Семинар №5
Spring Data для работы с базами данных

1. **Инструментарий:**

Урок
Презентация

---

2. **Цели семинара №5:**

- Получить понимание основных принципов работы Spring Data.
- Закрепить навыки работы с базами данных с использованием Spring Data

По итогам семинара №5 слушатель должен **знать**:

- Принципы работы Spring Data.
- Как настроить и использовать Spring Data для работы с базами данных.
- Основные аннотации и интерфейсы Spring Data

По итогам семинара №5 слушатель должен **уметь**:

- Создавать репозитории с использованием Spring Data.
- Оперировать основными CRUD-операциями с помощью Spring Data.
- Использовать возможности запросов и сортировки данных с помощью Spring Data

---

3. План Содержание:

Этап урока	Тайминг, минуты	Формат
Введение, обзор темы	20	Модерирует преподаватель
Задание 1	40	Студенты выполняют, преподаватель помогает в решении проблем
Задание 2	40	Студенты выполняют, преподаватель помогает в решении проблем
Вопросы и обсуждение	20	Модерирует преподаватель
Длительность:	**120**	

Блок 1.

Тайминг:

Объяснение правил – 10 минут
Работа – 30 минут

Задание:

Создайте Spring Boot приложение и интегрируйте в него Spring Data JPA. Ваша задача - создать простой CRUD для сущности "Книга" (Book).

Сущность "Книга" должна содержать следующие поля:

- ID (тип Long и автоинкрементное)
- Название (тип String, не может быть пустым)
- Автор (тип String, не может быть пустым)
- Год публикации (тип Integer)

Пример решения:

Сущность Book:

```
@Entity
public class Book {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(nullable = false)
   private String title;

   @Column(nullable = false)
   private String author;

   @Column(nullable = true)
   private Integer publicationYear;

   // геттеры, сеттеры, конструкторы, equals, hashCode
}
```

Репозиторий для Book:

```
public interface BookRepository extends JpaRepository<Book, Long> {}
```

Сервисный слой:

java

Copy code

```
@Service
public class BookService {

   @Autowired
   private BookRepository bookRepository;

   public Book save(Book book) {
      return bookRepository.save(book);
   }

   // другие CRUD операции
}
```

Часто встречающиеся ошибки:

- Не указать аннотацию @Entity для сущности - без нее Spring Data JPA не сможет распознать класс как сущность базы данных.
- Забыть про аннотацию @Id для поля ID.
- Не инициализировать репозитории или сервисы через Spring DI, например, не использовать @Autowired или не создать соответствующий Spring Bean.
- Неправильно настроить связи между сущностями или забыть про их ограничения (например, nullable = false).
- Не обработать исключения, которые могут возникнуть при работе с базой данных (например, при попытке добавить книгу с уже существующим ID).

4. Блок 2.

Тайминг:
Объяснение правил – 10 минут
Работа в команде – 30 минут

Задание:

Используя Spring Boot и Spring Data JPA, реализуйте многотабличную БД с двумя сущностями: "Студент" (Student) и "Курс" (Course). Необходимо реализовать отношение многие ко многим, так чтобы один студент мог посещать несколько курсов, а курсы могли иметь множество студентов.

Сущность "Студент" должна содержать следующие поля:

- ID (тип Long и автоинкрементное)
- Имя (тип String, не может быть пустым)

Сущность "Курс" должна содержать:

- ID (тип Long и автоинкрементное)
- Название (тип String, не может быть пустым)

Пример решения:

Сущность Student:

```
@Entity
public class Student {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(nullable = false)
  private String name;
```

```
    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name = "student_course",
            joinColumns = @JoinColumn(name = "student_id"),
            inverseJoinColumns = @JoinColumn(name = "course_id"))
    private Set<Course> courses = new HashSet<>();

    // геттеры, сеттеры, конструкторы, equals, hashCode
}
```

Сущность Course:

```
@Entity
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String title;

    @ManyToMany(mappedBy = "courses")
    private Set<Student> students = new HashSet<>();

    // геттеры, сеттеры, конструкторы, equals, hashCode
}
```

Репозитории:

```
public interface StudentRepository extends JpaRepository<Student, Long> {}
public interface CourseRepository extends JpaRepository<Course, Long> {}
```

Часто встречающиеся ошибки:

- Забывают про аннотации @ManyToMany и @JoinTable.
- Не инициализируют коллекции, например, new HashSet<>(), что может привести к NullPointerException.
- Неправильно конфигурируют отношения, ведущие к цикличным зависимостям или проблемам с "ленивой" загрузкой (LazyInitializationException).
- Не учитывают в equals и hashCode отношения многие ко многим, что может привести к бесконечным циклам.

Домашнее задание

Условие:
Вам предстоит создать приложение для управления списком задач с использованием Spring Boot и Spring Data JPA. Требуется реализовать следующие функции:

1. Добавление задачи.
2. Просмотр всех задач.
3. Просмотр задач по статусу (например, "завершена", "в процессе", "не начата").
4. Изменение статуса задачи.
5. Удаление задачи.

Структура задачи:

- ID (автоинкрементное)
- Описание (не может быть пустым)
- Статус (одно из значений: "не начата", "в процессе", "завершена")
- Дата создания (автоматически устанавливается при создании задачи)