Домашнее задание:

1. **Интерфейс и реализации:**
    - Создайте интерфейс **MessagingService** с методом **sendMessage**.
    - Реализуйте два класса, реализующих **MessagingService**: **EmailService** и **SMSService**.
    - В каждом из классов добавьте аннотацию **@Service**. Эта аннотация помечает класс-прослойку как сервисный класс.
2. **Конфигурационный класс:**
    - Создайте конфигурационный класс **AppConfig**, используя аннотацию **@Configuration**. Эта аннотация помечает класс как конфигурационный класс Spring`а, где указаны наши бины.
    - В конфигурационном классе определите бины для **EmailService** и **SMSService**.
    - Используйте аннотацию **@Primary** для одного из бинов. (Изучите эту аннотацию самостоятельно, её принцип работы схож с **@Qualifier,** но кое чем отличается).
3. **Класс NotificationService:**
    - Создайте класс **NotificationService**, который будет использовать **MessagingService**.
    - Внедрите оба бина **MessagingService** с использованием аннотации **@Autowired** и **@Qualifier** в методе-конструкторе.
4. **Тестирование:**
    - В методе **main** создайте контекст Spring, получите бин **NotificationService** и вызовите метод **sendNotification** с произвольным сообщением.
5. **Дополнительные задания:**
    - Создайте дополнительный бин, например, **PushNotificationService**, реализующий **MessagingService**.
    - Используйте этот дополнительный бин в вашем классе **NotificationService**.
    - Расширьте класс **Main**, чтобы включить использование нового бина.

Это домашнее задание поможет вам более подробно изучить внедрение бинов в Spring и работу с различными сценариями внедрения (например, когда есть несколько бинов одного типа или разных типов).

Рекомендация к прочтению: [Контекст приложения Spring | for-each.dev]

Также обязательно прочтите материал о Spring Framework, который я скинул на занятии!

Проект с занятия:

```
package org.example;

import java.text.MessageFormat;

public class LogicClass {

    private String name;
    private int code;

    public LogicClass() {
        System.out.println("LogicClass was initialize");
    }

    public LogicClass(String name, int code) {
        this.name = name;
        this.code = code;
    }

    public void simpleLogic() {
        System.out.println("Simple class logic");
    }

    public void printLogicData() {
        System.out.println(MessageFormat.format("Simple logic data: {0}, {1}", name,
code));
    }
}
```

```
package org.example;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;

@ComponentScan("org.example")
public class AppConfig {

    @Bean // С помощью этой аннотации помещаем бин данного метода в контейнер спринга
    public LogicClass simpleLogicClass() {
        return new LogicClass();
    }

    @Bean
    public LogicClass logicClassData() {
        return new LogicClass("Logic class", 52);
    }
}
```

```
package org.example;

import org.springframework.stereotype.Component;

@Component // С помощью этой аннотации помещаем бин данного класса в контейнер спринга
public class DataComponent {

    public DataComponent() {
        System.out.println("DataComponent init");
    }
```

```
    public void someWork() {
        System.out.println("Some component work");
    }
}
```

```
package org.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Worker {
    private LogicClass simpleLogic;
    private LogicClass dataSimpleLogic;
    private DataComponent dataComponent;

    @Autowired
    public Worker(LogicClass simpleLogicClass, LogicClass logicClassData, DataComponent
dataComponent) {
        this.simpleLogic = simpleLogicClass;
        this.dataSimpleLogic = logicClassData;
        this.dataComponent = dataComponent;
    }

    public void call() {
        simpleLogic.simpleLogic();
        dataSimpleLogic.printLogicData();
        dataComponent.someWork();
    }
}
```

```
package org.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        context.getBean(Worker.class).call();
//
//        LogicClass logicClass = context.getBean("logicClassData", LogicClass.class);
//
//        logicClass.simpleLogic();
//        logicClass.printLogicData();
//
////        DataComponent dataComponent = context.getBean(DataComponent.class);
////        dataComponent.someWork();
//        context.getBean(DataComponent.class).someWork();
    }
}
```