# Паттерны проектирования DTO, VO и как это связано с DDD.

# Общая цель паттернов проектирования состоит в том,

– чтобы предоставить готовые и проверенные решения типичных задач разработки программного обеспечения.

Они помогают разработчикам создавать код, который:

- Легко читается и поддерживается.
- Повторно используется.
- Уменьшает количество ~~ошибок~~.
- Улучшает модульность и гибкость.
- Ускоряет процесс разработки.

D

T

O

Data

T

O

# Data Transfer O

# Data Transfer Object

# для чего применяется?

применяется при:

- передачи данных между различными слоями системы

(например, между слоями представления и бизнес-логики, или между микросервисами)

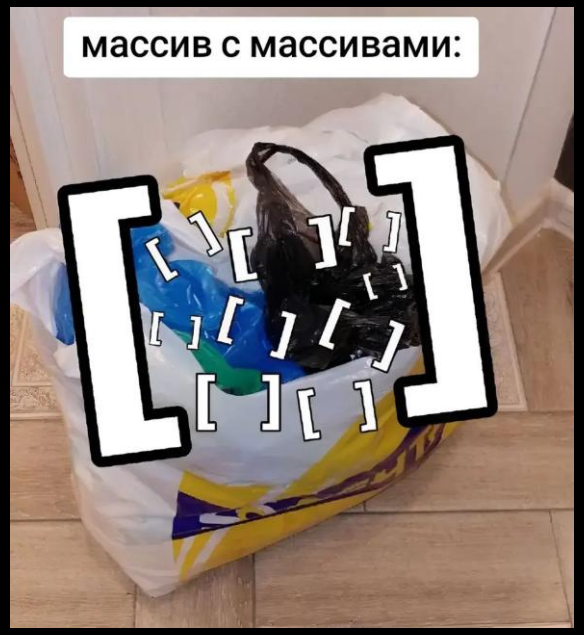
...при использовании DTO происходит:

# упрощение транспортировки данных

используется только для передачи данных в структурированном виде.

- не содержит бизнес-логики.

и это не ...



минимизирует количество данных, которые нужно передавать

обеспечивает инкапсуляцию

(сокрытие деталей реализации)

# аналогия из жизни:

органайзер для инструментов
собирает всё нужное
в одном месте,
чтобы вы легко переносили и
использовали инструменты.
DTO делает то же самое,
упаковывая данные в удобный
формат для передачи между
частями программы.

# Основная цель:

- передача информации без лишних зависимостей и логики

# Преимущества использования:

+ Чистая архитектура
+ Минимизация зависимости
+ Упрощение тестирования

Пример использования:

V

O

# Value

0

# Value Object

# объект, неизменяемая сущность

...имеет определенный набор данных

value object обладает следующими характеристиками:

неизменяемость
- после создания объект не может быть изменён

равенство по значению
- два объекта могут быть равными, если все их значения одинаковы

отсутствие идентификатора
- идентичность определяется только значением его атрибутов.

# ...а для чего?

представление бизнес-значений

- которые не имеют независимой идентичности и могут быть легко заменены, если значения изменяются.

Например, объекты, которые представляют собой валюту, адрес, дату или координаты.

# ...а для чего?

целостность данных

- данные остаются неизменяемыми и корректными на протяжении всего времени жизни объекта

# ...а для чего?

упрощение логики

- объекты VO могут быть легко проверены, сериализованы, переданы между компонентами системы без риска изменения их состояния

# ...а для чего?

улучшение бизнес-логики

VO может включать логику, которая проверяет, корректность значений, валидирует почтовый индекс, проверяет правильность валюты и т.д.

# аналогия из жизни:

- адрес не имеет уникального идентификатора,
его значимость зависит от данных,
которые он содержит,
таких как улица, номер дома, город и почтовый индекс.

- адрес — это неизменяемый объект,
и если нужно изменить его значения
(например, номер дома),
создается новый объект.

Пример использования:

# рассмотрим преимущества:

целостность данных

- можно быть уверенным в том, что данные, передаваемые через VO, остаются корректными

# рассмотрим преимущества:

инкапсуляция и изоляция

- бизнес-логика (например, валидность email, суммы и валюты) инкапсулирована внутри VO,
что упрощает код контроллеров и сервисов

# рассмотрим преимущества:

повторное использование

- объекты VO могут быть повторно использованы в разных частях

Domain

D

D

Domain
Driven
D

# Domain Driven Design

# ...внимание

DDD является подходом, а не конкретной реализацией
и ориентируется на:

## - сложную бизнес-логику

с акцентом на модель предметной области (domain)

Домен – один из основных принципов данного подхода

# основные концепты

унифицированный язык

-все участники разработки

используют общий язык, который отражает термины и концепты из предметной области

# основные концепты

Bounded Context (огр. контекст)

- система разбивается на ограниченные контексты,
  которые представляют собой независимые части системы,
  каждая из которых имеет свою модель предметной области

# основные концепты

Entities (Сущности)

- объекты, которые имеют уникальную идентичность, которая сохраняется в течение их жизненного цикла,

В сущностях важен их идентификатор (ID), и они могут изменять свое состояние

# основные концепты

## Value Objects (Объекты-значения)

- объекты, которые не имеют собственной идентичности и определяются только своими атрибутами

# основные концепты

Агрегаты

- группы связанных объектов (сущностей и VO), которые должны обрабатываться как единое целое. Каждый агрегат имеет корень (aggregate root)

# основные концепты

**Репозитории**

- отвечают за **хранение** и **извлечение** **агрегатов** и других объектов из базы данных или другого хранилища.
- предоставляют **интерфейс** для доступа к данным, **скрывая** детали их хранения.

# основные концепты

Сервисы

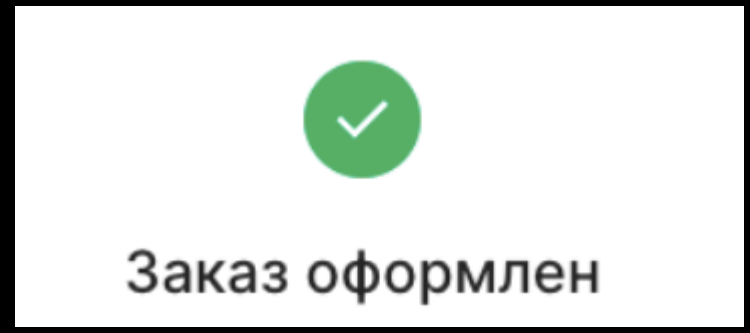
- реализуют бизнес-логику, которая не вписывается в модель сущности или VO
- используются для выполнения действий или операций, которые затрагивают несколько сущностей

# основные концепты

Фабрики

- объекты, которые создают агрегаты или другие сложные объекты
- обеспечивают создание объектов с соблюдением всех инвариантов системы

# основные концепты




**События домена**

- служат для оповещения системы или других частей системы о том, что произошло важное событие.

# дополнительные причины использования DTO в проекте

# Уменьшение количества параметров в методах (упрощение сигнатуры методов)

- сокращает сигнатуру методов
- облегчает рефакторинг

# Выделение общих параметров методов в DTO

- уменьшает дублирование
- позволяет унифицировать интерфейсы

# типобезопасность

- минимизировать ошибки при передаче данных между слоями
- использовать типизированные свойства и конструкторы

# снижение связности

- разделения ответственности между компонентами системы
- минимизации зависимости от изменений в базе данных или бизнес-логике

# централизация валидации данных

- использовать DTO для проверки входных данных на уровне приложения
- сочетать с типами и Yii2 правилами для удобной валидации

# удобство тестирования

- использовать DTO как простой объект без логики для упрощения тестирования
- легко создавать экземпляры с разными данными для юнит-тестов без обращения к БД или моделям

# упрощение передачи данных между слоями

- использовать DTO для передачи готовых объектов между контроллерами, сервисами и представлениями
- улучшает читаемость кода и упрощает поддержку по сравнению с массивами


# Спасибо за внимание!