Pro Java №40

Объект **DTO (Data Transfer Object)** – это специальная структура, используемая для передачи данных между различными слоями приложения. Основной целью DTO является минимизация избыточности данных и упрощение передачи информации, особенно когда речь идет о сложных и многослойных системах.

Основные характеристики DTO:

1. **Содержит только данные**:
    - DTO – это простой объект, обычно состоящий из полей, геттеров и сеттеров. Он не должен содержать бизнес-логику или сложные методы.
2. **Упрощает передачу данных**:
    - Используется для того, чтобы передавать только необходимые данные между слоями, избегая перегрузки или утечки лишней информации.
3. **Сокращает зависимость**:
    - DTO помогает избежать прямой зависимости между слоями, предоставляя независимый формат данных для передачи.
4. **Использование**:
    - Взаимодействие между клиентом и сервером (например, API).
    - Обмен данными между слоями приложения (контроллеры, сервисы, репозитории).

Преимущества использования DTO:

- Упрощение тестирования: DTO фокусируется только на данных, что делает его более предсказуемым и удобным для тестирования.
- Защита данных: DTO позволяет передавать только безопасную и нужную информацию, исключая чувствительные данные.
- Легкость интеграции: DTO можно легко преобразовать в другие форматы, такие как JSON, XML.

MapStruct

MapStruct – это библиотека для Java, которая позволяет автоматически генерировать код для преобразования объектов из одного типа в другой. Она используется для маппинга данных между DTO, сущностями и другими классами, обеспечивая высокую производительность и простоту в использовании.

Особенности MapStruct:

1. **Компиляция времени**:
    - Генерация маппингов происходит на этапе компиляции, что снижает нагрузку на производительность во время выполнения программы.
2. **Автоматический маппинг**:
    - Простые преобразования между классами выполняются автоматически на основе совпадающих имен полей.
3. **Кастомизация**:
    - Позволяет настроить сложные маппинги с помощью дополнительных методов и аннотаций.
4. **Аннотации**:
    - `@Mapper`: Определяет интерфейс или класс как маппер.
    - `@Mapping`: Позволяет настроить конкретный маппинг полей.

ModelMapper

ModelMapper – это библиотека для Java, которая предоставляет гибкий и мощный способ преобразования объектов. Она позволяет сопоставлять структуры данных и поддерживает как автоматический, так и ручной подход к маппингу.

Особенности ModelMapper:

1. **Удобство настройки**:
    - Простая настройка и использование, автоматическое сопоставление полей на основе их имени и типа.
2. **Глубокие маппинги**:
    - Поддерживает преобразование сложных вложенных структур данных.
3. **Гибкость**:
    - Можно настраивать правила сопоставления с помощью Condition и Custom Converter.

Аннотация `@JsonInclude` из библиотеки Jackson используется для настройки сериализации объектов в JSON. Она позволяет указать, какие поля класса должны включаться в JSON-результат, и помогает исключить ненужные данные.

Конвертер в Spring – это компонент, который используется для преобразования объектов одного типа в другой. Обычно он применяется

для преобразования данных при взаимодействии между слоями приложения или при обработке запросов и ответов.

Папка `target` создаётся при сборке Maven-проекта и обычно содержит **сгенерированный байт-код**, который компилируется из ваших Java-файлов.

Аннотация `@EnableMethodSecurity` (начиная с Spring Security 6) используется для включения безопасности на уровне методов в приложении. Это обновление замещает ранее использовавшуюся аннотацию `@EnableGlobalMethodSecurity`, упрощая и модернизируя конфигурацию безопасности.

Аннотация `@PreAuthorize` из Spring Security используется для проверки доступа **перед выполнением метода**. Она позволяет задавать сложные правила авторизации с использованием языка выражений Spring Expression Language (SpEL).

Слой обработки исключений (exception handling layer) в приложениях используется для централизованного управления ошибками и повышения надежности работы системы. В контексте Spring это часто достигается через аннотации и специфические инструменты.

ResponseEntity

- `ResponseEntity` – это класс в Spring Framework, который позволяет явно указать статус, заголовки и тело HTTP-ответа.
- Используется в контроллерах для формирования гибких ответов на запросы.

Аннотация `@ControllerAdvice` в Spring используется для создания глобального обработчика исключений и других аспектов, которые касаются всех контроллеров в вашем приложении. Она помогает централизовать логику обработки ошибок, настройки и общие функции для всех контроллеров.

Аннотация `@ExceptionHandler` в Spring используется для перехвата и обработки определённых исключений, возникающих в методах контроллера. Она позволяет создавать локальные обработчики ошибок или быть частью глобального обработчика в классе, помеченном аннотацией `@ControllerAdvice`.

Аннотация `@ResponseStatus` в Spring используется для установки HTTP-статуса ответа на уровне метода или класса контроллера. Она

позволяет назначать код статуса и описание, которые будут возвращены клиенту, без необходимости вручную конфигурировать объект `ResponseEntity`.