Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

Внедрение зависимостей в Spring
Тема

Преподаватель _____ А. С. Черниговский
подпись, дата — инициалы, фамилия

Студент КИ19-17/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2021

1 Цель

Цель настоящей работы состоит в ознакомлении с механизмом внедрения зависимостей в Spring.

2 Задачи

Необходимо создать два приложения в которых будут объявлены Spring-конфигурации. В одном – только при помощи xml, в другом – при помощи аннотаций и класса-конфигурации. В приложении, которое сконфигурировано с помощью аннотаций снабдить спроектированные классы init и destroy методами, а также использовать фабричный метод для любого из классов.

Есть сущность (класс), необходимо создать интерфейс и классы его имплементирующие. Объекты классов, имплементирующих данный интерфейс будут передаваться в качестве зависимостей. Выполнить связывание и получить объекты из контекста. Продемонстрировать результаты в простейшем консольном приложении.

3 Описание задания

Вариант 10. Сущность – холодильник.

4 Ход выполнения

4.1 Приложение с объявлением Spring-конфигурации через xml

Ниже представлены листинги программы и значимая в рамках работы часть xml-файла конфигурации по заданию.

Листинг 1 – Код интерфейса Compressor

```
package com.github.durakin.isdlabs.lab2.components;

public interface Compressor {
    String work();
}
```

Листинг 2 – код класса Fridge

```
package com.github.durakin.isdlabs.lab2.components;

public class Fridge {
    private final Compressor compressor;

    public Fridge(Compressor compressor) {
        this.compressor = compressor;
    }

    @Override
    public String toString() {
        return "Fridge, equipped with compressor: " + compressor;
    }

    public void statusOutput() { System.out.println(this.compressor.work());}

}
```

Листинг 3 – код класса InverterCompressor

```
package com.github.durakin.isdlabs.lab2.components;

public class InverterCompressor implements Compressor {
    private final int power;
    private int vibration;

    public InverterCompressor(int power) {
        this.power = power;
    }

    public InverterCompressor(int power, int vibration) {
        this.power = power;
        this.vibration = vibration;
    }

    public void setVibration(int vibration) {
        this.vibration = vibration;
    }

    @Override
    public String work() {
```

Окончание листинга 3

```
        return "Compressor works quite, but vibrates";
    }

    @Override
    public String toString() {
        return "Inverter compressor with power of " + power + " and vibration of
" + vibration;
    }
}
```

Листинг 4 – код класса LinearCompressor

```
package com.github.durakin.isdlabs.lab2.components;

public class LinearCompressor implements Compressor {
    private final int power;
    private int noiseLevel;

    public LinearCompressor(int power, int noiseLevel) {
        this.power = power;
        this.noiseLevel = noiseLevel;
    }

    public void setNoiseLevel(int noiseLevel) {
        this.noiseLevel = noiseLevel;
    }

    @Override
    public String work() {
        return "Compressor works loudly growling";
    }

    @Override
    public String toString() {
        return "Linear compressor with power of " + power + " and noise level of
" + noiseLevel;
    }
}
```

Листинг 5 – код класса Program

```
package com.github.durakin.isdlabs.lab2;

import com.github.durakin.isdlabs.lab2.components.Fridge;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Program {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
                ClassPathXmlApplicationContext("applicationContext.xml");
        Fridge linearFridge = context.getBean("linearFridgeBean", Fridge.class);
        Fridge inverterFridge = context.getBean("inverterFridgeBean",
Fridge.class);
        System.out.println(linearFridge);
        linearFridge.statusOutput();
        System.out.println(inverterFridge);
        inverterFridge.statusOutput();

        context.close();
    }
}
```

Листинг 6 – выдержка из кода xml-файла со Spring-конфигурцией

```
    <bean id="linearCompressorBean"
          class="com.github.durakin.isdlabs.lab2.components.LinearCompressor">
        <constructor-arg name="noiseLevel" value="300"/>
        <constructor-arg name="power" value="40"/>
    </bean>

    <bean id="linearFridgeBean"
          class="com.github.durakin.isdlabs.lab2.components.Fridge">
        <constructor-arg ref="linearCompressorBean"/>
    </bean>

    <context:property-placeholder
location="classpath:inverterCompressor.properties"/>
    <bean id="inverterCompressorBean"
class="com.github.durakin.isdlabs.lab2.components.InverterCompressor">
        <constructor-arg value="35"/>
        <property name="vibration" value="${compressorVibration}"/>
    </bean>

    <bean id="inverterFridgeBean"
```

Окончание листинга 6

```
          class="com.github.durakin.isdlabs.lab2.components.Fridge">
        <constructor-arg ref="inverterCompressorBean"/>
    </bean>


</beans>
```

4.2 Приложение с объявлением Spring-конфигурации через аннотации

Ниже представлены листинги программы.

Листинг 7 – Код интерфейса Compressor

```
package com.github.durakin.isdlabs.lab2.components;

public interface Compressor {
    String work();

    void startWorking();

    void stopWorking();
}
```

Листинг 8 – код класса Fridge

```
package com.github.durakin.isdlabs.lab2.components;

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

@Component
public class Fridge {

    private Compressor compressor;

    @Autowired
    public Fridge(@Qualifier("linearCompressor1") Compressor compressor) {
        this.compressor = compressor;
```

Окончание листинга 8

```
    }

    @Override
    public String toString() {
        return "Fridge, equipped with compressor: " + compressor;
    }

    public void statusOutput() {
        System.out.println(this.compressor.work());
    }

}
```

Листинг 9 – код класса InverterCompressor

```
package com.github.durakin.isdlabs.lab2.components;

public class InverterCompressor implements Compressor {
    private final int power;
    private final int vibration;

    public InverterCompressor(int power, int vibration) {
        this.power = power;
        this.vibration = vibration;
    }

    @Override
    public void startWorking() {
        System.out.println("Compressor starts working quietly");
    }

    @Override
    public void stopWorking() {
        System.out.println("Compressor stopped working without any sound");
    }

    @Override
    public String work() {
```

Окончание листинга 9

```
        return "Compressor works quite, but vibrates";
    }

    @Override
    public String toString() {
        return "Inverter compressor with power of " + power + " and vibration of
" + vibration;
    }
}
```

Листинг 10 – код класса LinearCompressor

```
package com.github.durakin.isdlabs.lab2.components;

public class LinearCompressor implements Compressor {
    private final int power;
    private int noiseLevel;

    public LinearCompressor(int power, int noiseLevel) {
        this.power = power;
        this.noiseLevel = noiseLevel;
    }

    public void setNoiseLevel(int noiseLevel) {
        this.noiseLevel = noiseLevel;
    }

    @Override
    public void startWorking() {
        System.out.println("Compressor starts working loudly");
    }

    @Override
    public void stopWorking() {
        System.out.println("Compressor stopped working with weird echo noises");
    }

    @Override
    public String work() {
        return "Compressor works loudly growling";
    }
```

Окончание листинга 10

```
    @Override
    public String toString() {
        return "Linear compressor with power of " + power + " and noise level of
" + noiseLevel;
    }
}
```

Листинг 11 – код класса CompressorFactory

```
package com.github.durakin.isdlabs.lab2.components;


public class CompressorFactory {
    public static Compressor getCompressor(String type, int power, int spec) {
        if ("Inverter".equalsIgnoreCase(type)) {
            return new InverterCompressor(power, spec);
        }
        if ("Linear".equalsIgnoreCase(type)) {
            return new LinearCompressor(power, spec);
        }
        return null;
    }
}
```

Листинг 12 – код класса Program

```
package com.github.durakin.isdlabs.lab2;


import com.github.durakin.isdlabs.lab2.components.Fridge;
import com.github.durakin.isdlabs.lab2.components.LinearCompressor;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;


public class Program {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context =
                new AnnotationConfigApplicationContext(Config.class);
        Fridge fridge1 = (Fridge) context.getBean("fridge");
        Fridge fridge2 = (Fridge) context.getBean("inverterFridge");
        LinearCompressor linearCompressor = (LinearCompressor)
context.getBean("linearCompressor2");
        System.out.println(linearCompressor);
        System.out.println(fridge1);
        fridge1.statusOutput();
```

Окончание листинга 12

```
        System.out.println(fridge2);
        fridge2.statusOutput();
        context.close();
    }
}
```

Листинг 13 – код класса Config

```
package com.github.durakin.isdlabs.lab2;

import com.github.durakin.isdlabs.lab2.components.CompressorFactory;
import com.github.durakin.isdlabs.lab2.components.Fridge;
import com.github.durakin.isdlabs.lab2.components.InverterCompressor;
import com.github.durakin.isdlabs.lab2.components.LinearCompressor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

@Configuration
@ComponentScan("com.github.durakin.isdlabs.lab2.components")
@PropertySource("classpath:linearCompressor.properties")
public class Config {

    @Bean (name="linearCompressor1", initMethod = "startWorking", destroyMethod
= "stopWorking")
    public LinearCompressor linearCompressor1(){
        return (LinearCompressor) CompressorFactory.getCompressor("LiNeAr",
1488, 228);
    }

    @Value("${compressorNoise}")
    private int noise;
    @Bean (initMethod = "startWorking", destroyMethod = "stopWorking")
    public LinearCompressor linearCompressor2(){
        LinearCompressor result = new LinearCompressor(1337, 0);
```

Окончание листинга 13

```
        result.setNoiseLevel(noise);
        return result;
    }

    @Bean (initMethod = "startWorking", destroyMethod = "stopWorking")
    public InverterCompressor inverterCompressor(){
        return (InverterCompressor) CompressorFactory.getCompressor("INVERTER",
40000, 100500);
    }

    @Bean ("inverterFridge")
    public Fridge inverterFridge(){
        return new Fridge(inverterCompressor());
    }
}
```

5 Выводы

Была написана программа, соответствующая поставленным задачам. В ходе работы были ознакомительно изучены механизмы внедрения зависимостей в Spring.