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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2

Spring, внедрение зависимостей
Тема

Преподаватель _____ А.С. Черниговский
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цель

Ознакомиться с механизмом внедрения зависимостей в Spring

2 Задачи

Необходимо создать два приложения (или одно) в котором 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 вами классы init и destroy методами, а также использовать фабричный метод для любого из классов.

В каждом варианте есть сущность (класс), необходимо создать интерфейс (самостоятельно на усмотрение студента) 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.

Необходимо:

1) Реализовать внедрение простых значений через конструктор;

2) Реализовать внедрение зависимости по ссылке через конструктор;

3) Интерфейс должен содержать как минимум один метод;

4) Классы, имплементирующие интерфейс, должны содержать как минимум одно поле (у разных классов разные);

5) Зависимый класс должен содержать метод, который бы на основе вызова метода у зависимости выводил бы некоторое сообщение в консоль;

6) Реализовать внедрение простых значений из внешнего файла через setter.

3 Описание варианта

Класс «Backpack».

4 Исходные тексты программ

4.1 XML-конфигурация

На листингах представлен код, реализующий задание через xml-конфигурацию.

Листинг 1 – XML-конфигурация

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd">

    <context:property-placeholder location="classpath:database.properties"/>

    <bean id="tent"
          class="ru.second.xmlConfig.Tent">
        <constructor-arg value="2"/>
        <property name="humanCapacity" value="${tent.humanCapacity}"/>
    </bean>

    <bean id="shovel"
          class="ru.second.xmlConfig.Shovel">
        <constructor-arg type="ru.second.xmlConfig.ShovelTypes" value="SPADE"/>
        <property                                                    name="type"
value="${ru.second.annotationConfig.shovelTypes}"/>
    </bean>

    <bean id="cannedFood"
          class="ru.second.xmlConfig.CannedFood">
        <constructor-arg value="Ананасы"/>
        <property name="product" value="${cannedFood.product}"/>
    </bean>

    <bean id="backpack"
          class="ru.second.xmlConfig.Backpack">
        <constructor-arg value="${backpack.brandName}"/>
        <property name="burden" ref="cannedFood"/>
```

Окончание листинга 1

```
    </bean>
</beans>
```

Листинг 2 – Интерфейс Portable

```
public interface Portable {
    void showInfo();
}
```

Листинг 3 – Класс Shovel

```
public @Data
@AllArgsConstructor
class Shovel implements Portable {
    private ShovelTypes type;

    @Override
    public void showInfo() {
        System.out.println("Название объекта: лопата");
        System.out.println("Тип лопаты: " + this.type.name().toLowerCase());
    }
}
```

Листинг 4 – Класс Tent

```
public @Data
@AllArgsConstructor
class Tent implements Portable {
    private int humanCapacity;

    @Override
    public void showInfo() {
        System.out.println("Название объекта: палатка");
        System.out.println("Вместимость палатки: " + this.humanCapacity + "
человека");
    }
}
```

Листинг 5 – Класс CannedFood

```
public @Data
@AllArgsConstructor
class CannedFood implements Portable{
    private String product;

    @Override
    public void showInfo() {
```

Окончание листинга 5

```
        System.out.println("Название объекта: консервы");
        System.out.println("Консервированный продукт: " +
this.product.toLowerCase());
    }
}
```

Листинг 6 – Класс Backpack

```
public @Data
class Backpack {
    private final String brandName;
    private Portable burden;
}
```

Листинг 7 – Класс Main

```
public class Main {

    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
                "applicationContext.xml"
        );

        Backpack backpack =  context.getBean("backpack", Backpack.class);
        System.out.println(backpack.getBrandName());
        backpack.getBurden().showInfo();

        context.close();
    }
}
```

4.2 Annotation-конфигурация

На листингах представлен код классов, реализующих задание через аннотации классов.

Листинг 8 – Класс-конфигуратор

```
@Configuration
@ComponentScan("ru.second.annotationConfig")
@PropertySource("classpath:database.properties")
public class SpringConfiguration {

    @Bean
```

Окончание листинга 8

```
    public Shovel shovel() {
        return Shovel.getShovel();
    }
}
```

Листинг 9 – Интерфейс Portable

```
public interface Portable {
    void showInfo();
}
```

Листинг 10 – Класс Shovel

```
@Component
public @Data
class Shovel implements Portable {
    @Value("${ru.second.annotationConfig.shovelTypes}")
    private ShovelTypes type;

    @PostConstruct
    public void init() {
        System.out.println("Создание бина лопаты.");
    }

    private Shovel() {
    }

    public static Shovel getShovel() {
        return new Shovel();
    }

    @Override
    public void showInfo() {
        System.out.println("Название объекта: лопата");
        System.out.println("Тип лопаты: " + this.type.name().toLowerCase());
    }

    @PreDestroy
    public void destroy() {
        System.out.println("Бин лопаты уничтожен");
    }
}
```

Листинг 11 – Класс Tent

```
@Component
public @Data
class Tent implements Portable {
    @Value("${tent.humanCapacity}")
    private int humanCapacity;

    @PostConstruct
    public void init() {
        System.out.println("Создание бина палатки.");
    }

    @Override
    public void showInfo() {
        System.out.println("Название объекта: палатка");
        System.out.println("Вместимость палатки: " + this.humanCapacity + "
человека");
    }

    @PreDestroy
    public void destroy() {
        System.out.println("Бин палатки уничтожен");
    }
}
```

Листинг 12 – Класс CannedFood

```
@Component
public @Data
class CannedFood implements Portable {
    @Value("${cannedFood.product}")
    private String product;

    @PostConstruct
    public void init() {
        System.out.println("Создание бина консервов.");
    }

    @Override
    public void showInfo() {
        System.out.println("Название объекта: консервы");
        System.out.println("Консервированный продукт: " +
this.product.toLowerCase());
```

Окончание листинга 12

```
    }

    @PreDestroy
    public void destroy() {
        System.out.println("Бин консервов уничтожен.");
    }
}
```

Листинг 13 – Класс Backpack

```
@Component
public @Data
class Backpack {
    private final String brandName;
    private Portable burden;

    @PostConstruct
    public void init() {
        System.out.println("Создание бина рюкзака.");
    }

    @Autowired
    public Backpack(@Qualifier("shovel") Portable burden,
                    @Value("${backpack.brandName}") String brandName) {
        this.burden = burden;
        this.brandName = brandName;
    }

    @PreDestroy
    public void destroy() {
        System.out.println("Бин рюкзака уничтожен");
    }
}
```

Листинг 14 – Класс Main

```
public class Main {

    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfiguration.class);

        System.out.println();
```

Окончание листинга 14

```
        Backpack backpack =  context.getBean("backpack", Backpack.class);
        System.out.println(backpack.getBrandName());

        System.out.println("Информация о содержимом рюкзака.");
        backpack.getBurden().showInfo();

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