Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Информатика»

Практический пример №2
Внедрение зависимостей в Spring Framework. Контекст приложения .

Составил: Старший преподаватель кафедры «Информатика»
Черниговский Алексей Сергеевич

Красноярск 2020

Содержание

1 Создание Maven проекта

Apache Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML.

Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.

В Eclipse выбираем File → New → Project выберите Maven Project (Рисунок 2).

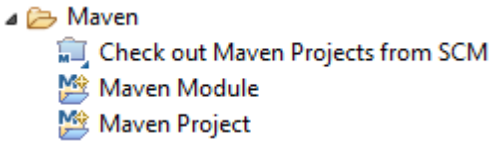

Рисунок 2 – Maven Project

Следующий экран пропускаем, нажимаем Next (Рисунок 3)

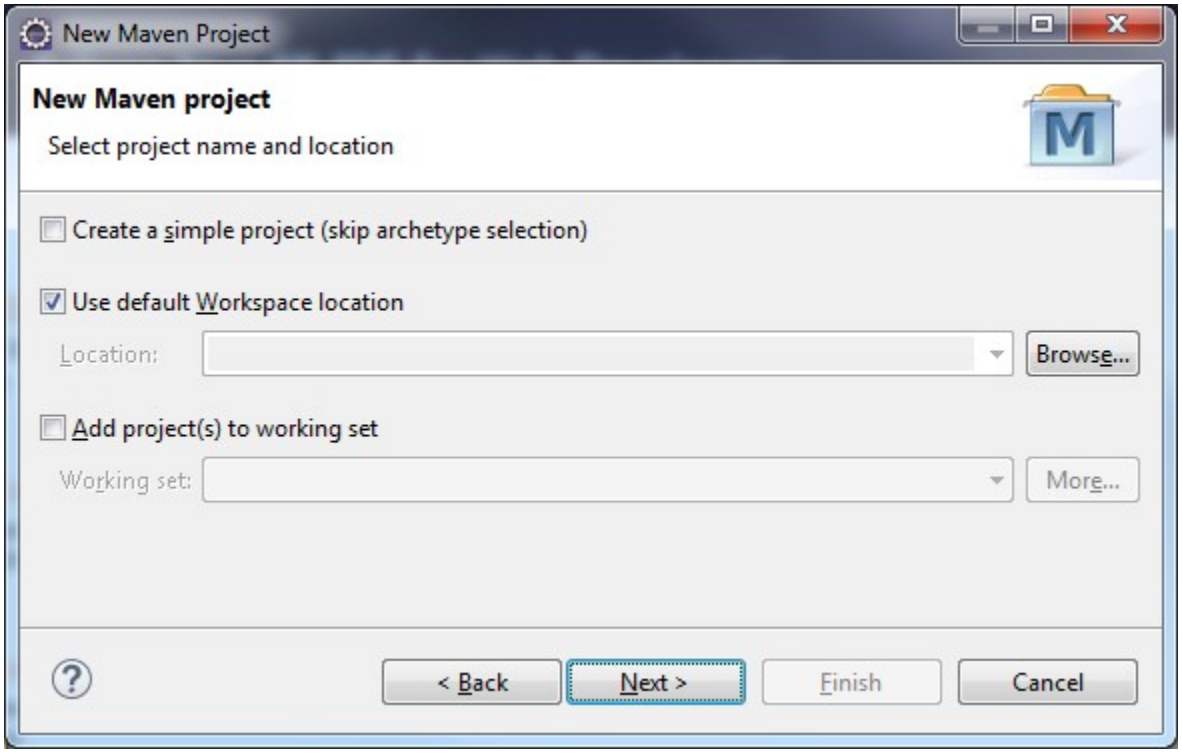

Рисунок 3 – Настройка Maven Project

На следующем экране выберите архетип maven-archetype-webapp (Рисунок 4)

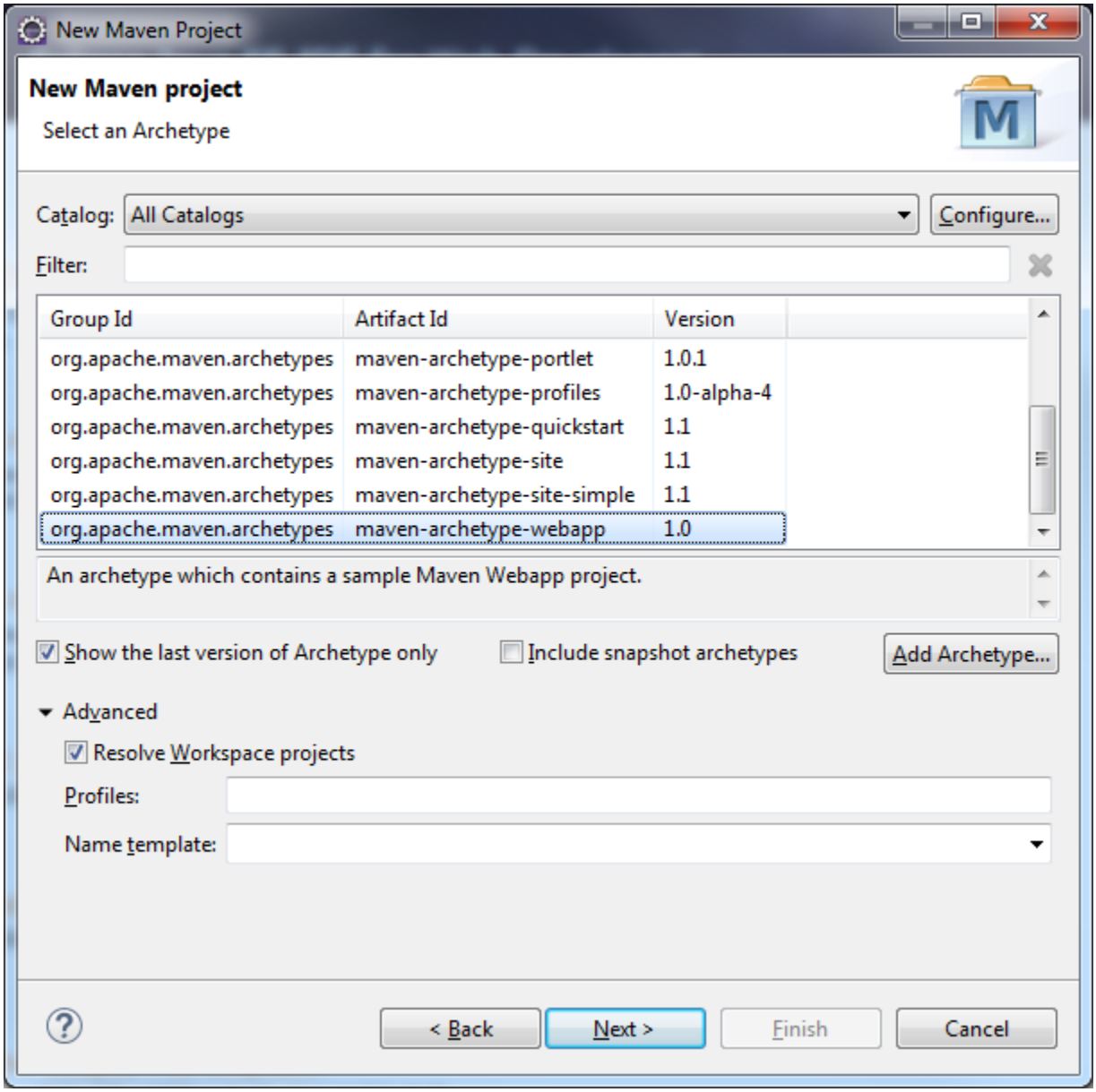

Рисунок 4 – maven-archetype-webapp

Maven использует принцип Maven-архетипов. Архетип — это инструмент шаблонов, каждый из которых определён паттерном или моделью, по аналогии с которой создаются производные.

Maven-archetype-webapp — архетип, создающий веб приложение для запусука в сервере приложений. Включает в себя необходимые дескрипторы и jsp страницу.

В следующем окне (Рисунок 5) укажите Group Id – обычно указывает на разработчика ПО, компанию. И Artifact Id – название проекта. И нажимаем Finish.

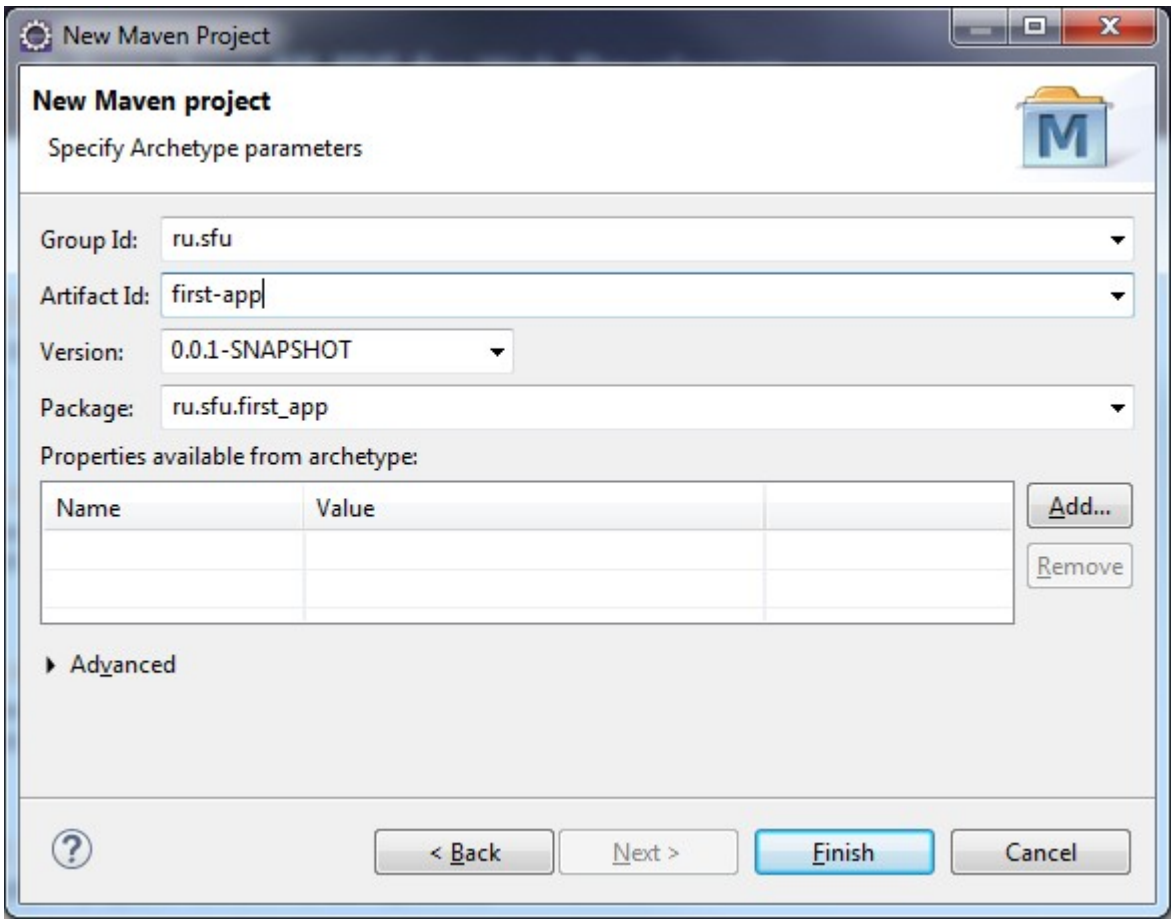

Рисунок 5 – Group Id и Artifact Id

После конфигурации Maven проекта рассмотрим его структуру.

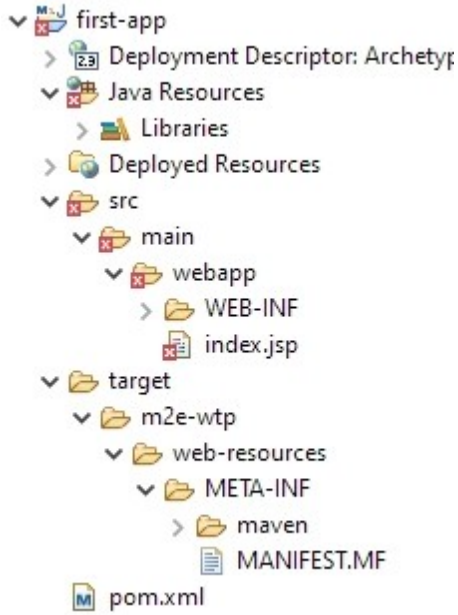

Рисунок 6 – Структура проекта

2 Добавление зависимостей к проекту

Нас интересует файл pom.xml – это главный файл управления конфигурацией Maven. Это и есть тот самый файл описания проекта, на основе которого осуществляются все операции Maven. Он написан на языке POM, входящим в семейство XML.

Нам необходимо добавить зависимости для Spring. Зависимости – следующая очень важная часть pom.xml, тут хранится список всех библиотек (зависимостей) которые используюся в проекте. Каждая библиотека идентифицируется также как и сам проект – тройкой groupId, artifactId, version (GAV). Объявление зависимостей заключено в тэг <dependencies>...</dependencies>.

На https://mvnrepository.com/ находим зависимость Spring Core (Рисунок 7)

Рисунок 7 – Spring Core

Выбираем последний релиз, и копируем зависимость для Maven в наш pom.xml (Рисунок 8).

Рисунок 8 – Зависимость для Maven

Таким же образом добавляем зависимость Spring Beans и Spring Context. Получаем следующий вид тега зависимостей

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.2.8.RELEASE</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>5.2.8.RELEASE</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.8.RELEASE</version>
  </dependency>
</dependencies>
```

Рисунок 9 – Зависимости

Для того чтобы наши изменения применились, в окне обозревателя проектов нажмите правой кнопкой по проекту, а затем Maven→Update Project (Рисунок 10).

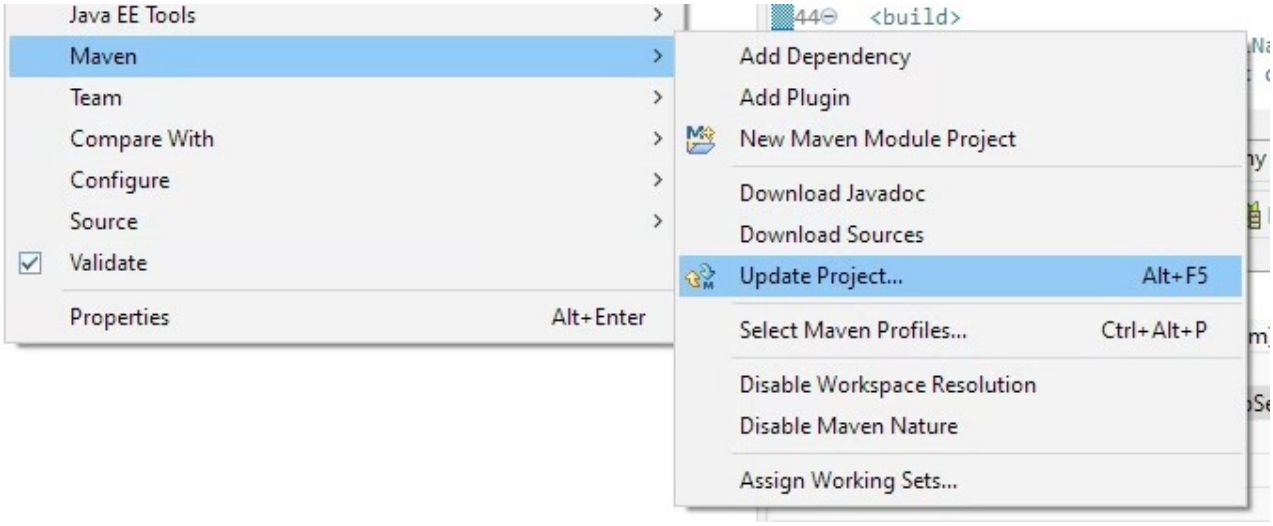

Рисунок 10 – Update Project

3 Добавление собственных Java файлов

Для начала нам необходимо вручную добавить папку, где будет храниться исходный код наших классов. В папке src→main создаем папку java, а в ней ru/firstapp.

Как мы можем увидеть, во вкладке Java Resources появилась наша папка ru.firstapp. Нажимаем правой кнопкой и создаем новый класс TestBean(Рисунок 11).

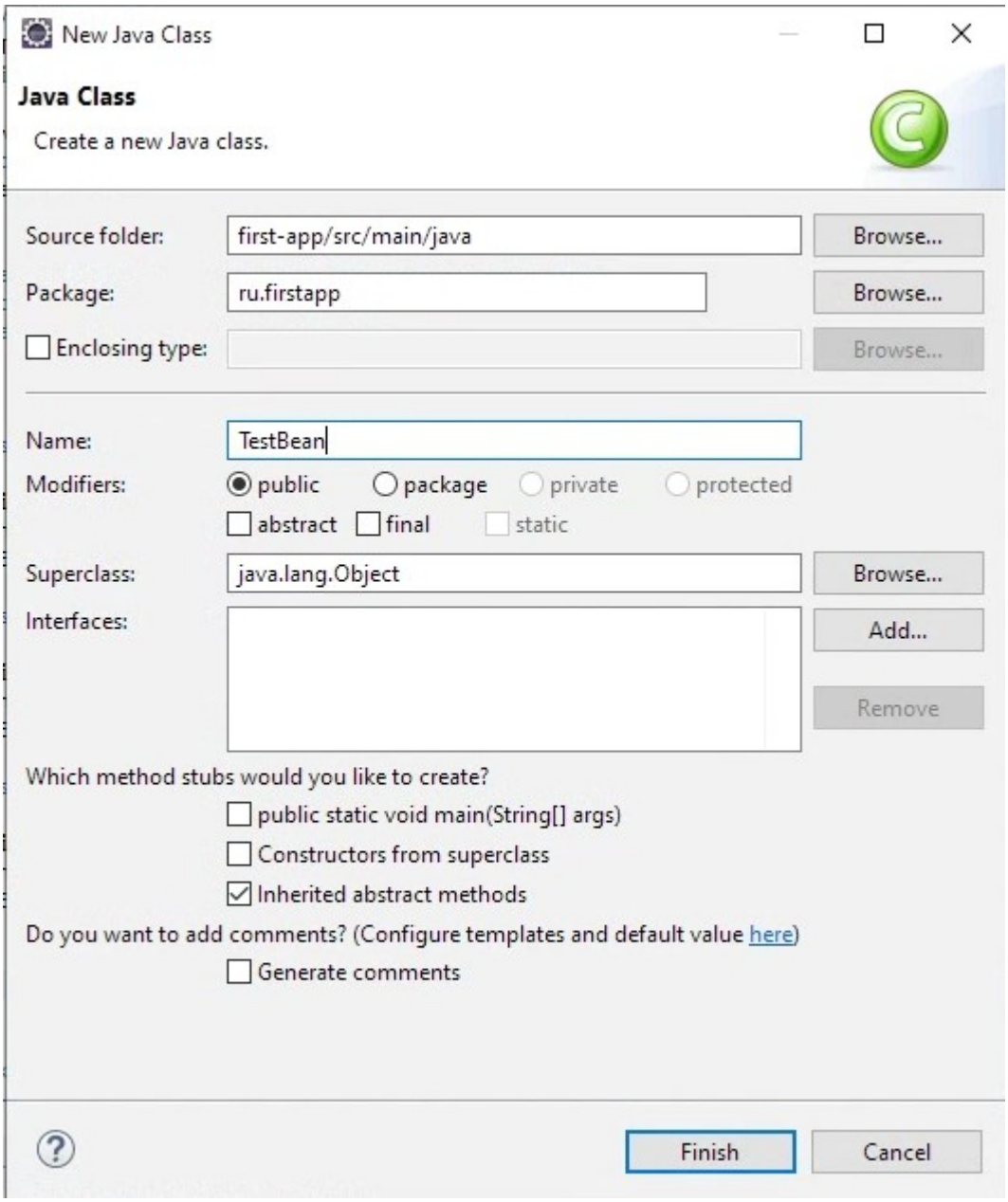

Рисунок 11 – Класс TestBean

Как мы можем увидеть, класс предлагается поместить в пакет ru.firstapp, что нам и необходимо.

Заполните файл следующим кодом:

```
public class TestBean {
	private String name;

	public TestBean(String name) {
```

```
		this.name = name;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
```

4 Создание определения контекста приложения

В папке java создайте папку resources, а в нем файл applicationContext.xml со следующим содержимым:

```
<?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">

	<bean id="testBean"
		class="ru.firstApp.TestBean">
		<constructor-arg value="Vasya"/>
	</bean>

</beans>
```

5 Запуск простейшего Spring приложения с использованием контекста приложения

После чего создайте класс TestSpring в пакете ru.firstapp со следующим содержимым:

```
public class TestSpring {
public static void main(String[] args) {

	ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

	TestBean testBean = context.getBean("testBean", TestBean.class);

	System.out.println(testBean.getName());

	context.close();
	}
}
```

Запустите класс. Должно появиться сообщение.

Об остальном подробнее читайте в конспекте лекции.