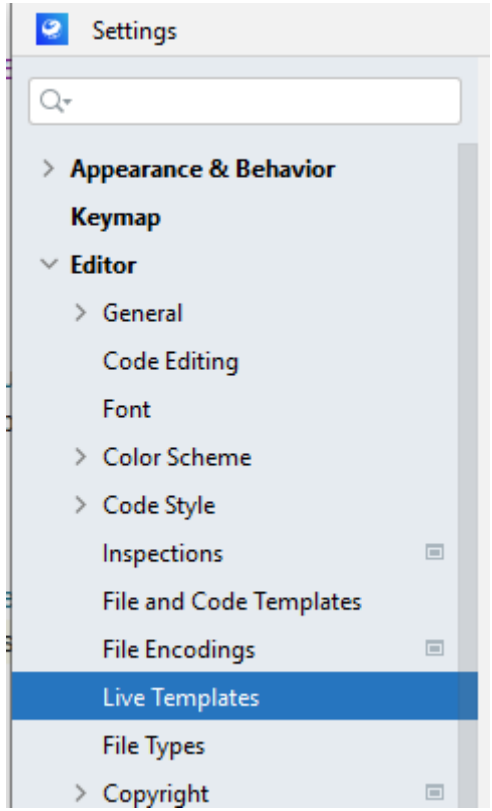
Бонус

Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов

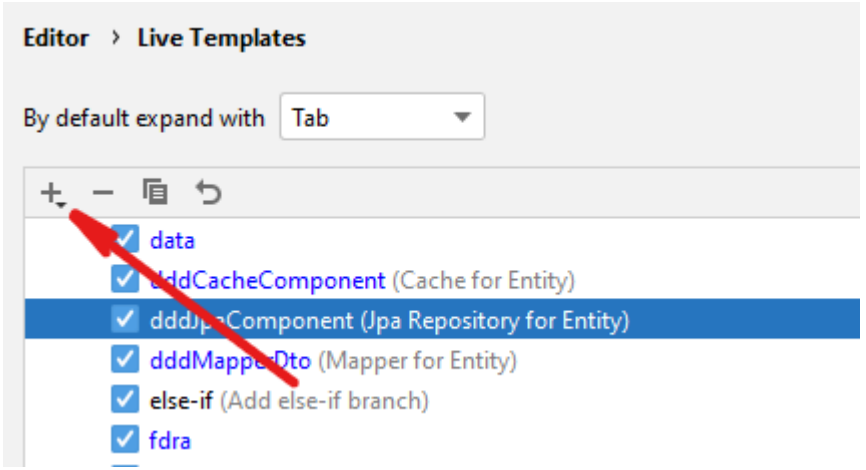
Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов



Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов

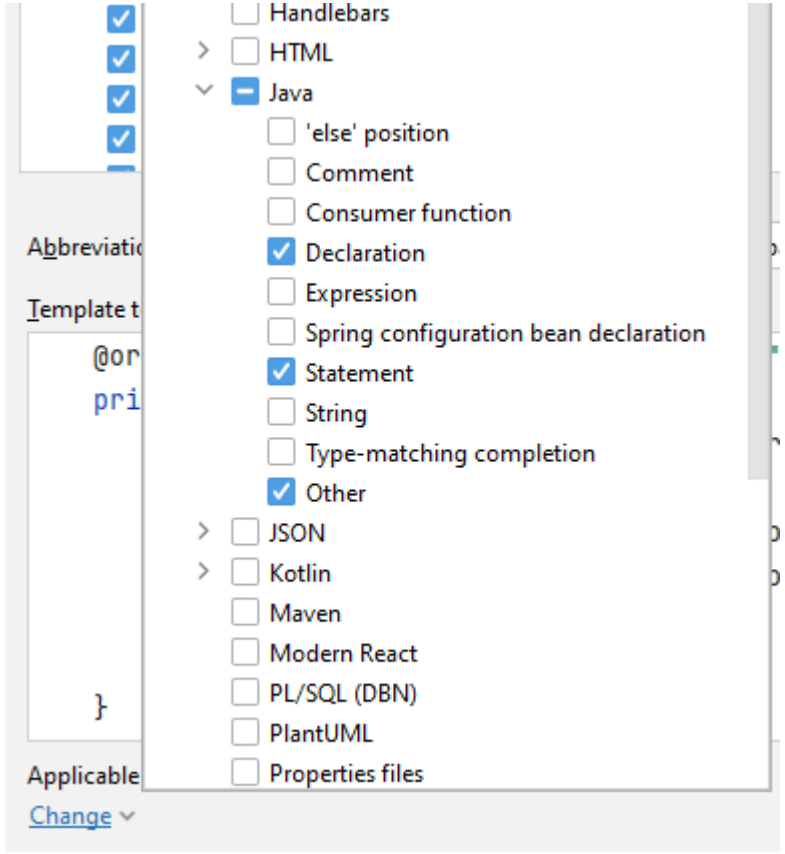


Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов

Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов



Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов
- Плагин Amplicode

Бонус: живые шаблоны и Amplicode

- Использование живых шаблонов
- Плагин Amplicode

https://amplicode.ru/documentation/installation-guide-intellij

Amplicode для IntelliJ IDEA

Amplicode для IntelliJ IDEA включает в себя поддержку экосистемы Spring и связанных технологий, а также предоставляет инструменты для работы с Docker и Docker Compose файлами.

Вопросы?