40. Ассоциация сущностей в Hibernate

В Hibernate существует несколько типов ассоциаций между сущностями: один к одному, один ко многим и многие ко многим. Эти типы ассоциаций позволяют моделировать связи между сущностями и устанавливать правильные отношения между ними.

Для определения связей между сущностями Hibernate использует аннотации @OneToOne, @OneToMany, @ManyToOne, @ManyToMany.

1. Ассоциация "Один-к-Один" (One-to-One) (например у одного человека может быть только один паспорт, и паспорт может пренадлежать только одному человеку)

@Entity

@Table (name="users")

public class Person

{

private String name;

@OneToOne (optional=false, cascade=CascadeType.ALL)

@JoinColumn (name="passport\_id")

private Passport passport;

}

@Entity

@Table (name="passports")

public class Passport

{

private String series;

private String number;

@OneToOne (optional=false, mappedBy="passport")

private Person owner;

}

optional – обязательное значение или нет, mappedBy – указывается сторона, кот. не несет ответственности за отношения (обратная стороны ассоциации, поле другого класса, которое имеет связь). Cascade – указывается на «хозяина» в отношениях (т.е. при удалении определенного юзера, сначала произойдет проверка, есть ли у него паспорт, затем удалится запись о паспорте, и только затем сам юзер)

1. Ассоциация "Многие-к-Одному" (Many-to-One, One -to- Many) (Гражданин имеет один основной адрес проживания, но по одному адресу могут проживать несколько человек.)

@Entity

@Table (name="users")

public class Person

{

private String name;

@ManyToOne (optional=false, cascade=CascadeType.ALL)

@JoinColumn (name="person\_id") // определяется поле связи в таблице БД private Address address;

}

@Entity

public class Address

{

private String city;

private String street;

private String building;

@OneToMany (mappedBy="address", fetch=FetchType.EAGER)

private Collection<Person> tenants;

}

адрес у гражданина только один, то используется аннотация @ManyToOne.

mappedBy также указывает на поле в классе владельца

fetch=FetchType.EAGER говорит о том, что при загрузке владеемого объекта необходимо сразу загрузить и коллекцию владельцев

1. Ассоциация "Многие-ко-Многим" (Many-to-Many) (каждый студент может быть записан на несколько курсов, и каждый курс может иметь множество студентов.)

@Entity

public class Student {

@Id

@GeneratedValue

private Long id;

private String name;

@ManyToMany

@JoinTable(name = "student\_course",

joinColumns = @JoinColumn(name = "student\_id"), // столбец в таблице "student" для внешнего ключа

inverseJoinColumns = @JoinColumn(name = "course\_id") // столбец в таблице "course" для внешнего ключа

)

private Set<Course> courses = new HashSet<>();

// ...

}

@Entity

public class Course {

@Id

@GeneratedValue

private Long id;

private String name;

@ManyToMany(mappedBy = "courses")

private Set<Student> students = new HashSet<>();

// ...

}

В классе Student мы используем аннотацию @ManyToMany для задания ассоциации с классом Course. С помощью аннотации @JoinTable мы указываем имя промежуточной таблицы "student\_course", которая содержит связь между "student" и "course". joinColumns и inverseJoinColumns указывают столбцы в промежуточной таблице для внешних ключей, которые связывают "student" и "course".

В классе Course мы используем аннотацию @ManyToMany(mappedBy = "courses"), чтобы указать, что связь с классом Student управляется полем courses.

41. Spring Boot: определение, характеристики, преимущества.

Spring Boot - фреймворк для разработки Java-приложений, который упрощает конфигурацию и развертывание приложений.

Характеристики:

* Простота использования и конфигурации.
* Автоматическая конфигурация и настройка.
* Встроенный сервер приложений.
* Управление зависимостями с помощью стартеров.
* Поддержка монолитных и микросервисных архитектур.

Преимущества:

* Ускоренная разработка благодаря автоматической конфигурации.
* Упрощенная конфигурация с использованием файлов свойств или YAML.
* Удобство тестирования с помощью инструментов для модульного тестирования.
* Гибкость и расширяемость за счет использования фреймворка Spring.
* Обширная экосистема Spring, предлагающая различные инструменты и библиотеки для разработки приложений.

В целом, Spring Boot предоставляет удобные и эффективные средства для разработки Java-приложений, упрощая конфигурацию и развертывание, а также ускоряя процесс разработки.

42. Spring Initializr, особенности и преимущества применения.

Spring Initializr - это инструмент для генерации инициализационных проектов на основе Spring Boot. Его особенности и преимущества:

Особенности:

* Простота использования: Интуитивно понятный интерфейс и возможность выбора необходимых компонентов и зависимостей.
* Гибкость конфигурации: Возможность указать версии фреймворка, язык программирования и другие параметры.
* Интеграция с средами разработки: Поддержка популярных IDE, таких как IntelliJ IDEA и Eclipse.
* Генерация исходного кода: Создание базовой структуры проекта, конфигурационных файлов и файлов сборки.

Преимущества:

* Ускорение разработки: Быстрое создание инициализационного проекта, что позволяет сосредоточиться на разработке функциональности.
* Конфигурация по умолчанию: Разумные дефолтные настройки для зависимостей и конфигурации, упрощающие начало работы.
* Легкое обновление и добавление зависимостей: Возможность обновлять и добавлять зависимости проекта без труда.
* Совместимость с экосистемой Spring: Интеграция с другими модулями и библиотеками Spring.

Spring Initializr позволяет быстро создавать инициализационные проекты на базе Spring Boot, сокращая время настройки и упрощая процесс разработки.