Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники**

**(ИИТР)**

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 00**

**по дисциплине**

**«Распределённые программные системы»**

**Тема «Типовая структура Java приложения. Использование системы управления ЖЦ проекта Apache Maven.»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Трифонов Д.А.

Владимир, 2019

ЦЕЛЬ РАБОТЫ

Познакомиться с принципами разработки Java-приложений с использованием автоматизированной системы управления жизненным циклом проекта Apache Maven. Получить практические навыки решения типовых задач при сборке Java проекта.

ВЫПОЛНЕНИЕ РАБОТЫ

1. Создать структуру каталогов, необходимую для организации типового проекта на Java.



Рисунок 1. Структура каталогов проекта.

1. Разработать приложение, решающее выбранную задачу. Приложение должно содержать не менее 2-х классов, а также вспомогательные ресурсы. Организовать сборку приложения с использованием фреймворка Apache Maven. Сгенерировать основу второго приложения с помощью Apache Maven Archetype. Реализовать вспомогательные классы, которые будут использоваться основным приложением.

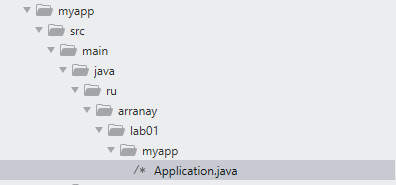


Рисунок 2.Расположения основного класса в каталоге.

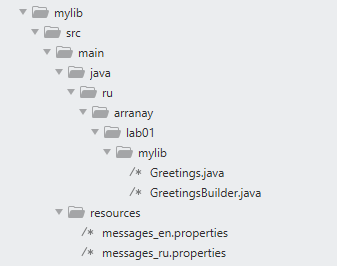


Рисунок 3. Расположение классов и вспомогательных ресурсов в каталоге.

1. Объединить оба приложения в один многомодульный Maven-проект. Определить второй проект как зависимость в конфигурационном файле основного проекта.

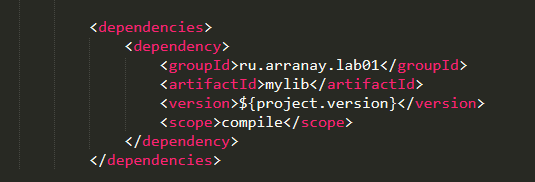


Рисунок 4. В конфигурационном файле myapp/pom.xml определим зависимость от библиотеки mylib.

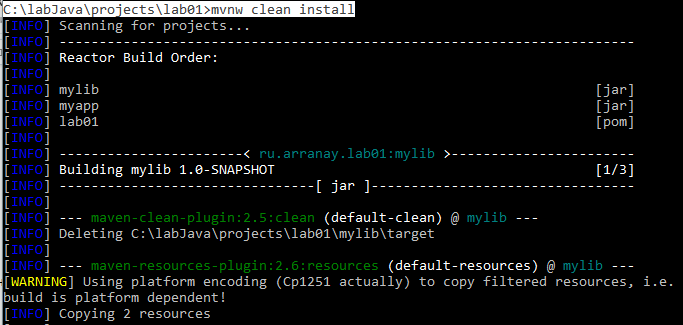


Рисунок 5. Сборка проекта.

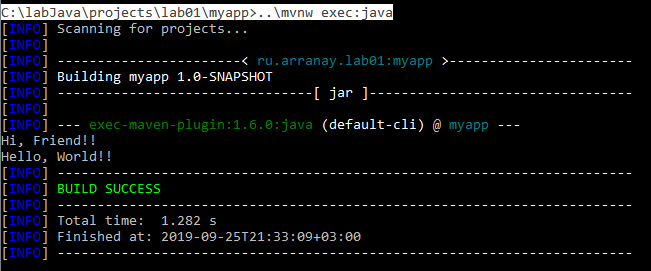


Рисунок 6. Выполнение проекта.

1. Организовать модульное тестирование приложения (1 класс, 2 теста).

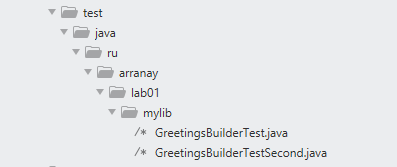


Рисунок 7. Расположение тестов в каталоге проекта.

GreetingsBuilderTest

**package ru.arranay.lab01.mylib;**

**import org.junit.\*;**

**/\*\***

**\* Unit test for GreetingsBuilder class.**

**\*/**

**public class GreetingsBuilderTest {**

**// Выполняется один раз до выполнения тест-методов, например, для соединения с БД, инициализации пула соединений**

**@BeforeClass**

**public static void runOnceBeforeClass() {**

**System.out.println("@BeforeClass - runOnceBeforeClass");**

**}**

**// Выполняется один раз уже после выполнения всех тест-методов, например, для закрытия соединения**

**// и освобождения других ресурсов**

**@AfterClass**

**public static void runOnceAfterClass() {**

**System.out.println("@AfterClass - runOnceAfterClass");**

**}**

**// Выполняется перед каждым тест-методом**

**// Позволяет, например, создать новый экземпляр объекта, необходимого тест-методу**

**@Before**

**public void runBeforeTestMethod() {**

**System.out.println("@Before - runBeforeTestMethod");**

**}**

**// Выполняется после каждого тест-метода**

**@After**

**public void runAfterTestMethod() {**

**System.out.println("@After - runAfterTestMethod");**

**}**

**@Test**

**public void testGreetingsBuilder1() {**

**System.out.println("@Test - run testGreetingsBuilder1");**

**String expectedValue = "Hi, Friend!";**

**GreetingsBuilder greetings = new GreetingsBuilder();**

**String value = greetings.to("Friend")**

**.withLanguage(Greetings.LANGUAGE\_EN)**

**.withStyle(Greetings.STYLE\_FRIENDLY)**

**.build();**

**Assert.assertEquals("English formal greetings check", expectedValue, value);**

**}**

**@Test**

**public void testGreetingsBuilder2() {**

**System.out.println("@Test - run testGreetingsBuilder2");**

**Assert.assertTrue(true);**

**}**

**}**

GreetingsBuilderSecondTest

**package ru.arranay.lab01.mylib;**

**import org.junit.\*;**

**/\*\***

**\* Unit test for GreetingsBuilder class.**

**\*/**

**public class GreetingsBuilderSecondTest {**

**// Выполняется один раз до выполнения тест-методов, например, для соединения с БД, инициализации пула соединений**

**@BeforeClass**

**public static void runOnceBeforeClass() {**

**System.out.println("@BeforeClass - runOnceBeforeClass");**

**}**

**// Выполняется один раз уже после выполнения всех тест-методов, например, для закрытия соединения**

**// и освобождения других ресурсов**

**@AfterClass**

**public static void runOnceAfterClass() {**

**System.out.println("@AfterClass - runOnceAfterClass");**

**}**

**// Выполняется перед каждым тест-методом**

**// Позволяет, например, создать новый экземпляр объекта, необходимого тест-методу**

**@Before**

**public void runBeforeTestMethod() {**

**System.out.println("@Before - runBeforeTestMethod");**

**}**

**// Выполняется после каждого тест-метода**

**@After**

**public void runAfterTestMethod() {**

**System.out.println("@After - runAfterTestMethod");**

**}**

**@Test**

**public void testGreetingsBuilderSecond1() {**

**System.out.println("@Test - run testGreetingsBuilder1");**

**String expectedValue = "Hello, World!";**

**GreetingsBuilder greetings = new GreetingsBuilder();**

**String value = greetings.to("World")**

**.withLanguage(Greetings.LANGUAGE\_EN)**

**.withStyle(Greetings.STYLE\_FORMAL)**

**.build();**

**Assert.assertEquals("English formal greetings check", expectedValue, value);**

**}**

**@Test**

**public void testGreetingsBuilderSecond() {**

**System.out.println("@Test - run testGreetingsBuilder2");**

**Assert.assertTrue(true);**

**}**

**}**

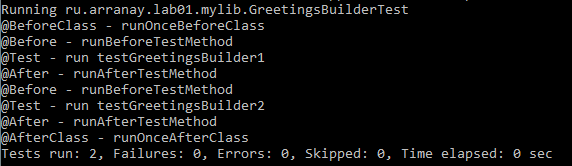


Рисунок 8. Тест GreetingsBuilderTest.

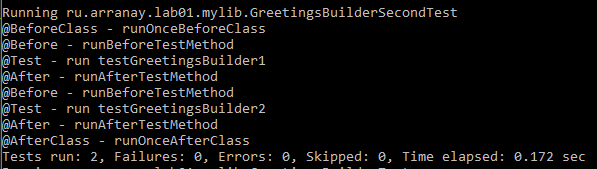


Рисунок 9. Тест GreetingsBuilderSecondTest.

ВЫВОД

В ходе лабораторной работы были изучены принцыпы разработки Java-приложений с использованием автоматизированной системы управления жизненным циклом проекта Apache Maven и получить практические навыки решения типовых задач при сборке Java проекта.