# **Семинар №3 Использование Spring для разработки серверного приложения**

1. **Инструментарий:**

[Текст урока](https://docs.google.com/document/d/10GesWA2SEqCbpLnmtttXs2J6W4tyPm5R/edit?usp=sharing&ouid=117975875178229561167&rtpof=true&sd=true)

[Презентация](https://docs.google.com/presentation/d/1nhX50-6N78k94FnDNhSuX586vftwQEzi/edit?usp=sharing&ouid=117975875178229561167&rtpof=true&sd=true)

**Цели семинара №3:**

1. Углубить понимание принципов работы Spring и Spring Boot.
2. Закрепить навыки создания более сложных сервисов на основе Spring.

**По итогам семинара №3 слушатель должен знать:**

1. Как создать сложный сервис с использованием внедрения зависимостей в Spring.
2. Как работает и как использовать @Autowired для внедрения зависимостей.

**По итогам семинара №3 слушатель должен уметь:**

1. Создавать и конфигурировать сложные сервисы в Spring.
2. Использовать @Autowired для внедрения зависимостей.

**План Содержание:**

| Этап урока | Тайминг, минуты | Формат |
| --- | --- | --- |
| Создание сложного сервиса с использованием Spring | 50 | Практическая работа |
| Использование @Autowired для внедрения зависимостей | 50 | Практическая работа |
| Итоги и домашнее задание | 20 | Заключение преподавателя |

Длительность: 120 минут

**Блок 1.**Тайминг:   
Объяснение правил – 15 минут  
Работа в команде – 35 минут

**Задание:**   
Создать сервис "DataProcessingService". Этот сервис должен принимать на вход список объектов типа User (с полями "name", "age", "email"), и выполнять следующие операции: сортировка списка пользователей по возрасту, фильтрация списка по заданному критерию (например, возраст больше 18), рассчет среднего возраста пользователей.

**Пример решения:**

1. Начнем с создания класса User. Он будет иметь поля "name", "age", "email". Пример кода:

public class User {

private String name;

private int age;

private String email;

// Геттеры и сеттеры для каждого поля

//...

}

2. Теперь, создадим сервис "DataProcessingService". Он будет содержать методы для работы с данными, которые будут использоваться в задании:

@Service

public class DataProcessingService {

public List<User> sortUsersByAge(List<User> users) {

return users.stream()

.sorted(Comparator.comparing(User::getAge))

.collect(Collectors.toList());

}

public List<User> filterUsersByAge(List<User> users, int age) {

return users.stream()

.filter(user -> user.getAge() > age)

.collect(Collectors.toList());

}

public double calculateAverageAge(List<User> users) {

return users.stream()

.mapToInt(User::getAge)

.average()

.orElse(0);

}

}

В этом сервисе используются методы Java Stream API для обработки данных: сортировки, фильтрации и вычисления среднего значения. Каждый метод принимает список пользователей в качестве параметра и возвращает результат после обработки.

**Блок 2.**Тайминг:   
Объяснение правил – 15 минут  
Работа в команде – 35 минут

**Задание:**   
Создать два сервиса - "UserService" и "NotificationService". UserService должен содержать метод createUser(Stringname, int age, String email), который создает пользователя и возвращает его. NotificationService должен иметь метод notifyUser(User user), который просто печатает сообщение о том, что пользователь был создан. Ваша задача - использовать @Autowired в UserService для внедрения NotificationService и вызвать метод notifyUserпосле создания нового пользователя.

**Пример решения:**

1. Сначала, создадим класс "User" (если он ещё не создан). Он будет иметь поля "name", "age", "email".

public class User {

private String name;

private int age;

private String email;

// Геттеры и сеттеры для каждого поля

//...

}

2. Создадим сервис "UserService", который будет создавать новых пользователей.

@Service

public class UserService {

private NotificationService notificationService;

// Внедрение зависимости через конструктор

public UserService(NotificationService notificationService) {

this.notificationService = notificationService;

}

public User createUser(String name, int age, String email) {

User user = new User();

user.setName(name);

user.setAge(age);

user.setEmail(email);

// Отправляем уведомление о создании нового пользователя

notificationService.notifyUser(user);

return user;

}

}

3. Создадим сервис "NotificationService", который будет отправлять уведомления.

@Service

public class NotificationService {

public void notifyUser(User user) {

System.out.println("A new user has been created: " + user.getName());

}

}

В этом примере мы используем Spring для внедрения зависимости "NotificationService" в "UserService" через конструктор. После создания нового пользователя вызывается метод "notifyUser", который отправляет уведомление.

**Домашнее задание:**Создать сервис "RegistrationService", который принимает на вход данные о пользователе (имя, возраст, email), создает пользователя с помощью UserService, затем использует DataProcessingService для добавления пользователя в список и выполнения операций над этим списком. После выполнения каждой операции, использовать NotificationService для вывода информации о выполненной операции.

**Рекомендации для преподавателей по оценке задания:**

1. Проверьте правильность использования аннотации @Autowired.

2. Убедитесь, что все операции выполняются в нужном порядке и результаты каждой операции корректны.