

Автоматизация.
Теория и практика тестирования.

Общие понятия и принципы автоматизации

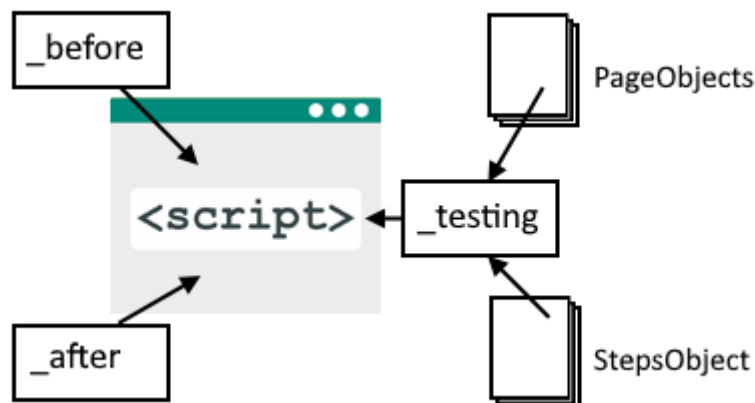
Автоматизированное тестирование - это процесс использования специального программного обеспечения для выполнения проверок с целью сравнения ожидаемого и фактического результата.

Автоматизированное тестирование применяется в:

- регресс тестировании (монотонные и однообразные проверки)
- объемном тестировании (проверка большого количества информации)
- специальное тестирование (проверка особых задач)
- другие виды тестирования требующие автоматизации

Структура автотеста:

1. Описание Page Object класса для получения доступа к элементам приложения.
2. Описание Steps Object класса для выполнения действий над элементами.
3. Инициализация необходимых средств работы автотеста (драйвер и пр.)
4. Описание действий необходимых для воспроизведения шагов теста.
5. Проверка соответствия фактического результата с ожидаемым результатом.
6. Завершение работы автотеста (очистка данных, закрытие драйвера и пр.)



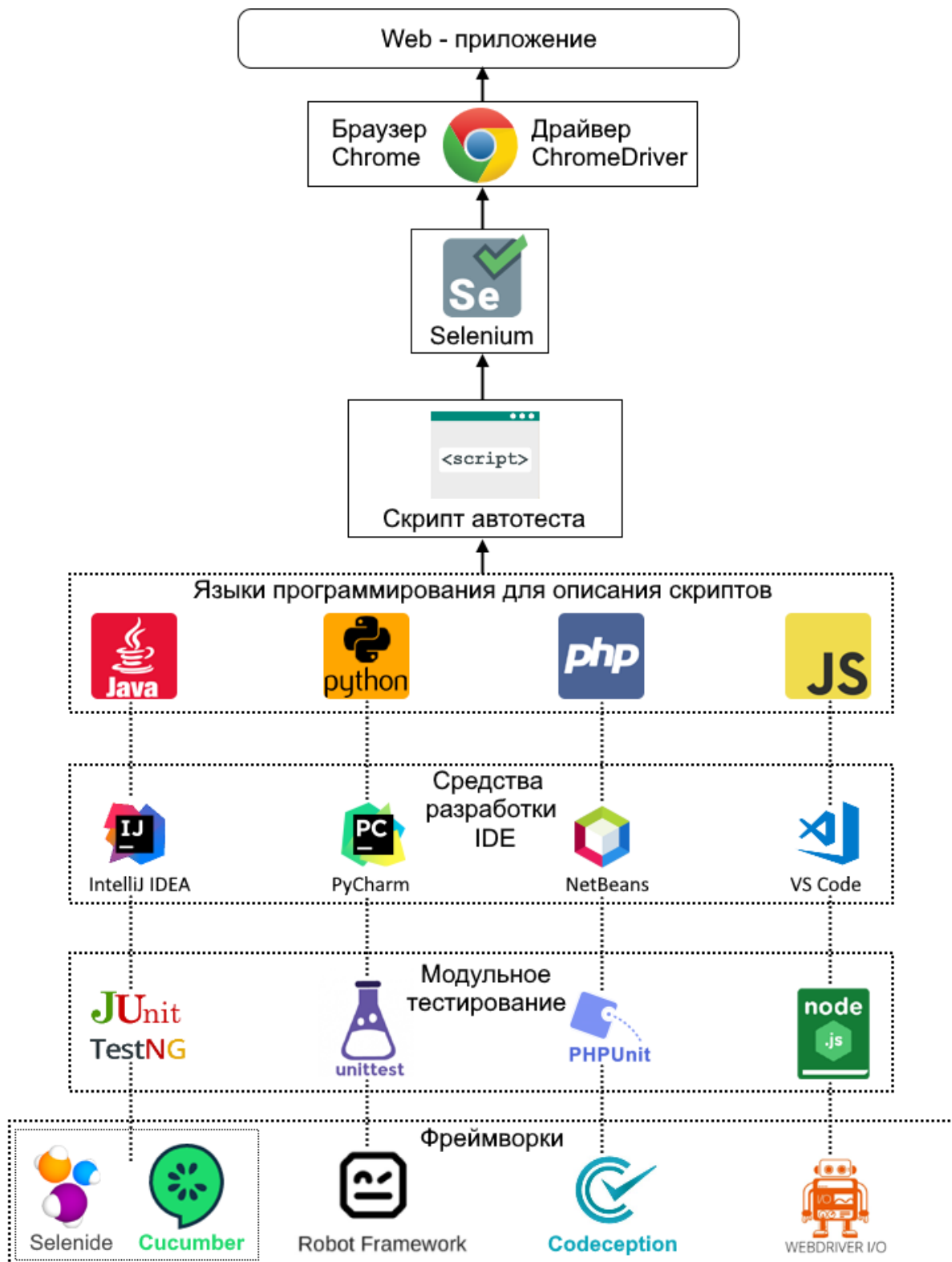
Паттерн (шаблон) суть которых в том, что для каждой страницы тестируемого приложения создаётся отдельный объект, методы которого инкапсулируют логику работы с отдельными элементами.

- Page Objects - описываются элементы страницы что позволяет избежать дублирования локаторов.
- Steps Object - описываются действий над объектами чтобы избежать повторения кода.

Методология автоматизированного тестирования Web приложений:

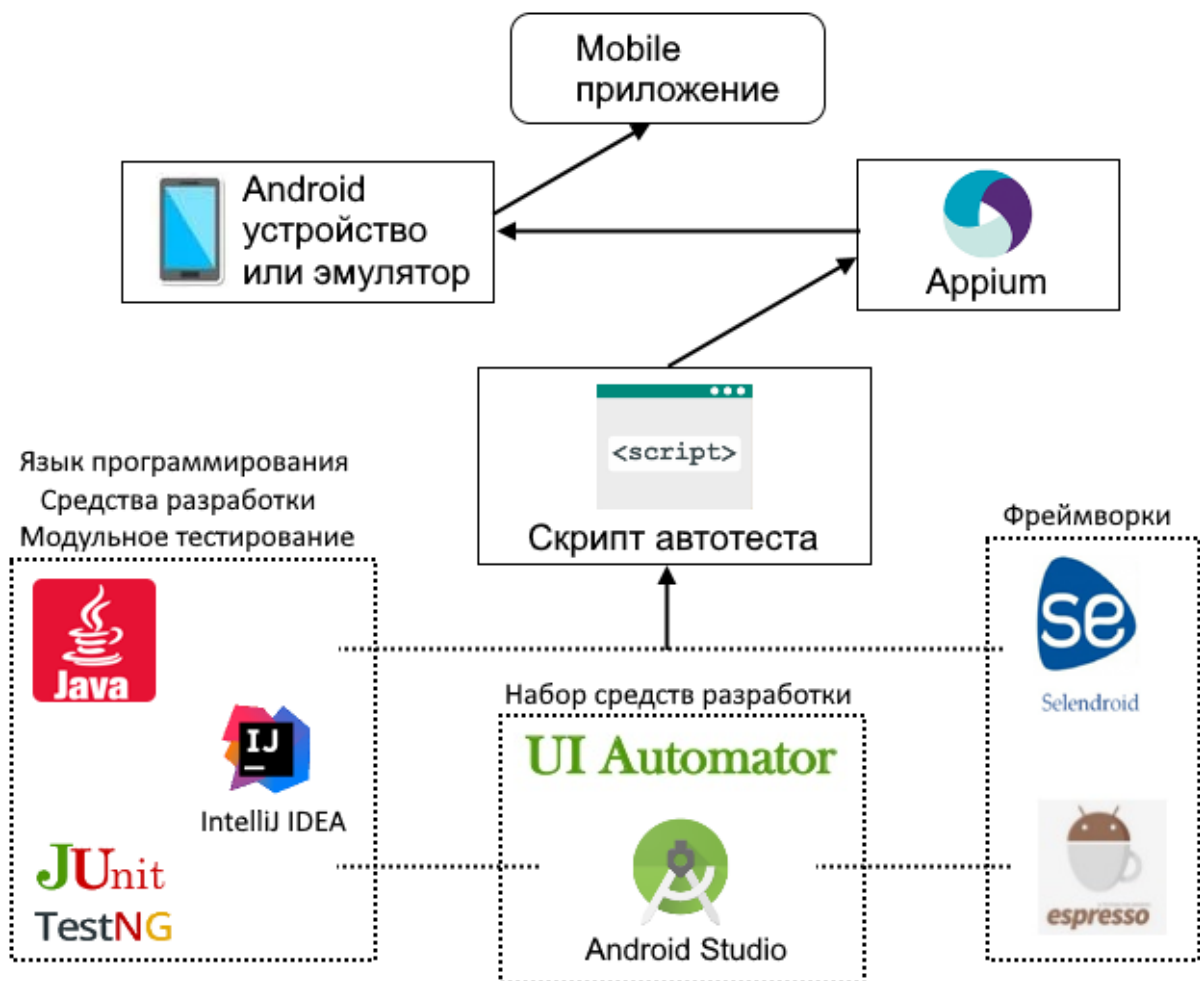
1. Объект тестирования (Web приложение)
2. Драйвер (Selenium)
3. Язык описания скриптов (Java, Python, PHP, JavaScript)
4. Среда разработки IDE (IntelliJ IDEA, PyCharm, NetBeans, Visual Studio Code)
5. Модульное тестирование (JUnit, TestNG, Unittest, pytest, PHPUnit)
6. Фреймворк (Selenide, Cucumber, Robot framework, Codeception, WebDriverIO)
7. Процесс непрерывной интеграции (Jenkins, TeamCity)

Схема автоматизированного тестирования Web приложений:



Методология автоматизированного тестирования Mobile приложений:

1. Объект тестирования (Mobile приложение)
2. Драйвер (Appium)
3. Язык описания скриптов (Java)
4. Среда разработки IDE (IntelliJ IDEA)
5. Модульное тестирование (JUnit, TestNG)
6. Средства разработки (Android SDK from Android Studio)
7. Дополнительные средства (UIAutomator 2)
8. Фреймворк (Selendroid, Espresso)
9. Процесс непрерывной интеграции (Jenkins, TeamCity)



Многофункциональные средства автоматизированного тестирования:



Язык программирования Java, Python, PHP, JS

Инкапсуляция - механизм программирования объединяющий код и данные которыми он манипулирует.

- Объект - это компонент поддерживающий инкапсуляцию.
- Класс - определяет тип объекта.

Полиморфизм - это свойство, позволяющее с помощью одного интерфейса обращаться к общему классу действий. (один интерфейс - множество методов)

Наследование - это процесс в ходе которого один объект приобретает свойства и методы другого объекта.

Объектно-ориентированное программирование - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Синтаксис языка Java

Переменные <pre>int x; int y = 10; double d = 4.5d; float f = 3.14159265f; byte b = 28; boolean b = false; long l = 111111111; char c = 'a'; String s = "text"; String s = new String("text"); public static int x; private ArrayList<String> args;</pre>	Константа <pre>final double pi = 3.14d; public static final String URL = "url";</pre> Преобразование типов данных <pre>long lg = 111111111; Short sh; sh = (Short) lg; int x = 10; String str = (String)x; str = s.toString();</pre>
Массивы <pre>int[] a = new int[5]; String[] s = new String[8]; int[] arrayInt = new int[3]; arrayInt[0] = 10; arrayInt[1] = 20; arrayInt[2] = 30;</pre>	Многомерные массивы <pre>int[][] m = new int[10][6]; m[5][4] = 10; m[5][5] = 20; String[][] s = new String[2][]; s[0] = new String[7]; s[0][6] = "text";</pre>

<pre>String[]str = {"one", "two"}; String[]str = new String[] {"Hello", "World"}; str[1] = "John";</pre>	<p>Преобразование строки в массивы</p> <pre>String s = "text"; char[] c = s.toCharArray();</pre>
<p>ArrayList - неопределенный тип массива</p> <pre>ArrayList<String> list = new ArrayList<String>(); list.add("text"); println(list[0]);</pre>	
<p>Условия</p> <pre>int a = 10; if(a < 10) { println("a < 10"); } else if(a > 10) { println("a > 10"); } else { println("a = 0"); } if(a == 10 && b != 10) println("PASSED"); else println("FAILED");</pre>	<p>Условный оператор ?</p> <pre>int a = 10; a = a < 0 ? 1 : -1; boolean b = a>10 ? true : false;</pre> <hr/> <p> == (равно) != (не равно) > (больше) >= (больше или равно) < (меньше) <= (меньше или равно) && (логическое И) (логическое ИЛИ) </p>
<p>Циклы</p> <pre>for(int i = 0; i < 10; i++) { println(i.toString()); } int a = 10; while(a > 0) { a--; println(a.toString()); } int a = 0; do{ a++; println(a.toString()); }while(a < 10);</pre>	<p>Конструкция выбора</p> <pre>int num = 100; switch (num) { case 99: println("99"); break; case 100: println("100"); break; default: println("other"); }</pre>

Перечисление

```
enum Days
{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}

Day current = Days.MONDAY;
```

Обработка исключений

```
import System.out;

try {
    //...
} catch (Exception ex) {
    printf(ex.getMessage());
}
```

Классы - Простой класс

```
public class MyClass
{
    public String name;
    public void setName(String value)
    {
        name = value;
    }
}

MyClass mc = new MyClass();
mc.setName("John");
println(mc.name);
```

Классы - Конструктор класса

```
class MyClass
{
    String name;
    int age;

    MyClass()
    {
        name = "John";
        age = 30;
    }

    MyClass(String n, int a)
    {
        name = n;
        age = a;
    }

    void displayInfo()
    {
        System.out.printf("Name: %s \t Age: %d \n", name,
age);
    }
}
```

Классы - Наследование

```
public class MyClass1
{
    public String name;
    public int age;
    public void setName(String n, int a)
    {
        name = n;
        age = a;
    }
}

public class MyClass2 extends MyClass1
{
    public MyClass2(String n, int a)
    {
        super(n, a);
    }
}
```

Классы - Абстрактный класс (призван предоставлять базовый функционал для классов-наследников)

```
abstract class MyClass1
{
    private String _name;
    public MyClass1(String name)
    {
        this._name = name;
    }
    public String getName()
    {
        return _name;
    }
    public abstract void show();
}
```

пример использования абстрактного класса

```
public class MyClass2 extends MyClass1
{
    public void show()
    {
        System.out.printf("Name: %s", super.getName());
    }
}

MyClass2 mc2 = new MyClass2();
mc2.show();
```


Интерфейсы (может определять константы и методы, которые могут иметь или не иметь реализацию)

```
interface IMyInterface
{
    void show();
}
public class MyClass implements IMyInterface
{
    public void show()
    {
        System.out.printf("message");
    }
}
MyClass mc = new MyClass();
mc.show();
```

Коллекции

ArrayList - обобщенная коллекция

```
import java.util.ArrayList;

ArrayList<String> list =
    new
    ArrayList<String>();

list.add("John");
list.add("Tom");
list.add(2, "Paul");
list.get(1);
list.set(1, "Bill");
list.size();
list.contains("Bob");
list.remove("Bob");
list.remove(0);
```

ArrayDeque - обобщенная двунаправленная очередь

```
import java.util.ArrayDeque;

ArrayDeque<String> list =
    new
    ArrayDeque<String>();

list.add("John");
list.addFirst("Bill");
list.push("Tom");
list.addLast("Bob");
list.getFirst();
list.getLast();

while(list.peek() != null)
{
    println(list.pop());
}
```

LinkedList - структура данных в виде связанных списков

```
import java.util.LinkedList;

LinkedList<String> list = new
LinkedList<String>();
list.add("Tom");
list.addLast("John");
list.addFirst("Bill");
list.size();
list.get(1);
list.set(1, "Bob");
list.contains("Paul");
list.remove("Bob");
list.removeFirst();
list.removeLast();
```

List - список

```
import java.util.List;

List<String> list = new
List<String>();
list.add("John");
list.get(0);
list.clear();
list.remove(0);
list.set(1, "Tom");

File dir = new
File(pathFolder);
File[] files = dir.listFiles();
List<File> listFiles =
```

```
list.remove(1);

for(String value:list)
{
    println(value);
}
```

```
Array.asList(files != null ?
files:new File[0]);

for(File file:listFiles)
{
    println(file.getPath());
}
```

Лямбда - выражение

```
interface Option {
    int calculate(int x, int y);
}

Option option;
option = (x, y) -> x + y;
option.calculate(10, 20);
```

Чтение и запись текстовых файлов

```
import java.io.*;

// записать в файл
try(FileWriter writer = new FileWriter("note.txt", false))
{
    String text = "Hello World";
    writer.write(text);
    writer.append("\n");
    writer.flush();
}
catch (IOException ex)
{
    println(ex.getMessage());
}

// прочитать из файла
try(FileReader reader = new FileReader("note.txt"))
{
    int i;
    while((i = reader.read()) != -1){
        println((char)i);
    }
}
catch (IOException ex)
{
    println(ex.getMessage());
}
```

Чтение json файлов

```
import org.json.simple.JSONObject;
import org.json.simple.JSONArray;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

Object obj;
File file = new File("C:/file.json");
if(file.exists())
{
    obj = new JSONParser().parser(
                                new
    FileReader("C:/file.json"));
}
JSONObject jsonObj = (JSONObject)obj;

String title = jsonObj.get("title").toString();
ArrayList<String> arguments = new ArrayList<String>();
JSONArray args = (JSONArray) jsonObj.get("arguments");

Iterator argsItr = args.iterator();
while (argsItr.hasNext()) {
    String argument = argsItr.next().toString();
    arguments.add(argument);
}
```

Разное

Определить расширение имени файла

```
if(filename.lastIndexOf("json") >= 0) { }
```

Определить адрес папки с проектом

```
System.getProperty("user.dir");
```

Установить глобальное свойство

```
System.setProperty("webdriver.chrome.driver",
                   "C:/driver/chromedriver.exe");
```

Работа с датой

```
Date date = new Date();
date.toString();
```

Поиск подстроки в строке

```
if(link.contains("mc.yandex.ru")) { }
```

Замена символов в строке

```
String name = filename.replaceAll(" ", "_");
```

Синтаксис языка Python

В Python не используются фигурные скобки для отделения блоков кода, вместо них используются отступы и двоеточие.

Переменные (указывать тип данных не нужно)

```
name = "John"          # текст
age = 30                # число
a = b = c = 1000        # множественное присвоение
print name              # вывод на экран
```

Удаление переменной

```
num = 100
del num                 # удаление переменной
```

Вывод подстроки пять символов из строки

```
text = "Hello World!"
print text[0:5]
```

Списки (массив)

```
my_list = [True, 786, 3.14, 'text', 70.2]
print my_list
print my_list[0]
```

Кортежи (элементы не могут быть изменены)

```
my_tuple = (3.14, 'Pi', False)
print my_tuple[0]
```

Словари (ассоциативный массив)

```
my_dict = {}
my_dict['name'] = 'John'
my_dict['age'] = 30
print my_dict['name']

my_dict = {"name": "John", "age": 30, 2: True, "list": [1, 2, 3]}
print my_dict.keys()
print my_dict.values()
```

Сеты (хранит только уникальные значения)

```
my_set = set()
my_set = {"Hello", "World"}
```

Преобразование типов

```
int(12.4)          # в целое число 12
long(20)           # в длинное число 20L
float(10)          # в число с запятой 10.0
complex(20)        # в комплексное число 20+0j
str(10)            # в строку '10'
tuple("hello")     # в кортеж ("h","e","l","l","o")
list("hello")      # в список ["h","e","l","l","o"]
dict([(1,2),(3,4)]) # в словарь {1:2, 3:4}
```

Условия

```
if n > 100:
    print 'больше 100'
elif n < 100:
    print 'меньше 100'
else:
    print 'равно 100'
```

```
if n > 0:
    print 'больше 0'
    if m > 10:
        print 'больше 10'
    elif m < 10:
        print 'меньше 10'
elif n < 0:
    print 'меньше 0'
else:
    print 'равно 0'
```

Циклы While

```
i = 10
while i > 0:
    print 'шаг %s' % i
    i -= 1
print "цикл завершен"
```

Цикл For

```
word = "text"
for letter in word:
    print letter

list_1 = ["01", "02", "03"];
for item in list_1:
    print item

list_2 = {"01":"text1", "02":"text2"}
for key in list_2:
    print 'ключ %s значение %s' % (key, list_2[key])
```

Перебор элементов по индексу

```
days = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
for index in range(len(days)):
    print days[index]
```

Команда break

```
for day in days:
    if day == "Wed":
```

```
while True:
    if n < 0:
        break
```

<pre>break</pre>	
<p>Функции</p> <pre>def my_function(argument): print argument</pre> <p>вызов функции</p> <pre>my_function("Hello World!")</pre>	
<p>Обязательные аргументы функции</p> <pre>def my_function(a, b): if a > b: print a else print b</pre> <p>вызов функции</p> <pre>my_function(5, 6)</pre>	<p>Аргументы - ключевые слова</p> <pre>def my_function(name, age): print name, "is", age</pre> <p>вызов функции</p> <pre>my_function(age=30, name="John")</pre>
<p>Аргументы заданные по умолчанию</p> <pre>def my_function(x=10, y=20) print x, " : ", y</pre> <p>вызов функции</p> <pre>my_function()</pre>	<p>Аргументы произвольной длины</p> <pre>def my_function(*args): for a in args: print a</pre> <p>вызов функции</p> <pre>my_function(1,2,3,4,5) my_function("Hello", "World")</pre>
<p>Возврат значения</p> <pre>def my_function(a, b): return a+b</pre> <p>вызов функции</p> <pre>n = my_function(5, 4)</pre>	<p>Область видимости</p> <pre>age = 44 def global_info(): global age age += 1 print age</pre> <pre>def local_info(): age = 22 print age</pre> <p>вызов функции</p> <pre>print age # показано 44 global_info() # показано 45</pre>

	<code>local_info()</code> # показано 22
--	---

Функциональное программирование

Лямбда выражение (анонимная функция)

```
multiply = lambda x,y: x * y  
n = multiply(21, 2)
```

Функция **map()** - применяется к каждому элементу списка

```
old_list = ['1','2','3','4','5']  
new_list = list(map(int, old_list))
```

аналогично можно так

```
new_list = []  
for item in old_list:  
    new_list.append(int(item))
```

использование **map** и **lambda**

```
array1 = [1,2,3]  
array2 = [4,5,6]  
new_list = list(map(lambda x,y: x+y, array1, array2))  
print(new_list)              # в результате [5,7,9]
```

Функция **filter()** - фильтрация элементов последовательности

```
mixed = ['windows', 'linux', 'mac']  
result = list(filter(lambda x: x=='mac', mixed))  
print(result)              # в результате [mac,mac,mac]
```

Функция **reduce()**

```
from functools import reduce  
  
items = [1,24,17,14,9,32,2]  
result = reduce(lambda a,b: a if(a>b) else b, items)  
print(result)              # в результате 32 наибольший элемент
```

Функция **zip()** - объединяет элементы в кортеж

```
a = [1,2,3]  
b = "xyz"  
c = (None, True)  
  
result = list(zip(a,b,c))
```

```
print(result)          # в результате [(1, 'x', None), (2, 'y', True)]
```

Объектно-ориентированное программирование

создание класса

```
class MyClass:
    myperem = 0

    def __init__(self, name):
        self.name = name
        self.myperem = 0

    def add(self):
        self.myperem += 1
        MyClass.myperem += 1

    def info(self):
        print(self.name, self.myperem, MyClass.myperem)
```

создание экземпляров класса

```
mc1 = MyClass("text1")
mc2 = MyClass("text2")
```

обращение к атрибутам и методам класса

```
mc1.add()
mc1.info()
```

Модули

импортирует модуль

```
import math
print(math.sqrt(9))
print(math.pi)
```

импортировать определенный метод модуля

```
from math import sqrt
print(sqrt(144))
```

импортировать все методы из модуля

```
from math import *
print(sqrt(121))
print(pi)
```

получение всех модулей


```
help("modules")
```

создание своего модуля

```
def hello():  
    print("Hello World!")
```

(сохранить в файл mymodule.py)

вызов модуля

```
from mymodule import hello  
hello()
```

Работа с файлами

```
my_file = open("file.txt", "w")  
print(my_file.name)  
print(my_file.closed, my_file.mode, my_file.softspace)  
my_file.write("text")  
str = my_file.read()  
print(str)  
my_file.close()
```

Обработка исключений

```
try:  
    i = float(input("Введите число"))  
    print(100/i)  
except ValueError:  
    print("Это не число")  
except ZeroDivisionError:  
    print("На ноль делить нельзя")  
except:  
    print("Неожиданная ошибка");  
else:  
    print("Выполнено без ошибок")  
finally:  
    print("Это выполняется в любом случае")
```

Классы

Наследование класса

```
class MyClass1:  
    def info_1(self):  
        print("это класс 1")  
  
class MyClass2:  
    def info_2(self):  
        print("это класс 2")
```

применение наследование при создании класса

```
class MyClass3(MyClass1, MyClass2):  
    pass
```

```
mc = MyClass3()
mc.info_1()
mc.info_2()
```

Абстрактные классы

```
from abc import ABC, abstractmethod
class Basic(ABC):
    @abstractmethod
    def show(self):
        print("Hello World!");
```

Синтаксис языка PHP

<p>Переменные</p> <pre>\$a = 'строка'; \$b = "строка"; \$c = 25; \$d = 1.25; \$e = false;</pre>	<p>Комментарий</p> <pre>/* многострочный комментарий */ // однострочный комментарий # однострочный комментарий</pre> <hr/> <p>Константы</p> <pre>define("name", "value"); const MIN_VALUE = 0.0;</pre>
<p>Массивы</p> <pre>\$arr = array(1, 2, 3, 4); echo \$arr[1]; \$arr = array("name" => "user", "pass" => "0000"); echo \$arr["name"]; print_r(\$arr);</pre>	<p>Ассоциативный массив</p> <pre>\$arr = array(0 => array(1,2), 1 => array(1,2), 2 => array(1,2)); echo \$arr[0]; // Начиная с PHP 5.4 \$arr = ["name" => "user", "pass" => "0000"]; echo \$arr["name"];</pre>
<p>Условия</p> <pre>if(value > 10){ print("больше 10 \n"); }elseif(value < 10) { print("меньше 10 \n");</pre>	<pre>\$result = (value > 0) ? 'Yes' : 'No'; == (равно) != (не равно) > (больше) >= (больше или равно)</pre>

<pre> }else { print("равно 0 \n"); } </pre>	<p>< (меньше) <= (меньше или равно) && (логическое И) (логическое ИЛИ)</p>
<p>Циклы</p> <pre> \$i = 10; while (\$i > 0){ print("Число: \$i \n"); \$i--; } \$i = 0; do { \$i++; print('Число: '.\$i.\n'); } while (\$i < 10); for (\$i = 0; \$i < 10; \$i++){ print_r(\$i); } </pre>	<pre> \$arr = ["name" => "user", "pass" => "0000"]; foreach (\$arr as \$key => \$value){ echo \$key." : ".\$value; } \$arr = [1,2,3,4]; foreach (\$arr as &\$value){ \$value = \$value * 2; print("Число: \$value \n"); } </pre>
<p>Конструкция выбора</p> <pre> \$i = 1; switch (\$i) { case 0: echo "i равно 0"; break; case 1: echo "i равно 1"; break; default: echo "i больше 1"; } </pre>	<p>Функция</p> <pre> \$a = 25; function myfunc(\$arg1, &\$arg2) { \$arg2 = 0; // меняет знач. return \$arg1 * 2; } \$b = myfunc(5, \$a); print("\$a : \$b"); // 0 : 10 </pre> <hr/> <p>Анонимная функция</p> <pre> echo function () { return 'Hello World!'; } </pre>
<p>Классы</p> <pre> class MyClass { function myFunc() { return "text"; } } \$obj = new MyClass; \$message = \$obj->myFunc(); </pre>	<p>Статичный класс</p> <pre> class MyClass { public static \$my_name = 'no name'; public static function myFunc() { echo \$this->my_name; } } </pre>

<pre>print_r(\$message);</pre>	<pre> } } MyClass::my_name = 'John'; MyClass::myFunc(); </pre>
<p>Конструктор и наследование класса</p> <pre> class MyClass1 { function __construct() { print "Конструктор 1"; } } class MyClass2 extends MyClass1 { protected int \$x; protected int \$y; function __construct(int \$x = 0, int \$y = 0) { parent::__construct(); \$this->x = \$x; \$this->y = \$y; print "Конструктор 2"; } } \$mc2 = new MyClass1(); \$mc2 = new MyClass2(10, 20); </pre>	
<p>Модули и пространство имен</p> <pre> namespace Step\Acceptance; class TestSteps { const URL = "url"; public static \$url = "url"; public function test(){ //... } } </pre> <p>использование модуля</p> <pre> use Step\Acceptance\TestSteps as TestTester; class Testing { function admin(){ \$admin = new TestTester(); \$admin->test(); } } </pre>	<p>Обработка исключений</p> <pre> try { //... } catch (Exception \$e) { echo \$e->getMessage(); } finally { //... } /* Выключение * протоколирования ошибок */ error_reporting(0); </pre>

<pre> } } </pre>	
------------------------------	--

Синтаксис языка JavaScript

Переменные <pre> var a = 'строка'; var b = "строка"; var c = 25; var d = 1.25; let x = 100; // ES6 </pre>	Константы <pre> const MIN_VALUE = 5; // ES6 </pre>
Преобразование типов <pre> 10 + "Текст" // число 10 преобразуется в текст "10 Текст" "7" * "4" // текст преобразуется в число 7 * 4 n + " Текст" // n равно NaN это будет преобразовано в строку "NaN Текст" </pre> <p>Явное преобразование типов с помощью функций:</p> <pre> Number("3") String(100) Boolean([]) Object(3) n.toString() </pre> <p>Функции синтаксического анализа:</p> <pre> parseInt("3 мышки") // получим число 3 parseFloat("3.14 метра") // получим число 3.14 </pre>	
Условия <pre> if(value > 10){ //... }else if(value > 0) { //... }else { //... } </pre> <p>Условный оператор ?</p>	Циклы <pre> var i = 10; while (i > 0){ i--; } do { i++; } while (i < 10); for (var i = 0; i < 10; i++){ </pre>

<pre>var result = value > 0 ? 'yes' : 'no';</pre>	<pre> console.log(i); } var obj = {x:0, y:0}; for(var key in obj) console.log(obj[key]);</pre>
<pre>var arr = [1,2,3,4,5]; arr.forEach(function(item, i, arr){ alert(i + " : " + item); });</pre>	
<p>Конструкция выбора</p> <pre>var i = 5; switch (i) { case 0: alert("i равно 0"); break; case 1: alert("i равно 1"); break; default: alert("i больше 1"); }</pre>	<p>Массивы</p> <pre>var arr = new Array(1, 2, 3, 4); var arr = [1, 2, 3, 4]; var arr = []; arr["name"] = "user"; arr["pass"] = "0000"; var matrix = [[1,2],[3,4],[5,6]]; var arr = [1, {x:1, y:2}], 2, {x:3, y:4}]];</pre>
<p>Объекты</p> <pre>var point = {x:0, y:0}; var book = { "main title": "JavaScript", "sub-title": "Pocket Ref" }; var obj = new Object(); var obj = Object.create({x:1, y:2}); var obj = { x: 1.0, get funcValue() { return this.x; }, set funcValue(value){ this.x = value; } };</pre>	<p>Создание объекта — выполняется с помощью ключевого слова new</p> <p>Если аргументы передавать не нужно скобки можно опустить</p> <pre>new Object; new Data; new Object(); new Point(2,3);</pre>

Функции

```
var a = 25;
function myfunc(arg1, arg2)
{
    window.a = 0;
    return $arg1 * 2;
}
myfunc(5, a);
```

```
var point = function(arg1,
arg2)
{
    window.a = 0;
    return $arg1 * 2;
}
```

Анонимные функции

```
alert(function () {
    return 'Hello World!';
});
```

Классы

```
function MyClass(value)
{
    this.value = value;
}

MyClass.prototype.myFunc = function()
{
    return this.value;
};
```

использование класса

```
var c = new MyClass('текст');
c.myFunc();
```

```
function MyClass()
{
    this.value = 'значение';
    this.myFunc = function() {
        return this.value;
    };
}
```

использование класса

```
var c = new MyClass();
c.myFunc();
```

```
function MyClass(value)
{
    var class = Object.create(MyClass.methods);
    class.value = value;
```

```

        return class;
    }

    MyClass.methods = {
        myFunc: function() {
            return this.value;
        }
    };
};

```

использование класса

```

var c = MyClass('текст');
c.myFunc();

```

Наследование

```

function MyClass1()
{
    //...
}
MyClass1.prototype.func = function(){}

function MyClass2()
{
    //...
}
MyClass2.prototype = Object.create(MyClass1.prototype);

```

Модули и пространство имен

```

(function () {
    var CONNECT_OK = 1;
    function connect() {
        //...
    }
})();

```

```

var MyProject = (function() {
    return {
        CONNECT_OK: 1,
        connect: connect() {
            //...
        }
    }
})();

```

События

```

document.getElementById("myButton").
addEventListener("click", onClick);

document.getElementById("myButton").
removeEventListener("click", onClick);

function onClick(){}

window.onload = function() {
    alert('Документ загружен' );
};

```

Обработка исключений

```

try
{

```


<pre>try { //... } catch (er) { document.write(er); }</pre>	<pre>//... } finally { //... }</pre>
Исключение — сигнал о произошедшей ошибке <pre>if(x < 0) throw new Error("x должен быть положительным");</pre>	
Директива строгого режима <pre>"use strict"</pre>	Вывод сообщения в консоль <pre>console.log('ERROR', 'message');</pre>

Язык запросов SQL

SELECT - выборка данных

Выбрать информацию всех полей из таблицы

```
SELECT * FROM ИмяТаблицы
```

Выбрать информацию указанных полей из таблицы

```
SELECT Поле1, Поле2, ПолеN FROM ИмяТаблицы
```

DISTINCT - устраняет повторяющиеся значения.

```
SELECT DISTINCT Поле1, Поле2 FROM ИмяТаблицы
```

ALL - покажет все записи даже те которые повторяются

```
SELECT ALL Поле1, Поле2 FROM ИмяТаблицы
```

Отбор строк по условию

```
SELECT Поле1, Поле2 FROM ИмяТаблицы WHERE (Поле1 = Значение)
```

Реляционные операторы

- = равный чему-либо;
- > больше чем;
- < меньше чем;
- >= больше чем или равно;
- <= меньше чем или равно;
- <> не равно;

```
SELECT * FROM ИмяТаблицы WHERE (text = 'Значение')
```

```
SELECT * FROM ИмяТаблицы WHERE (num = 100)
```

```
SELECT * FROM ИмяТаблицы WHERE (num > 100)
```

```
SELECT * FROM ИмяТаблицы WHERE (num < 100)
```

```
SELECT * FROM ИмяТаблицы WHERE (num >= 100)
```

```
SELECT * FROM ИмяТаблицы WHERE (num <= 100)
```

```
SELECT * FROM ИмяТаблицы WHERE (myDate = 05/12/2012)
```

Стандартный булевый оператор AND(И).

```
SELECT * FROM ИмяТаблицы WHERE (Поле1 = 100 AND Поле2 = 'Знач')
```

Стандартный булевый оператор OR (ИЛИ)

```
SELECT * FROM ИмяТаблицы WHERE (Поле1 = 100 OR Поле2 = 'Знач')
```

Стандартный булевый оператор NOT (НЕ) отрицание.

```
SELECT * FROM ИмяТаблицы WHERE NOT(Поле1 = 100 AND Поле2 = 'Знач')
```

Отбор значений в которые не входят указанные значения

```
SELECT * FROM ИмяТаблицы WHERE Поле NOT IN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE NOT Поле IN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE Поле NOT BETWEEN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE NOT Поле BETWEEN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE Поле NOT LIKE ('Знач%')
SELECT * FROM ИмяТаблицы WHERE NOT Поле LIKE ('%нач%')
SELECT * FROM ИмяТаблицы WHERE NOT ИмяПоле IS NULL
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле NOT NULL)
SELECT * FROM ИмяТаблицы WHERE ИмяПоле NOT IN (2000, 2001)
```

Оператор IN определяет набор значений с помощью списка.

```
SELECT * FROM ИмяТаблицы WHERE Поле IN ('Значение1', 'Значение2')
SELECT * FROM ИмяТаблицы WHERE Поле IN (100, 200)
SELECT * FROM ИмяТаблицы WHERE Поле IN (17.00, 25.50)
```

Отбор значений в которые не входят указанные значения

```
SELECT * FROM ИмяТаблицы WHERE Поле NOT IN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE NOT Поле IN (2000, 2001)
```

Оператор BETWEEN определяет диапазон значений

```
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле BETWEEN 3000 AND 5000)
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле BETWEEN 'к' AND 'с')
```

Отбор значений которые не входят в диапазон

```
SELECT * FROM ИмяТаблицы WHERE Поле NOT BETWEEN (2000, 2001)
SELECT * FROM ИмяТаблицы WHERE NOT Поле BETWEEN (2000, 2001)
```

Оператор LIKE ищет подстроки (применяется только к полям типа CHAR)

 Символ подчёркивания заменяет любой одиночный символ

% Знак процента заменяет последовательность любого числа символом.

```
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле LIKE 'Знач%')
```

Отбор значений которые не соответствуют значению условия

```
SELECT * FROM ИмяТаблицы WHERE Поле NOT LIKE ('Знач%')
SELECT * FROM ИмяТаблицы WHERE NOT Поле LIKE ('%нач%')
```

Оператор IS различает неверное и неизвестное значение.

(ключевое слово NULL)

```
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле IS NULL)
```

```
SELECT * FROM ИмяТаблицы WHERE NOT ИмяПоле IS NULL
SELECT * FROM ИмяТаблицы WHERE (ИмяПоле NOT NULL)
SELECT * FROM ИмяТаблицы WHERE ИмяПоле NOT IN (2000, 2001)
```

INSERT - добавление данных

Добавление новой записи в таблицу

```
INSERT INTO ИмяТаблицы (Поле1, ПолеN) VALUES (Значение, Значение)
INSERT INTO ИмяТаблицы (Поле1, ПолеN) VALUES (1500, 'Текст')
```

UPDATE - обновление данных

Внесение изменений в запись таблицы

```
UPDATE ИмяТаблицы SET Поле1 = Значение, Поле2 = Значение
WHERE ПолеID = Значение
```

DELETE - удаление данных

Удалить запись в таблице

```
DELETE * FROM ИмяТаблицы WHERE ПолеИдентификатор = Значение
DELETE * FROM ИмяТаблицы WHERE id = 13
```

Получение итоговых результатов

COUNT - Производит подсчет количества строк или NULL значений полей, которые выбрал запрос.

SUM - рассчитывает арифметическую сумму всех выбранных значений данного поля.

AVG - производит усреднение всех выбранных значений данного поля.

MAX - находит и возвращает наибольшее из всех выбранных значений данного поля.

MIN - находит и возвращает наименьшее из всех выбранных значений данного поля.

COUNT - подсчет количества строк

```
SELECT COUNT (*) FROM ИмяТаблицы
SELECT COUNT (DISTINCT ИмяПоля) FROM ИмяТаблицы
SELECT COUNT (ALL ИмяПоля) FROM ИмяТаблицы
```

MAX - находит и возвращает наибольшее значение

```
SELECT MAX (ИмяПоля * 2) FROM ИмяТаблицы
```

MIN - находит и возвращает наименьшее значение

```
SELECT MIN (ИмяПоля) FROM ИмяТаблицы GROUP BY ИмяПоля
```

GROUP BY - позволяет группировать результат запроса.

```
SELECT * FROM ИмяТаблицы GROUP BY Поля1, Поле2
SELECT * FROM ИмяТаблицы WHERE (ПолеDate = 10/06/2012)
```

GROUP BY ИмяПоля

HAVING - отбирает все значения входящие в условие

```
SELECT * FROM ИмяТаблицы GROUP BY ИмяПоля  
HAVING MIN (ИмяПоля) < Значение
```

```
SELECT * FROM ИмяТаблицы GROUP BY Num HAVING MIN (Num) < 5
```

Сортировка результата запроса

ASC - сортирует по возрастанию

DESC - сортирует по убыванию

Можно указать значение по которому нужно сортировать прямо в запросе.

```
SELECT Поле1, Поле2, 'Значение' FROM ИмяТаблицы GROUP BY ИмяПоля.
```

ASC сортирует по возрастанию.

ORDER BY позволяет сортировать вывод запроса согласно значениям.

```
SELECT * FROM ИмяТаблицы ORDER BY ИмяПоля ASC
```

DESC сортирует по убыванию.

ORDER BY позволяет сортировать вывод запроса согласно значениям.

```
SELECT * FROM ИмяТаблицы ORDER BY ИмяПоля ASC
```

Оператор UNION объединяет вывод двух и более SQL запросов

```
SELECT * FROM ИмяТаблицы UNION SELECT * FROM ИмяТаблицы
```

```
SELECT * FROM ИмяТаблицы  
WHERE ИмяПоля BETWEEN 'к' AND 'с'  
UNION  
SELECT * FROM ИмяТаблицы  
WHERE ИмяПоля BETWEEN 'к' AND 'с'
```

Многотабличные запросы

При обращении можно указать несколько таблиц, а к полям обращаться через точку.

```
SELECT Таблица1.Поле, Таблица2.Поле  
FROM Таблица1, Таблица2  
WHERE Таблица1.Поле < Таблица2.Поле  
AND Таблица1.Поле BETWEEN 'к' AND 'с'
```

Объединение таблиц

Объединение вместе двух копий одиночной таблицы.

Чтобы сослаться к столбцам внутри запроса, необходимо иметь два различных имени для этой таблицы.

Это можно сделать с помощью определения временных имён, называемых "псевдонимами".

Объединение одной таблицы

```
SELECT Псевдоним1.Поле, Псевдоним2.Поле
FROM ИмяТаблицы Псевдоним1,
      ИмяТаблицы Псевдоним2
WHERE Псевдоним1.Поле = Псевдоним2.Поле
```

ИмяТаблицы - это одна и та же таблица.

Объединение разных таблиц

```
SELECT Таблица1.Поле, Таблица2.Поле
FROM Таблица1, Таблица2
WHERE Таблица1.Поле = Таблица2.Поле
```

Внутреннее соединение (без JOIN)

```
SELECT Таблица1.Поле, Таблица2.Поле
FROM Таблица1, Таблица2
WHERE Таблица1.Поле *= Таблица2.Поле
```

```
SELECT Таблица1.Поле, Таблица2.Поле
FROM Таблица1, Таблица2
WHERE Таблица1.Поле *=* Таблица2.Поле
```

JOIN - внешнее соединение

```
SELECT Таблица1.Поле2 FROM Таблица1
JOIN Таблица2 ON Таблица1.Поле1 = Таблица2.Поле1
WHERE Таблица1.Поле3 > 100;
```

INNER JOIN - соединение по равенству

```
SELECT Таблица1.Поле2 FROM Таблица1
INNER JOIN Таблица2 ON Таблица1.Поле1 = Таблица2.Поле1
WHERE Таблица1.Поле3 > 100;
```

NATURAL JOIN - естественное соединение

(связанные столбцы имеют одни и те же имена в обеих таблицах)

```
SELECT Поле1, Поле2 FROM Таблица1 NATURAL JOIN Таблица2;
```

USING - связанные столбцы

(можно явно указать связанные столбцы)

```
SELECT Поле2 FROM Таблица1 JOIN Таблица2  
USING (Таблица1.Поле1, Таблица2.Поле1);
```

AS - Псевдонимы таблиц

```
SELECT Таб1.Поле2, Таба2.Поле2  
FROM Таблица1 AS Таб1, Таблица2 AS Таб2  
WHERE Таб1.Поле1 = Таб2.Поле1;
```

FULL — Полное соединение (выводит все строки из обеих таблиц)

```
SELECT * FROM Таблица1  
FULL OUTER JOIN Таблица2  
ON Таблица1.Поле2 = Таблица2.Поле2
```

LEFT — Левое соединение (все строки из левой таблицы, и связанные строки из правой таблицы)

```
SELECT * FROM Таблица1 LEFT OUTER JOIN Таблица2  
ON Таблица1.Поле1 = Таблица2.Поле1;
```

RIGHT — Правое соединение (все строки из правой таблицы, и связанные строки из левой таблицы)

```
SELECT * FROM Таблица1 RIGHT OUTER JOIN Таблица2  
ON Таблица1.Поле1 = Таблица2.Поле1;
```

CROSS — Перекрёстные соединения

```
SELECT * FROM Таблица1 CROSS JOIN Таблица2
```

Использование вложенных запросов

Вложенный запрос вывода значений двух различных таблиц.

```
SELECT * FROM Таблица1  
WHERE Поле =  
    (SELECT Поле1, Поле2 FROM Таблица2 WHERE Поле2 = 'Значение');
```

EXISTS - берет подзапрос как аргумент и оценивает его как истину, если он осуществляет любой вывод, или как ложный, если он не делает этого.

```
SELECT * FROM Таблица WHERE Таблица.Поле = Значение  
AND EXISTS (SELECT * FROM Таблица WHERE Таблица.Поле = Значение)
```

SOME и ANY определяет вывод в подзапросе

Эти операторы берут все значения, выведенные подзапросом и оценивает их как верные, если любое из них равняется условию запроса.

```
SELECT * FROM Таблица1 WHERE Поле = ANY  
(SELECT Поле1 FROM Таблица2)
```

ALL определяет вывод в подзапросе

Очень эффективно используется со значением неравенства <> возвращая все значения которые не равны условию.

```
SELECT * FROM Таблица WHERE Поле <> ALL  
(SELECT Поле FROM Таблица WHERE ПолеДата = 10/06/2012)
```

Работа с базой данных

CREATE - создает базу данных

DROP - удалить базу данных

ALTER - изменить определение базы данных.

Создать базу данных

```
CREATE DATABASE ИмяБазы ON Файл1, Файл2
```

Создать таблицу

```
CREATE TABLE ИмяТаблицы (Поле Тип Размер)
```

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL UNIQUE,  
Поле1 INTEGER,  
Поле2 CHAR (20),  
Поле3 CHAR (100),  
ПолеДата DATE)
```

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL PRIMARY KEY,  
Поле1 INTEGER,  
Поле2 CHAR (20),  
Поле3 CHAR (100),  
ПолеДата DATE)
```

Изменить таблицу

```
ALTER TABLE ИмяТаблицы ADD ИмяПоля Тип Размер
```

Удаление таблицы

```
DROP TABLE ИмяТаблицы
```

Создать индекс

```
CREATE INDEX ИмяИндекса ON ИмяТаблицы (ИмяПоля)
```

Создание уникального (не содержащего повторов значений) индексов

```
CREATE UNIQUE INDEX ИмяИндекса ON ИмяТаблицы (ИмяПоля)
```

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL UNIQUE,  
Поле1 INTEGER,  
Поле2 CHAR (20),  
Поле3 CHAR (100))
```

Установка первичного ключа

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL PRIMARY KEY,  
Поле1 INTEGER,  
Поле2 CHAR (20),  
Поле3 CHAR (100),  
ПолеДата DATE)
```

Оператор CHECK проверяет вводимые в таблицу значение до того, как оно будет сохранено.

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL PRIMARY KEY,  
Поле1 INTEGER CHECK (Поле1 <= 25)  
Поле2 INTEGER NOT NULL,  
Поле3 INTEGER NOT NULL,  
ПолеДата DATE,  
UNIQUE (Поле2, Поле3))
```

Установка значения по умолчанию

```
CREATE TABLE ИмяТаблицы  
(ПолеID INTEGER NOT NULL PRIMARY KEY,  
Поле1 INTEGER CHECK (Поле1 <= 25) DEFAULT = 5  
Поле2 INTEGER NOT NULL,  
Поле3 INTEGER NOT NULL,  
ПолеДата DATE,  
UNIQUE (Поле2, Поле3))
```

Синоним - это альтернативное имя для таблицы.

```
CREATE SYNONYM ИмяТаблицы FOR Пользователь.ИмяТаблицы
```


Локаторы XPATH и CSS

Язык запросов к элементам XML-документа

child::	элементы потомки
descendant::	полное множество элементов потомков
descendant-or-self::	полное множество потомков и текущий элемент
ancestor::	элементы предки
ancestor-or-self::	элементы предки и текущий элемент
parent::	элемент предок на один уровень назад
self::	текущий элемент
following::	элементы ниже текущего элемента
following-sibling::	братские элементы того же уровня
preceding::	элементы выше текущего уровня
preceding-sibling::	братские элементы предшествующие текущему
attribute::	атрибуты текущего элемента
namespace::	элементы относительно пространства имен
* - все элементы	
Функции над множествами узлов (n - это node-set; 0 - это object)	
node()	возвращает сам узел
text()	возвращает текст узла
current()	возвращает множество из текущего элемента

position()	возвращает позицию элемента в множестве элементов
last()	возвращает номер последнего элемента в множестве
count(n)	возвращает кол-во элементов в узле
name(n)	возвращает полное имя первого тега в множестве
namespace-url(n)	возвращает ссылку на URL
localname(n)	возвращает имя первого тега без пространства имен
id(o)	находит элемент с уникальным ID
Строковые функции	
string(obj)	возвращает текстовое содержимое элемента
concat(str, str, str*)	соединяет строки
length(str)	возвращает длину строки
contains(str, str)	возвращает true если первая строка содержит вторую
substring(str, num, num?)	возвращает вырезанную строку
substring-before(str, str)	если найдена вторая строка в первой возвращает строку до первого вхождения второй строки
substring-after(str, str)	если найдена вторая строка в первой возвращает строку после первого вхождения второй строки
starts-with(str, str)	возвращает true если вторая строка входит в начало первой (иначе false)
end-with(str, str)	возвращает true если вторая строка входит в конец первой (иначе false)
normalize-space(str)	удаляет лишние и повторные пробелы
translate(str, str, str)	заменяет символы в строке
Логические функции и операторы	
and	логическое И
or	логическое ИЛИ
=	логическое равно
<	логическое меньше
>	логическое больше
<=	логическое меньше или равно

>=	логическое больше или равно
boolean(obj)	приводит объект к логическому типу
true()	возвращает истину
false()	возвращает ложь
not(boolean)	отрицание, возвращает истину если аргумент ложь
Числовые функции и операторы	
+, -, *;	сложение, вычитание, умножение
div	деление
mod	остаток от деления
number(obj)	переводит объект в число
sum(node)	возвращает сумму множества
floor(number)	возвращает наибольшее целое число
ceiling(number)	возвращает наименьшее целое число
round(number)	округление
Системные функции	
document(obj, node)	возвращает документ
format-number(num,str,str)	формат числа
generate-id(node)	уникальный идентификатор
key(str, obj)	возвращает множество с указанным ключом
unparsed-entity-uri(str)	возвращает не проанализированный URL
element-available(str)	проверяет доступен ли элемент
function-available(str)	проверяет доступна ли функция
system-property(str)	возвращает системные параметры
lang(str)	возвращает true если у текущего тега есть
Прочие обозначения	
*	любое имя
@name	имя переменной или параметра (@id, @class)
[]	дополнительные условия выбора
{ }	если применяется внутри тега другого языка

/	определяет уровень дерева
	объединяет результат
Примеры	
<pre>//div[@id='example_id']//div/a //div[@class='example_class']//div/a //input[@id='user']/following-sibling::input[4] //input[@name='login' and @type='submit'] //*[contains(text(), 'login')] //a[contains(@href, 'google.com')] //label[contains(text(), 'интернет')]/parent::div</pre>	

CSS	XPATH
div > a	//div/a
div a	//div//a
#user_id	//div[@id='user_id']
.user_class	//div[@class='user_class']
#login + input	//div[@id='login']/following-sibling::input[1]
input[name='user']	//input[@name='user']
input[id='btn'][type='submit']	//input[@id='btn' and @type='submit']
Использование нескольких имен классов	
<div><div class="value test"></div></div> <div><div class="value test " ></div></div> <div><div class="first value test last"></div></div> <div><div class="test value"></div></div>	
div[class='value test '] div[class*='value test'] div.value.test	//div[@class='value test '] //div[contains(@class, 'value') and contains(@class, 'test')]
Дочерние элементы	
<div><ul id="list"></div> <div>value1</div> <div>value2/></div> <div></div>	
#list li:nth-of-type(4) #list li:nth-child(4)	

#list *:nth-child(4)	
Частичное совпадение строк	
a[id ^= 'id_prefix_'] a[id \$= '_id_sufix'] a[id *= 'id_pattern']	
Соответствие по внутреннему тексту	
a:contains('log out')	//a[contains(@text, 'log out')

Selenium установка и запуск

Необходимо скачать с официального сайта <https://www.selenium.dev/downloads/> приложения и в разделе загрузки скачать версию **3.141.59**

Selenium Server (Grid)

The Selenium Server is needed in order to run Remote Selenium WebDriver (Grid).

Latest stable version **3.141.59**

To use the Selenium Server in a Grid configuration see the [documentation](#).

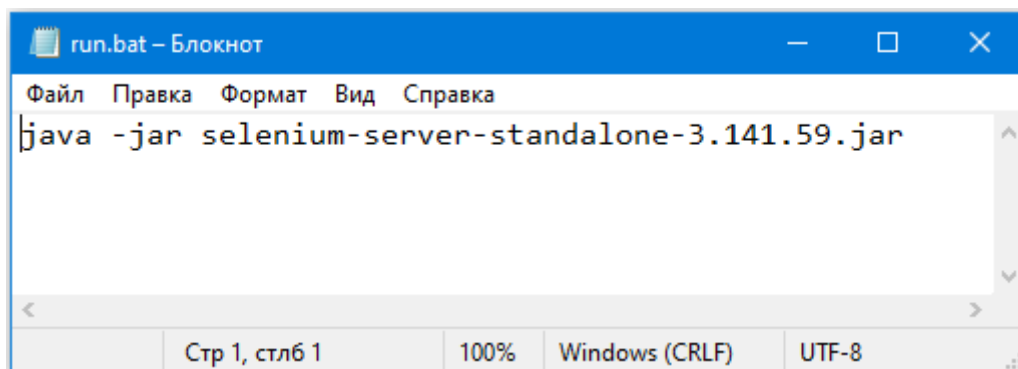
Latest Selenium 4 Beta version **4.0.0-beta-3**

Далее необходимо скачать драйвер для Chrome с сайта <https://chromedriver.chromium.org/> последней версии которая соответствует версии установленного браузера




All versions available in [Downloads](#)

- Latest stable release: [ChromeDriver 90.0.4430.24](#)
- Latest beta release: [ChromeDriver 91.0.4472.19](#)

Далее создаем файл run.bat в котором прописываем команду запуска
 java -jar selenium-server-standalone-3.141.59.jar



Сохраняем файлы в одну папку. Например C:\Users\...\Desktop\selenium

Имя	Дата изменения	Тип	Размер
 chromedriver.exe	16.04.2021 9:03	Приложение	11 046 КБ
 run.bat	11.05.2021 9:50	Пакетный файл ...	1 КБ
 selenium-server-standalone-3.141.59.jar	10.03.2020 10:41	Executable Jar File	10 401 КБ

Чтобы запустить Selenium дважды нажимаем на файл run.bat

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Catfish\Desktop\selenium>java -jar selenium-server-standalone-3.141.59.jar
10:25:33.274 INFO [GridLauncherV3.parse] - Selenium server version: 3.141.59, revision: e82be7d358
10:25:33.414 INFO [GridLauncherV3.lambda$buildLaunchers$3] - Launching a standalone Selenium Server on port 4444
2021-05-11 10:25:33.496:INFO::main: Logging initialized @1091ms to org.seleniumhq.jetty9.util.log.StdErrLog
10:25:33.783 INFO [WebDriverServlet.<init>] - Initialising WebDriverServlet
10:25:34.084 INFO [SeleniumServer.boot] - Selenium Server is up and running on port 4444
```

Чтобы остановить нужно нажать Ctrl+C и ввести символ Y

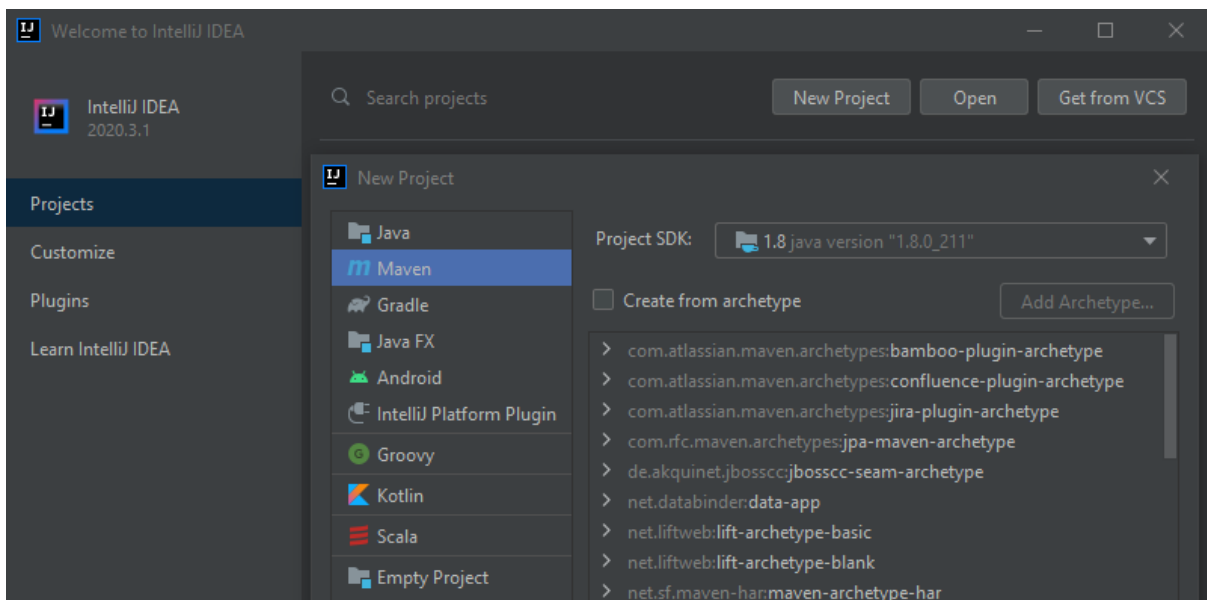
Web - приложение

Автоматизированное тестирование Web приложений

Стек: Java + Selenium + JUnit + TestNG

В качестве IDE используется IntelliJ IDEA

Создаем новый проект Maven с выбранным SDK 1.8 (JDK 8)



Указываем имя и путь для размещения проекта

New Project

Name:

Location:

▼ Artifact Coordinates

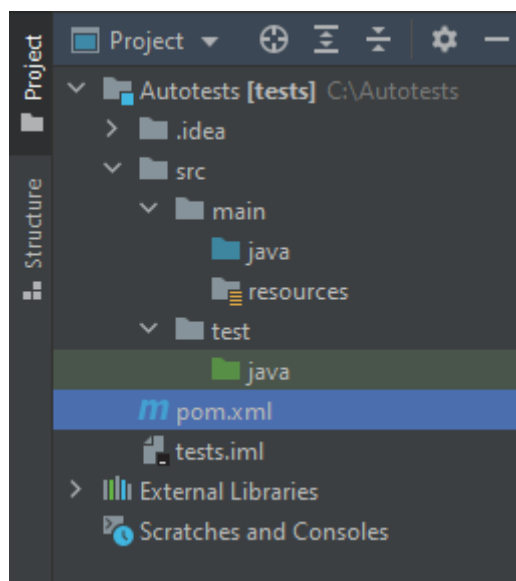
GroupId:
The name of the artifact group, usually a company domain

ArtifactId:
The name of the artifact within the group, usually a project name

Version:

Нажать кнопку Finish

Пустой шаблон проекта готов



Теперь необходимо подключить библиотеки прописав их в файле pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.tests</groupId>
  <artifactId>tests</artifactId>
  <version>1.0-SNAPSHOT</version>
```

```

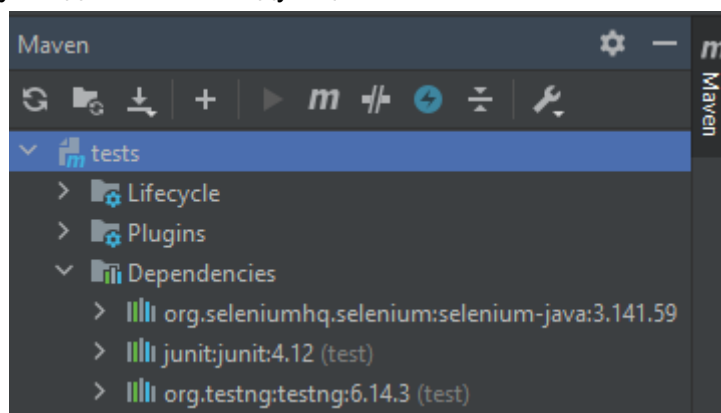
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java
-->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/junit/junit -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>

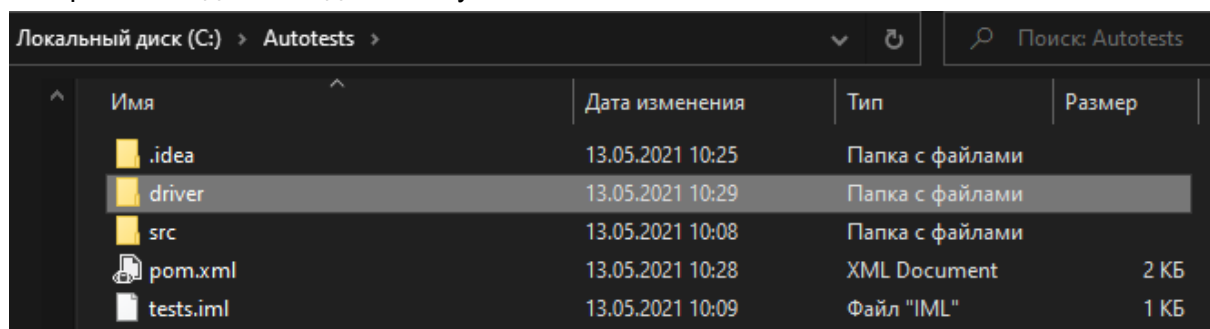
  <!-- https://mvnrepository.com/artifact/org.testng/testng -->
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.14.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>

```

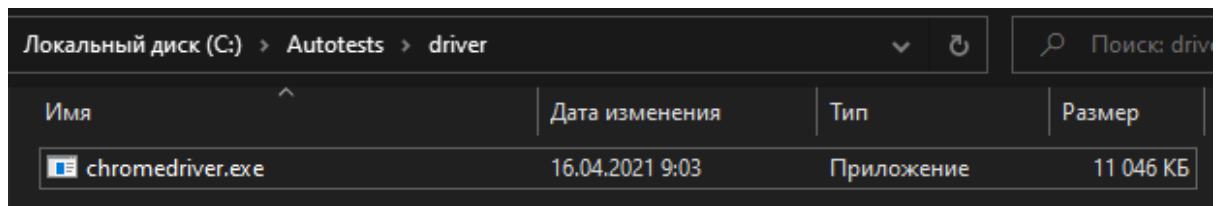
В результате будут подключены следующие библиотеки



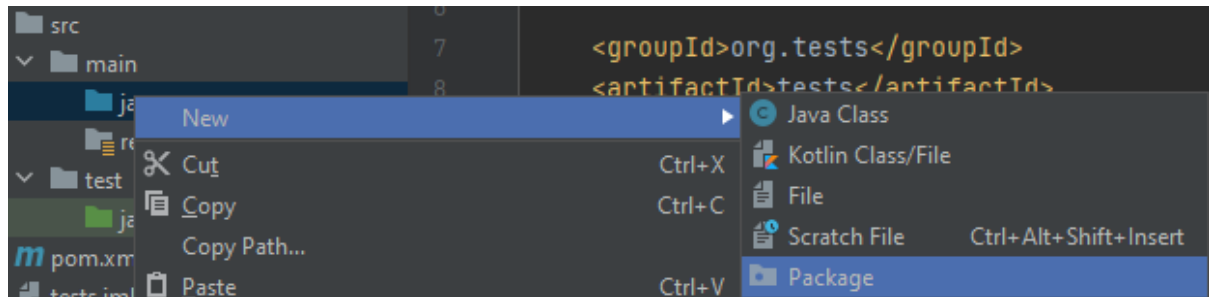
Теперь необходимо создать папку driver



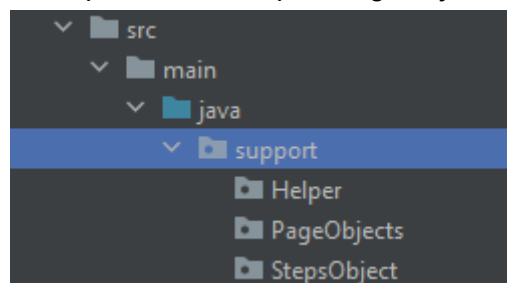
Скачайте webdriver для браузера Chrome и поместите в папку driver



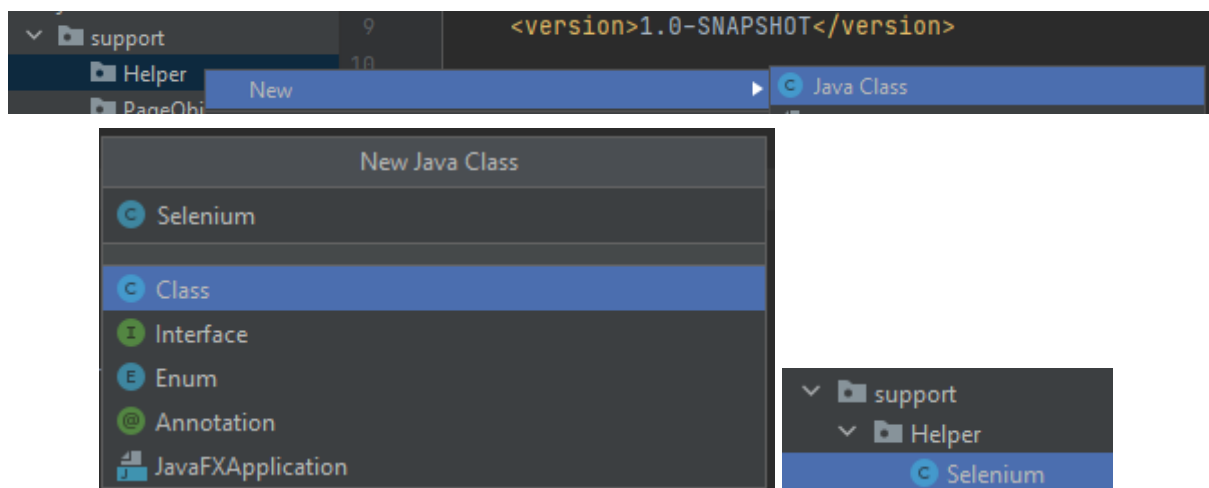
Создаем пакет support в папке main/java



В пакете support создать еще три пакета: Helper, PageObjects, StepsObject



В пакете Helper создайте класс Selenium



Описание файла Selenium.java

```
package support.Helper;

import org.openqa.selenium.*;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.CapabilityType;
```

```

import org.openqa.selenium.remote.DesiredCapabilities;

public class Selenium {
    public static WebDriver driver;

    public static String getDriverPath() {
        String path = System.getProperty("user.dir") +
            "\\driver\\chromedriver.exe";
        System.out.println("WebDriver: path [" + path + "]);
        return path;
    }

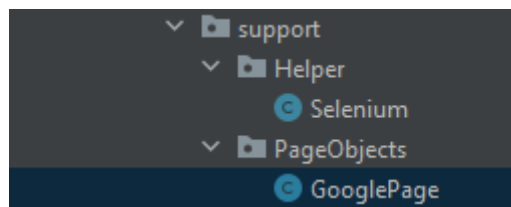
    public static void initWebDriver() {
        System.setProperty("webdriver.chrome.driver", getDriverPath());
        driver = new ChromeDriver();
        System.out.println("WebDriver: init");
    }

    public static void browserFullScreen() {
        driver.manage().window().maximize();
        System.out.println("Browser: full screen");
    }

    public static void quitWebDriver() {
        driver.close();
        driver.quit();
        System.out.println("WebDriver: quit");
    }
}

```

В пакете PageObjects создайте класс GooglePage



Описание файла GooglePage.java

```

package support.PageObjects;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

import java.util.List;

public class GooglePage {
    public static String inputSearchName = "q";
    public static String searchResultsClass = "g";

    public static WebElement getInputSearch(WebDriver driver)
    {

```

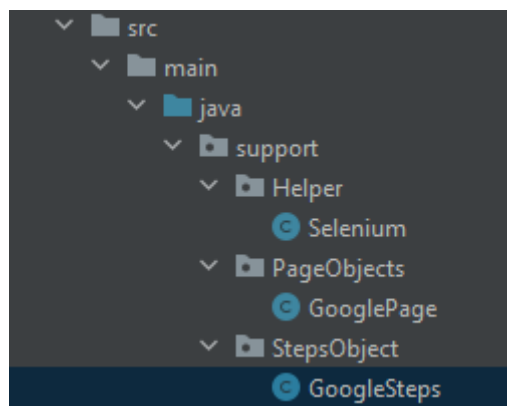
```

        By inputSearch = By.name(inputSearchName);
        WebElement element = driver.findElement(inputSearch);
        return element;
    }

    public static List<WebElement> getListResultsSearch(WebDriver driver)
    {
        By searchResult = By.className(searchResultsClass);
        List<WebElement> elements = driver.findElements(searchResult);
        return elements;
    }
}

```

В пакете StepsObject создайте класс GoogleSteps



Описание файла GoogleSteps.java

```

package support.StepObjects;

import support.PageObjects.GooglePage;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

import java.util.List;

public class GoogleSteps {
    public final WebDriver driver;

    public GoogleSteps(WebDriver webdriver) {
        driver = webdriver;
    }

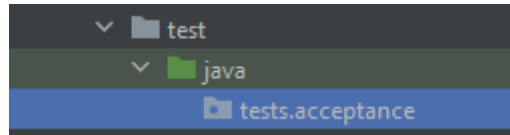
    public void setValueInSearch(String value) {
        WebElement inputSearch = GooglePage.getInputSearch(driver);
        inputSearch.sendKeys(value);
        inputSearch.sendKeys(Keys.ENTER);
    }

    public int getCountResultSearch()
    {
        List<WebElement> resultElements =

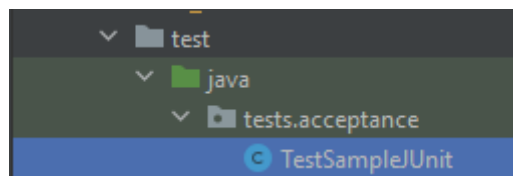
```

```
        GooglePage.getListResultsSearch(driver);  
        return resultElements.size();  
    }  
}
```

Создаем пакет tests и в нем пакет acceptance в папке test/java



В пакете tests.acceptance создайте класс автотеста TestSampleJUnit



Описание автотеста на основе JUnit в файле TestSampleJUnit.java

```
package tests.acceptance;  
  
import org.junit.AfterClass;  
import org.junit.BeforeClass;  
import org.junit.Test;  
import org.junit.Assert;  
import support.Helper.Selenium;  
import support.PageObjects.GooglePage;  
import support.StepsObject.GoogleSteps;  
  
public class TestSampleJUnit {  
    @BeforeClass  
    public static void setup() {  
        Selenium.initWebDriver();  
    }  
}
```

```

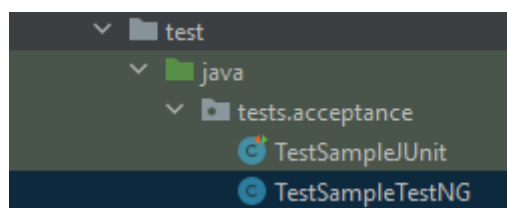
        Selenium.browserFullScreen();
    }

    @Test
    public void testSearch() {
        GoogleSteps tester = new GoogleSteps(Selenium.driver);
        tester.driver.get("https://www.google.com/");
        tester.setValueInSearch("GeForce 1650");
        int result = tester.getCountResultSearch();
        Assert.assertNotEquals(0, result);
        System.out.println("Tests finished: SUCCESS");
    }

    @AfterClass
    public static void tearDown() {
        Selenium.quitWebDriver();
    }
}

```

В пакете tests.acceptance создайте класс автотеста TestSampleTestNG



Описание автотеста на основе TestNG в файле TestSampleTestNG.java

```

package tests.acceptance;

import org.testng.Assert;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import org.testng.annotations.AfterTest;
import support.Helper.Selenium;
import support.PageObjects.GooglePage;
import support.StepsObject.GoogleSteps;

public class TestSampleTestNG {

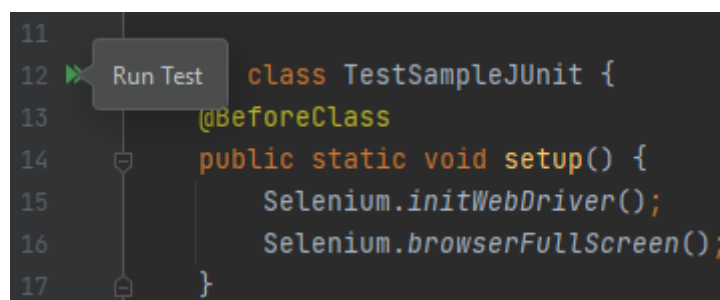
```

```
@BeforeTest
public static void setup() {
    Selenium.initWebDriver();
    Selenium.browserFullScreen();
}

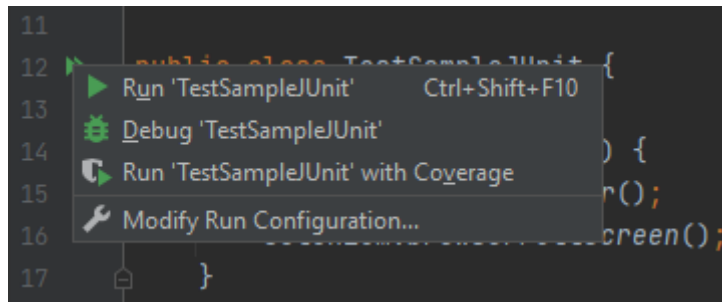
@Test
public void testSearch() {
    GoogleSteps tester = new GoogleSteps(Selenium.driver);
    tester.driver.get("https://www.google.com/");
    tester.setValueInSearch("GeForce 1650");
    int result = tester.getCountResultSearch();
    Assert.assertNotEquals(0, result);
    System.out.println("Tests finished: SUCCESS");
}

@AfterTest
public static void tearDown() {
    Selenium.quitWebDriver();
}
}
```

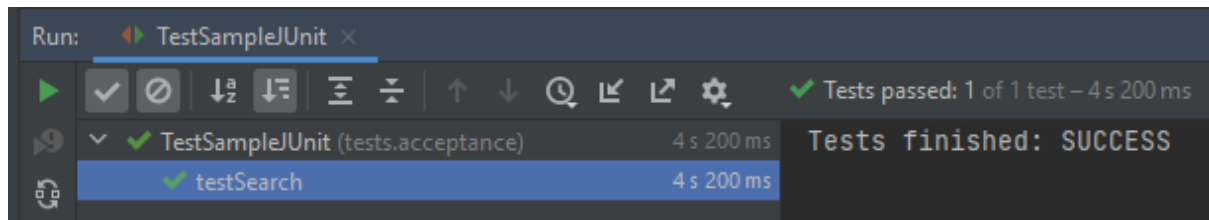
Чтобы запустить автотест в рамках IntelliJ IDEA нужно нажать на зеленую стрелку напротив класса.



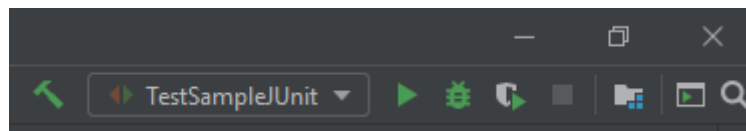
И нажать на пункт Run



Результат запуска в консоли

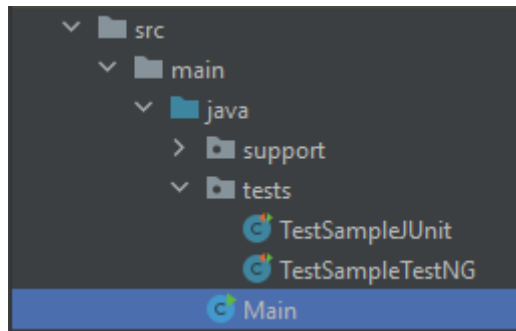


После запуска автотеста на панели инструментов будет сформирована команда запуска которой можно пользоваться вместо запуска из класса.



Запуск автотестов из основного класса Main

Для этого нужно сначала перенести автотесты в папку main и создать класс Main



Теперь нужно изменить scope библиотек в файле pom.xml на значение compile

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>compile</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.14.3</version>
    <scope>compile</scope>
</dependency>
```

Описание файла Main.java для запуска JUnit автотестов

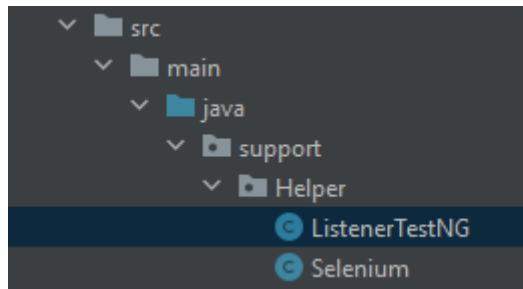
```
import org.junit.internal.TextListener;
import org.junit.runner.JUnitCore;
import tests.TestSampleJUnit;

public class Main {

    public static void main(String[] args){
        Result result = JUnitCore.runClasses(TestSampleJUnit.class);

        JUnitCore junit = new JUnitCore();
        junit.addListener(new TextListener(System.out));
        junit.run(TestSampleJUnit.class);
    }
}
```

Для запуска TestNG автотестов необходимо создать вспомогательный класс ListenerTestNG в пакете support.



Описание файла ListenerTestNG.java

```
package support.Helper;

import org.testng.ISuite;
import org.testng.ISuiteListener;

public class ListenerTestNG implements ISuiteListener {
    public void onStart(ISuite suite){
        System.out.println("directory = " + suite.getOutputDirectory());
    }

    public void onFinish(ISuite suite) {
        System.out.println("methods = " + suite.getAllMethods());
        System.out.println("results = " + suite.getResults());
    }
}
```

Описание файла Main.java для запуска TestNG автотестов

```
import org.testng.TestNG;
import support.Helper.ListenerTestNG;
import tests.TestSampleTestNG;

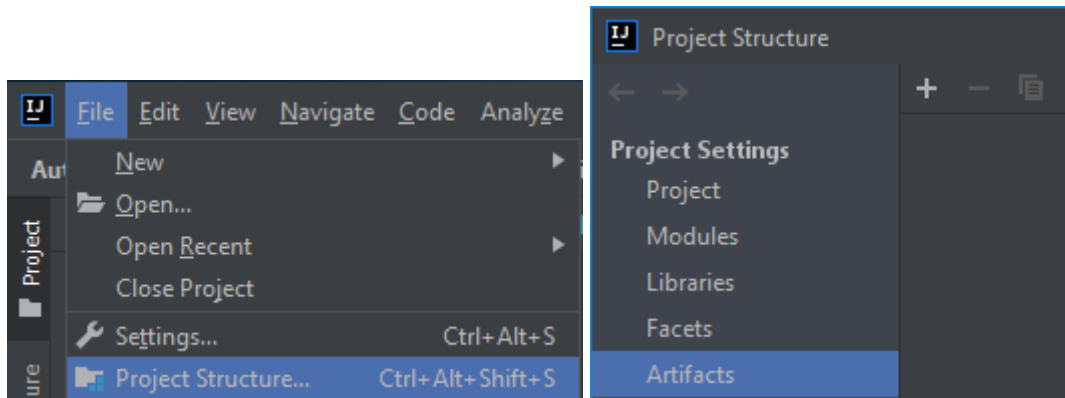
public class Main {

    public static void main(String[] args){

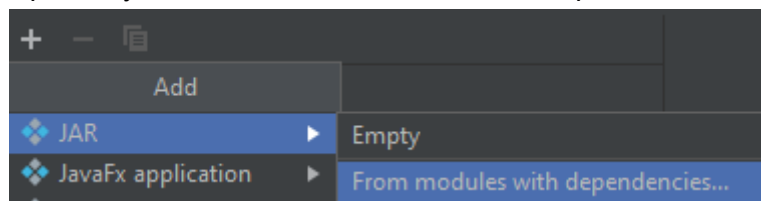
        TestNG test = new TestNG();
        test.setTestClasses(new Class[]{
            TestSampleTestNG.class
        });
        test.addListener(new ListenerTestNG());
        test.setDefaultSuiteName("TestSampleTestNG");
        test.setOutputDirectory("/out");
        test.run();
    }
}
```

Сборка артефакта для запуска автотестов

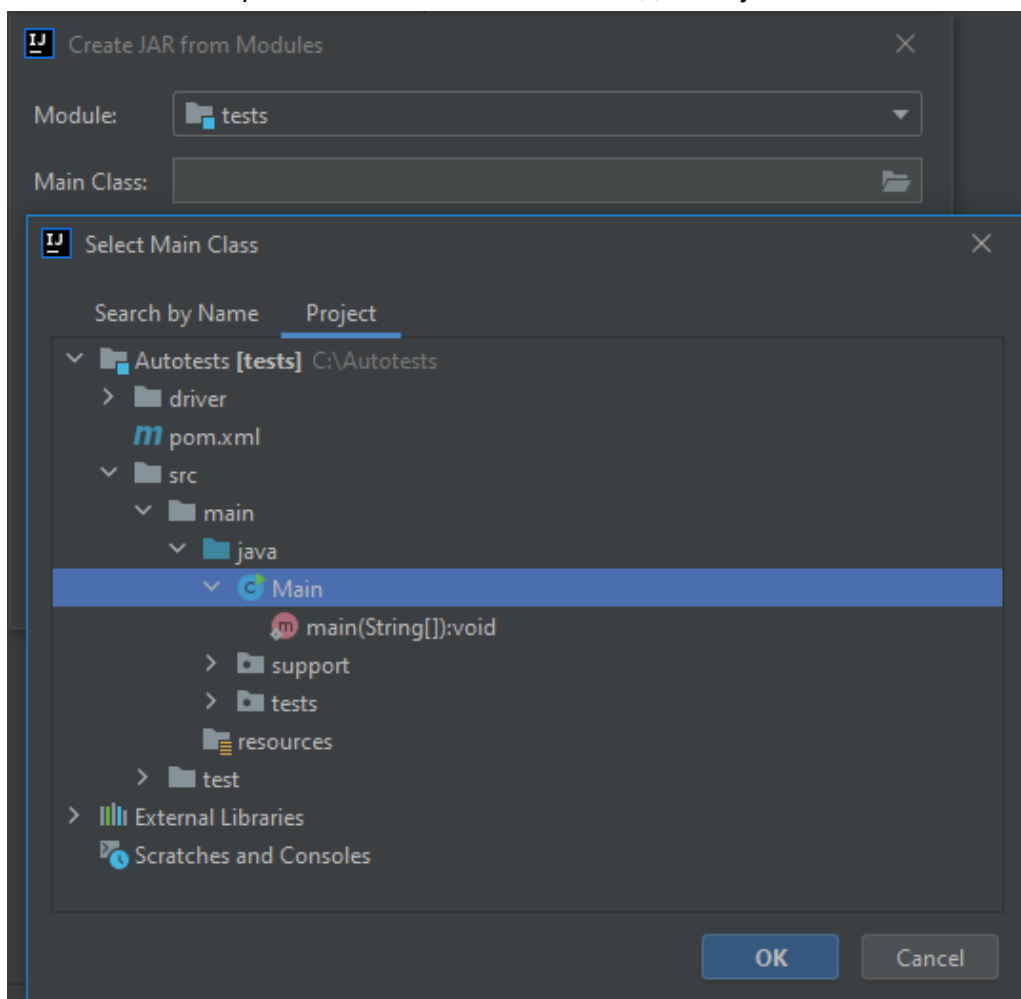
Для этого перейдем в настройки Project Structure и в разделе Artifacts добавим новую сборку нажав на кнопку плюс.



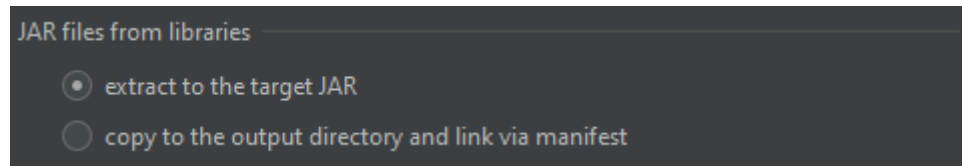
В меню Add выбираем пункт JAR -> From modules with dependencies...



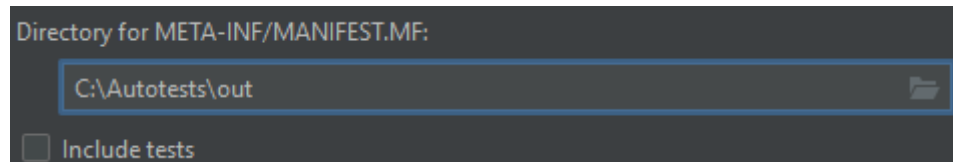
В поле Main Class выбираем наш Main класс на вкладке Project



Включаем добавление библиотек в артефакт (extract to target JAR)



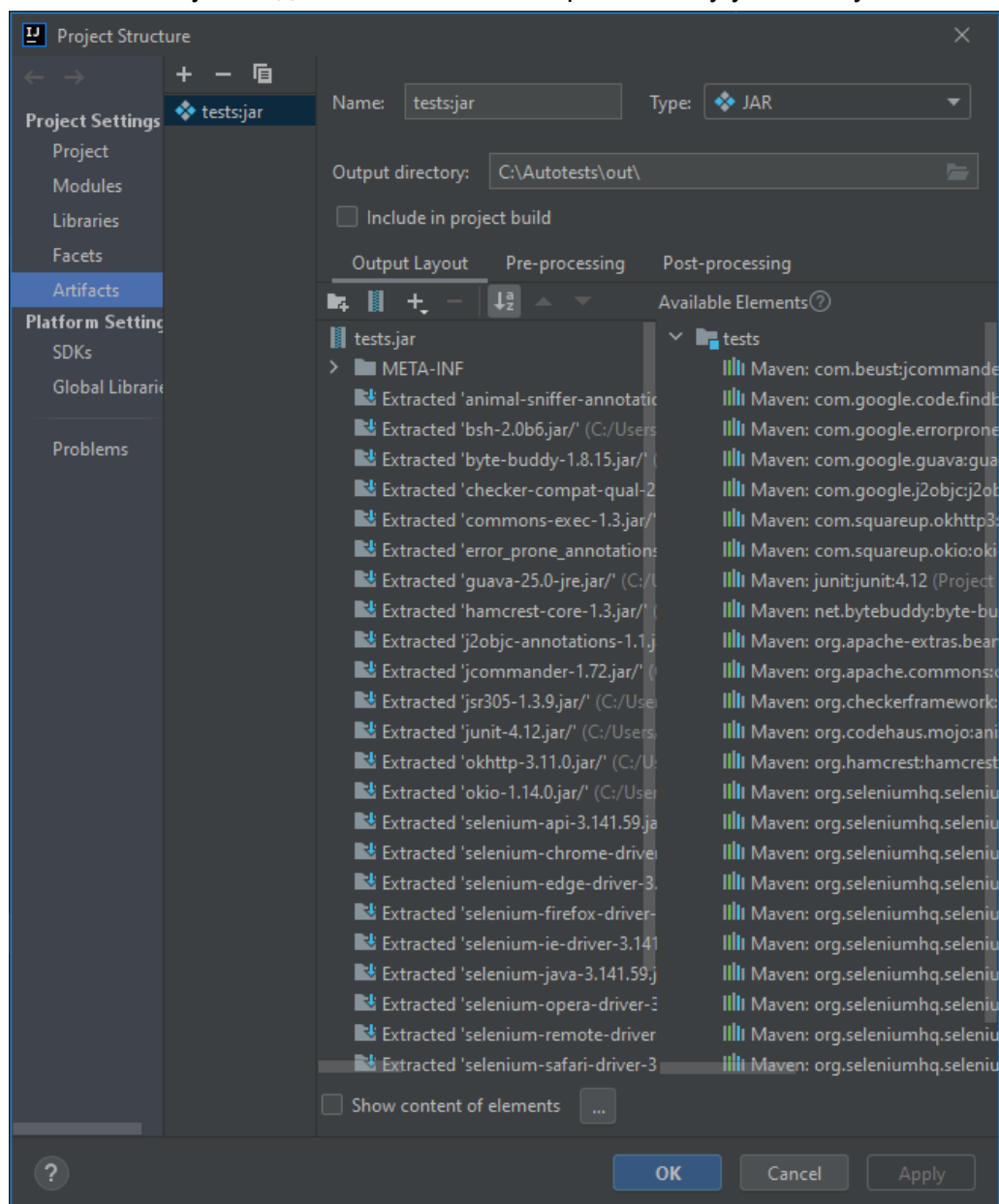
Указываем путь где должен храниться файл манифест (в папку out)



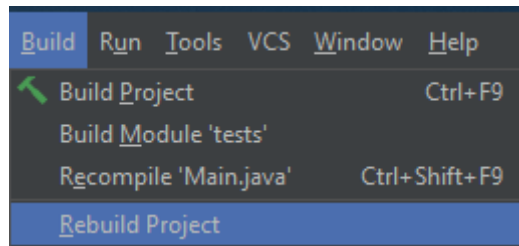
Нажимаем ОК

В результате будет создана настройка для сборки артефакта

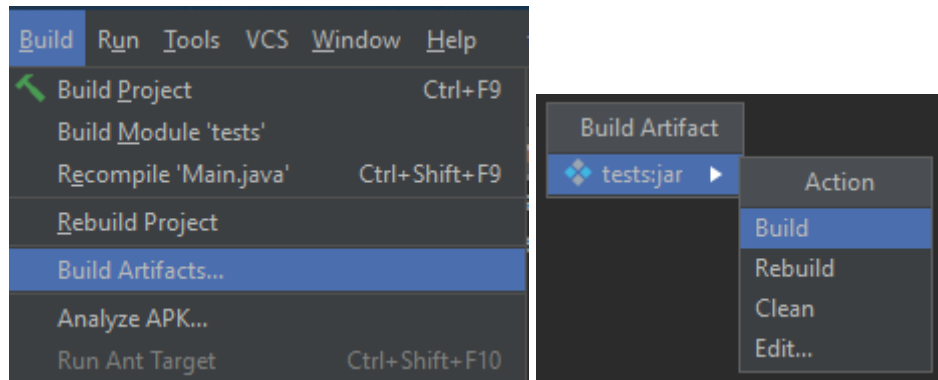
(единственное что нужно сделать это в поле Output directory указать путь к папке out)



Выполним сборку всего проекта

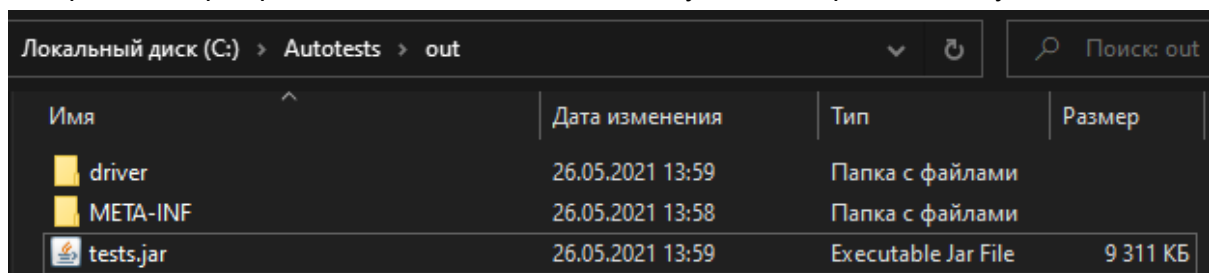


Выполним сборку артефакта

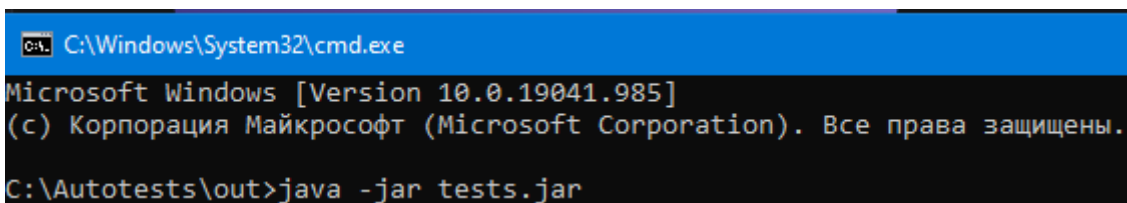


(если нужно пересобрать артефакт - тогда выполняйте команду Rebuild)

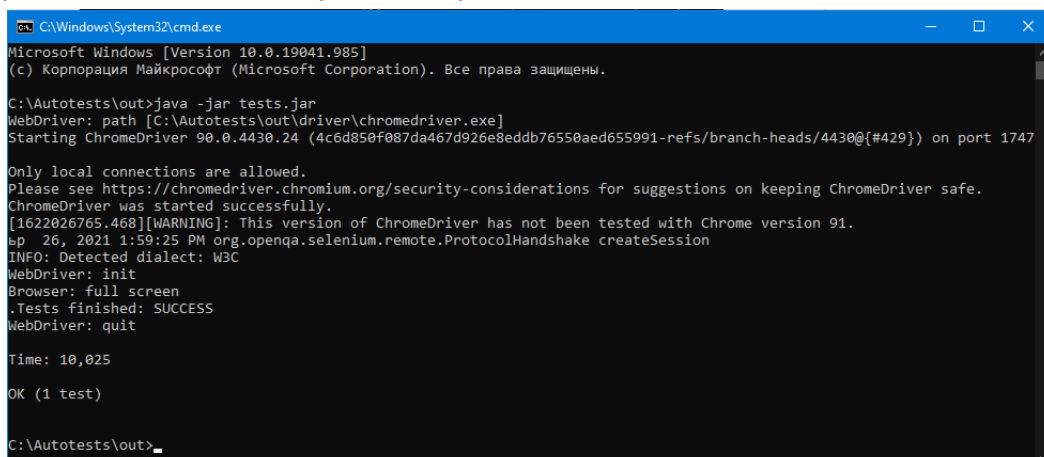
Теперь когда артефакт создан необходимо в папку out скопировать папку driver



Запустим артефакт tests.jar в консоли выполнив команду: `java -jar tests.jar`



В результате работы мы получим следующий отчет



Стек: Java + Selenide + TestNG

В качестве IDE используется IntelliJ IDEA

Создаем новый проект Maven с выбранным SDK 1.8 (JDK 8)

Указываем имя и путь для размещения проекта:

Artefact Coordinates:

Name: Tests

GroupId: org.tests

Location: C:\Autotests

ArtifactId: Tests

Нажать кнопку Finish

Теперь необходимо подключить библиотеки прописав их в файле pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

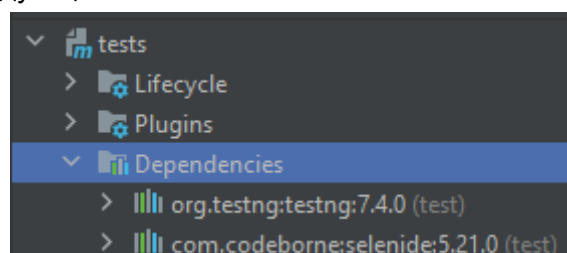
  <groupId>org.tests</groupId>
  <artifactId>tests</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
  </properties>

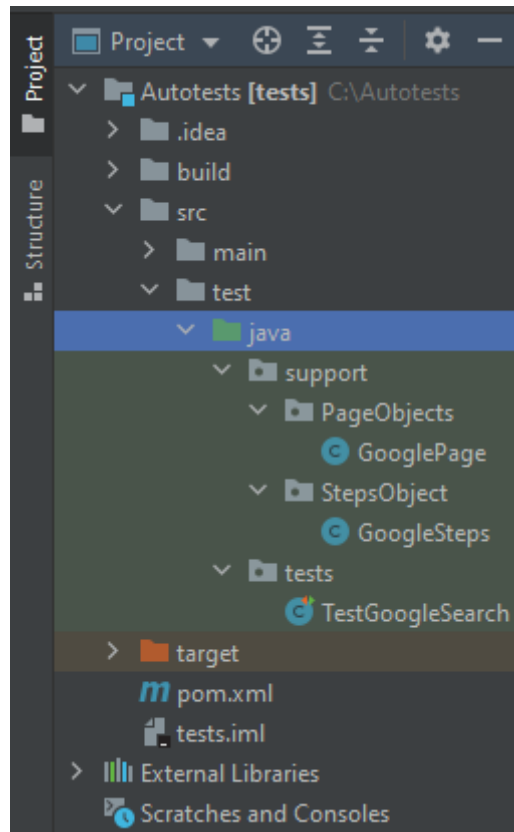
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.4.0</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/com.codeborne/selenide -->
    <dependency>
      <groupId>com.codeborne</groupId>
      <artifactId>selenide</artifactId>
      <version>5.21.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Были подключены следующие библиотеки Selenide и TestNG



Проект имеет следующую структуру:



Описание класса GooglePage

```
package support.PageObjects;

import static com.codeborne.selenide.Selenide.*;
import static com.codeborne.selenide.Condition.*;
import static com.codeborne.selenide.Selectors.*;
import static com.codeborne.selenide.CollectionCondition.*;
import com.codeborne.selenide.SelenideElement;
import java.util.List;

public class GooglePage {
    public static String inputSearchName = "q";
    public static String searchResultsClass = "g";

    public static SelenideElement getInputSearch() {
        SelenideElement inputElement = element(byName(inputSearchName));
        return inputElement;
    }

    public static List<SelenideElement> getListResultsSearch() {
        List<SelenideElement> list =
            elements(byClassName(searchResultsClass));
        return list;
    }
}
```

Описание класса GoogleSteps

```
package support.StepsObject;

import com.codeborne.selenide.SelenideElement;
import java.util.List;
import support.PageObjects.GooglePage;

public class GoogleSteps {
    public void searchValue(String value) {
        SelenideElement inputElement = GooglePage.getInputSearch();
        inputElement.setValue(value).pressEnter();
    }

    public int getCountResultSearch() {
        List<SelenideElement> list = GooglePage.getListResultsSearch();
        return list.size();
    }
}
```

Описание автотеста, класс TestGoogleSearch

```
package tests;

import static com.codeborne.selenide.Selenide.*;
import static com.codeborne.selenide.Condition.*;
import static com.codeborne.selenide.Selectors.*;
import static com.codeborne.selenide.CollectionCondition.*;
import static org.openqa.selenium.support.ui.ExpectedConditions.titleIs;

import com.codeborne.selenide.Configuration;
import com.codeborne.selenide.ElementsCollection;
import com.codeborne.selenide.SelenideElement;
import org.apache.hc.core5.util.Asserts;

import org.testng.Assert;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import org.testng.annotations.AfterTest;

import support.StepsObject.GoogleSteps;

public class TestGoogleSearch {
    @BeforeTest
    public static void setup() {
        System.out.println("STEP: Setup");
    }

    @Test
    public void test1() {
        System.out.println("STEP: Test 1");
        Configuration.startMaximized = true;
        open("https://www.google.com/");
        element(byName("q")).setValue("GeForce 1650").pressEnter();
        //ElementsCollection list = elements("#search .g");
        elements("#search .g").shouldHave(sizeGreaterThanOrEqualTo(6))
            .first().shouldHave(text(
                "Игровая видеокарта GeForce GTX 1650 | NVIDIA"))
            .find("#rso > div:nth-child(1) > div > div > div.yuRUBf > a")
            .click();
    }
}
```

```

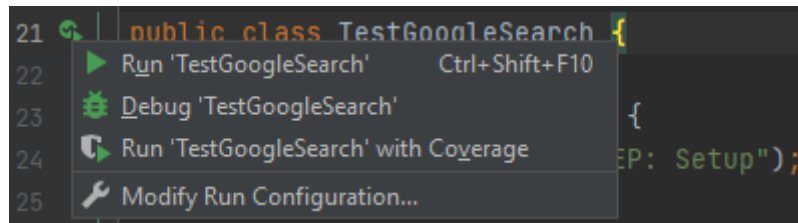
        Wait().until(titleIs(
            "Игровая видеокарта GeForce GTX 1650 | NVIDIA"));
    }

    @Test
    public void test2() {
        System.out.println("STEP: Test 2");
        open("https://www.google.com/");
        GoogleSteps tester = new GoogleSteps();
        tester.searchValue("Radeon RX 5500");
        int amount = tester.getCountResultSearch();
        Asserts.check((amount > 0), "Error: amount is 0");
    }

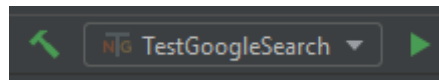
    @AfterTest
    public static void tearDown() {
        System.out.println("STEP: tear down");
    }
}

```

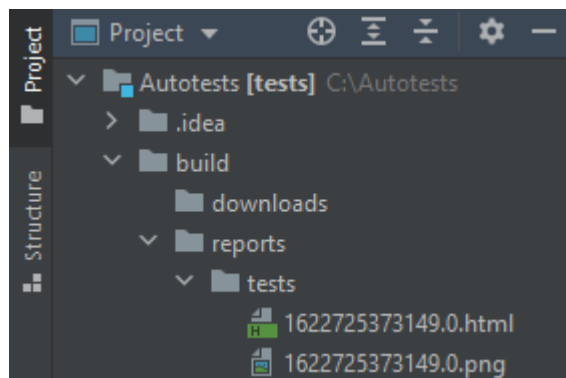
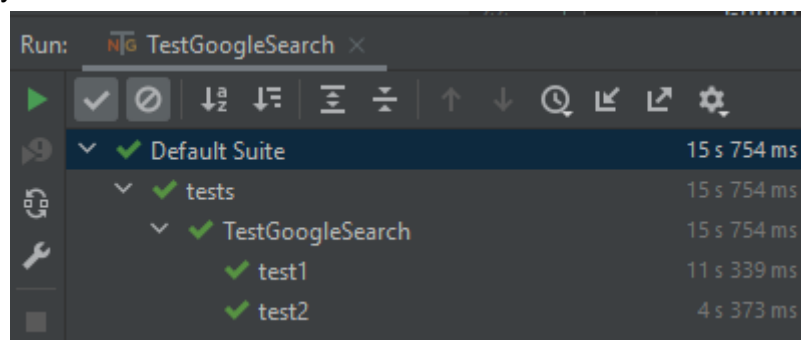
Запускаем автотест из под скрипта



или через панель быстрого доступа



Получаем результат выполнения



Отчет о провальных тестах в папке build

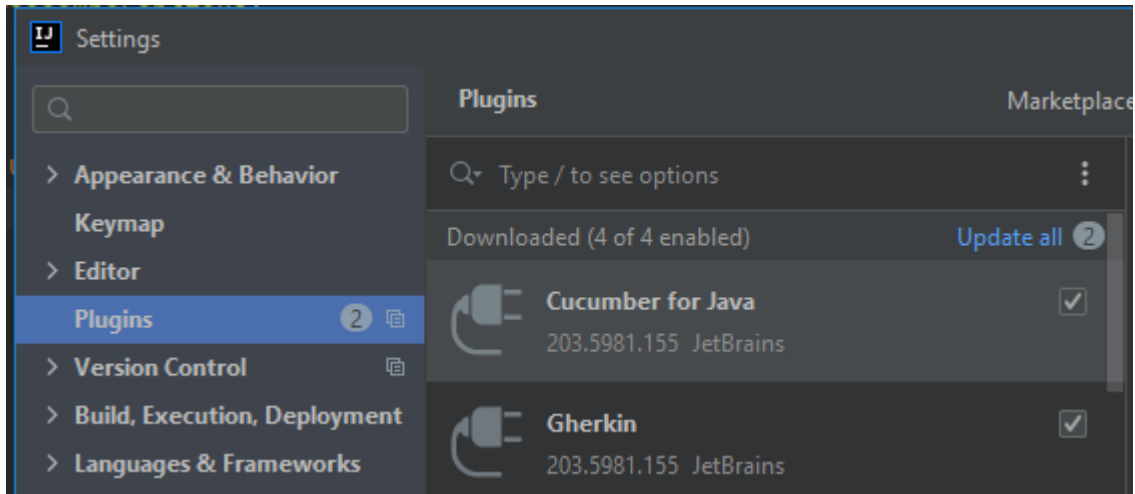
Стек: Java + Selenide + Cucumber + TestNG

В качестве IDE используется IntelliJ IDEA

К [предыдущему](#) проекту подключим Cucumber и используем его.

Необходимо установить плагин Cucumber for Java и Gherkin

Для этого нужно зайти в меню File -> Settings - Plugins и через поиск найти плагин.



Теперь необходимо добавить подключение библиотеки прописав в файле pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.tests</groupId>
  <artifactId>tests</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.4.0</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/com.codeborne/selenide -->
    <dependency>
      <groupId>com.codeborne</groupId>
      <artifactId>selenide</artifactId>
```

```

        <version>5.21.0</version>
        <scope>test</scope>
    </dependency>

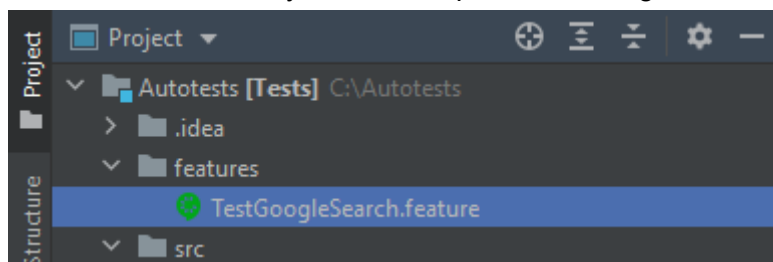
    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>6.10.4</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-core</artifactId>
        <version>6.10.4</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-testng</artifactId>
        <version>6.10.4</version>
    </dependency>
</dependencies>
</project>

```

В проекта необходимо создать папку features и файл TestGoogleSearch.feature



Описание файла TestGoogleSearch.feature

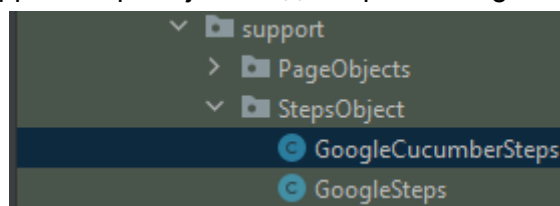
```

Feature: Test Google service

  Scenario: Value search
    Given Launch Chrome browser
    When Open Google page
    Then Input value in search field
    Then Check search result
    And Close browser

```

В папке src/test/java/support/StepsObject создать файл GoogleCucumberSteps.java



Описание файла GoogleCucumberSteps.java

```
package support.StepsObject;

import io.cucumber.java.en.*;
import static com.codeborne.selenide.Selenide.*;
import static com.codeborne.selenide.Condition.*;
import static com.codeborne.selenide.Selectors.*;
import static com.codeborne.selenide.CollectionCondition.*;
import static org.openqa.selenium.support.ui.ExpectedConditions.titleIs;
import com.codeborne.selenide.Configuration;
import com.codeborne.selenide.ElementsCollection;
import com.codeborne.selenide.SelenideElement;
import support.PageObjects.GooglePage;

public class GoogleCucumberSteps {
    @Given("Launch Chrome browser")
    public void launch_chrome_browser() {
        Configuration.startMaximized = true;
    }

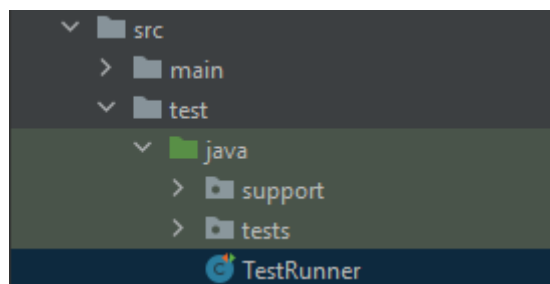
    @When("Open Google page")
    public void open_google_page() {
        open("https://www.google.com/");
    }

    @Then("Input value in search field")
    public void input_value_in_search_field() {
        element(byName(GooglePage.inputSearchName))
            .setValue("GeForce 1650").pressEnter();
    }

    @Then("Check search result")
    public void check_search_result() {
        elements("#search .g").shouldHave(sizeGreaterThanOrEqual(6))
            .first()
            .shouldHave(text("Игровая видеокарта GeForce GTX 1650 | NVIDIA"))
            .find("#rso > div:nth-child(1) > div > div > div.yuRUbf > a")
            .click();
        Wait().until(titleIs("Игровая видеокарта GeForce GTX 1650 | NVIDIA"));
    }

    @And("Close browser")
    public void close_browser() {
        closeWindow();
        closeWebDriver();
    }
}
```

В папке src/test/java нужно создать класс TestRunner

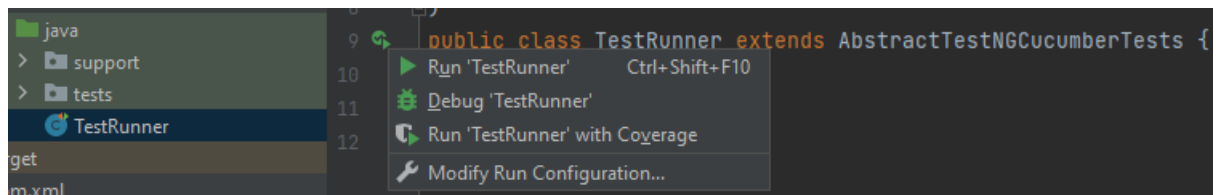


Описание файла TestRunner.java

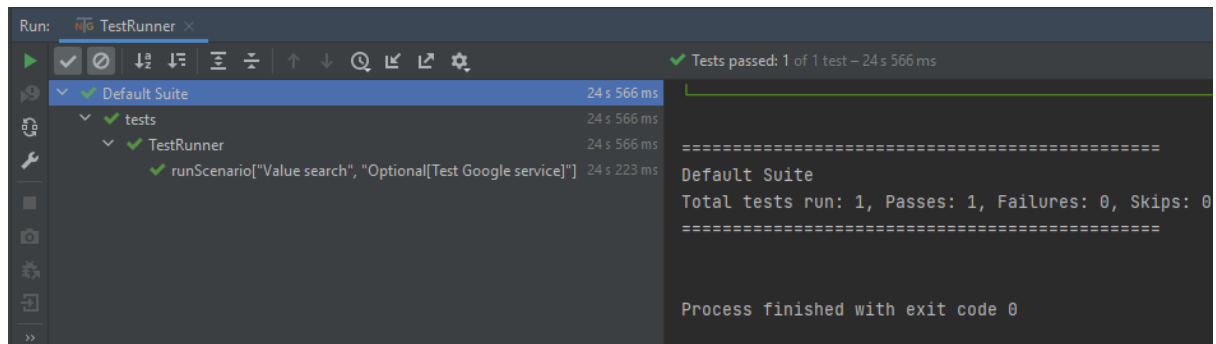
```
import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;

@CucumberOptions
(
    features = "features/TestGoogleSearch.feature",
    glue = "support/StepsObject"
)
public class TestRunner extends AbstractTestNGCucumberTests {
}
```

Запускаем класс TestRunner чтобы выполнить автотест



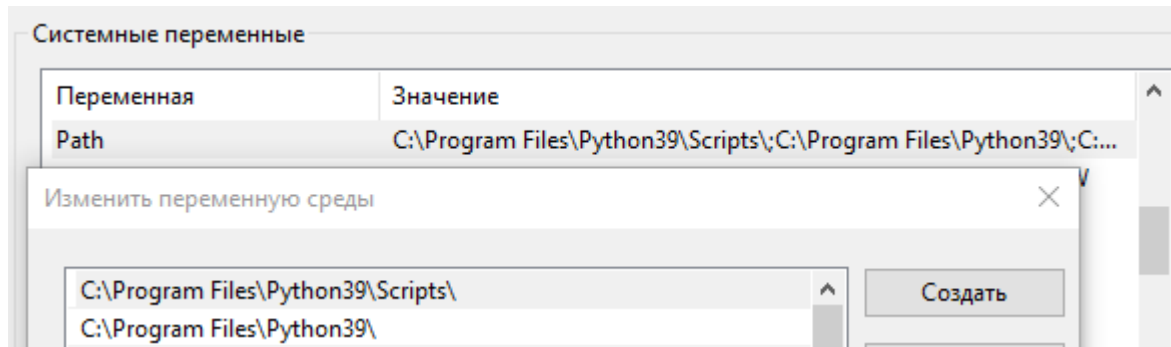
Получаем результат выполнения



Стек: Python + Selenium + Unittest

Необходимо установить Python (текущая версия 3.9.6)

Прописать пути в системной переменной



Далее в командной строке (cmd) выполнить:

- проверить установленные библиотеки командой: **pip list**
- обновить pip командой: **python -m pip install --upgrade pip**
- установить библиотеку Selenium командой: **pip install -U selenium**

```
cmd. Командная строка
Microsoft Windows [Version 10.0.19041.1052]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip list
Package      Version
-----
pip          21.1.3
setuptools   56.0.0

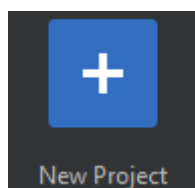
C:\Users\Catfish>python -m pip install --upgrade pip
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in c:\program files\python39\lib\site-packages (21.1.3)

C:\Users\Catfish>pip install -U selenium
Defaulting to user installation because normal site-packages is not writeable
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Collecting urllib3
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
  |████████████████████| 138 kB 1.3 MB/s
Installing collected packages: urllib3, selenium
Successfully installed selenium-3.141.0 urllib3-1.26.6

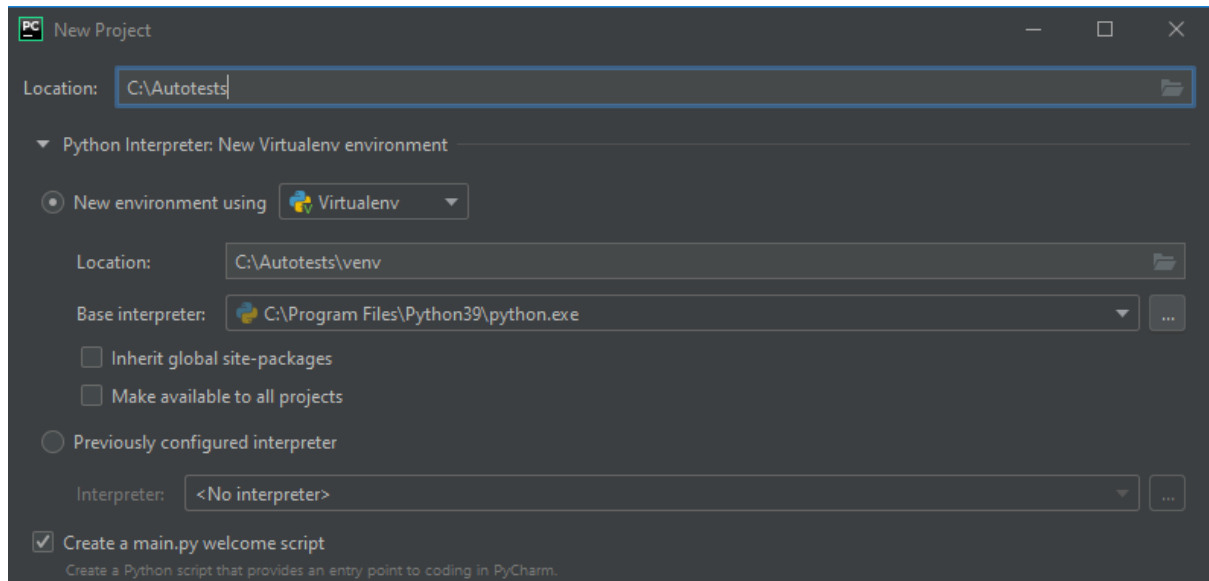
C:\Users\Catfish>
```

В качестве IDE используется PyCharm (текущая версия 2021.2)

Создадим проект.

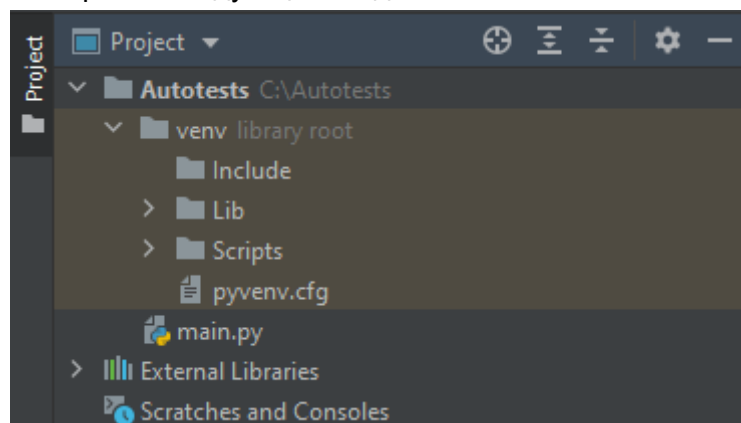


В поле Location указать адрес к папке C:\Autotests



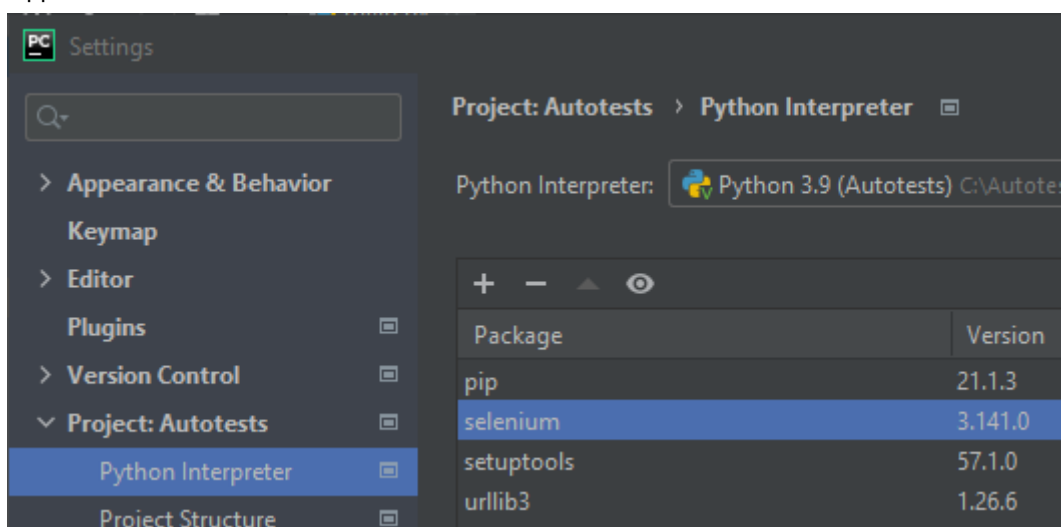
Нажать кнопку Create.

Будет создан пустой проект следующего вида:

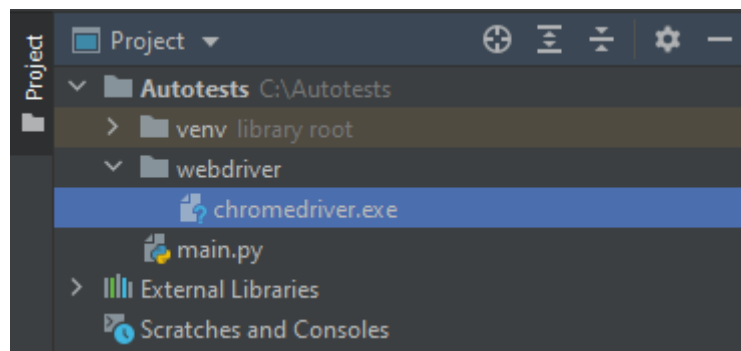


Настройки: меню File -> Settings -> вкладка Project -> Project Interpreter

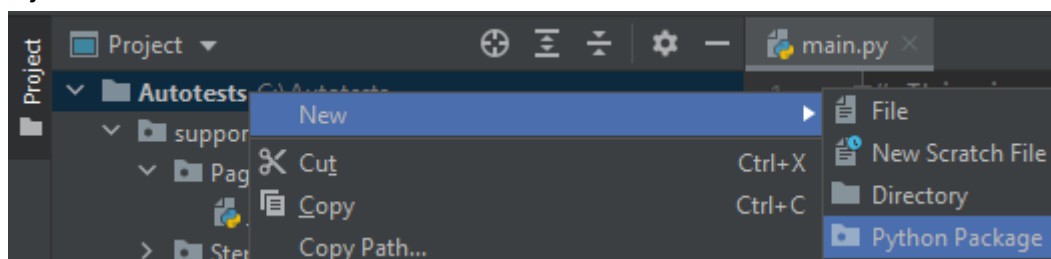
- выбрать Project Interpreter: Python 3.9
- добавить пакет selenium



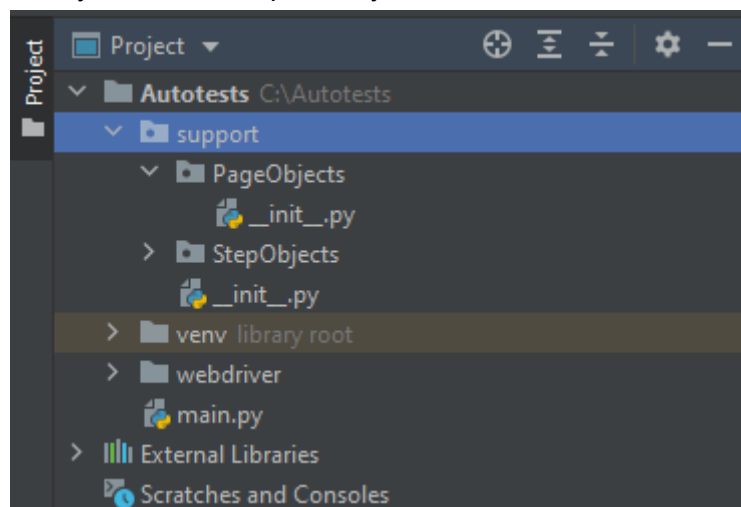
Создадим папку webdriver в корневой папке каталога и поместим в неё программу chromedriver.exe



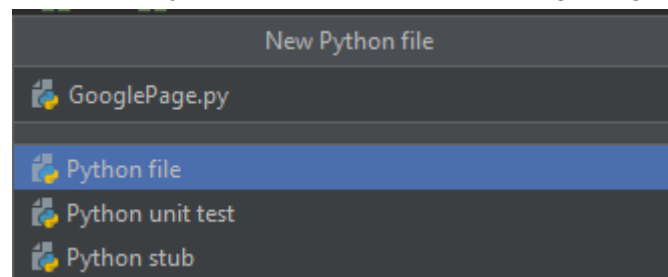
Необходимо создать пакет support и вложенные в него пакеты PageObjects и StepObjects



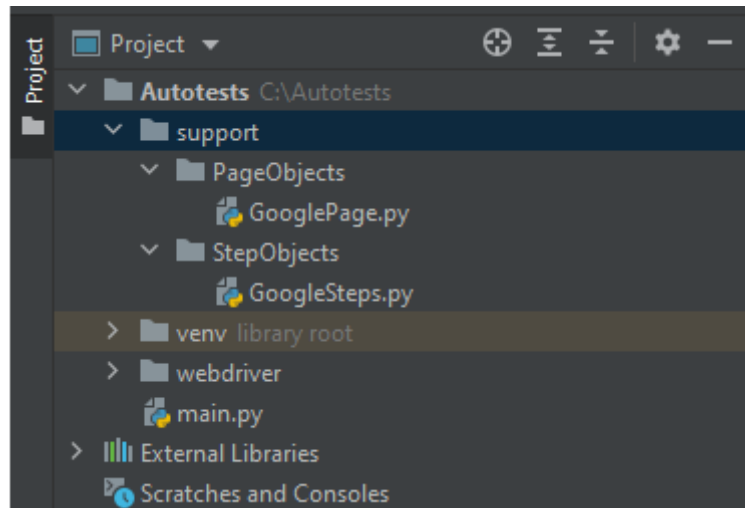
Когда пакет support будет создан, проект будет выглядеть так:



В пакете PageObjects и StepObjects создадим скрипты GooglePage.py и GoogleSteps.py



Когда файлы будут созданы, проект будет выглядеть так:



Описание файла GooglePage.py

```
class GooglePage:
    inputSearchName = "q"
    searchResultsClass = "g"

    def getInputSearch(self):
        inputSearch =
            self.driver.find_element_by_name(GooglePage.inputSearchName)
        return inputSearch

    def getListResultsSearch(self):
        searchResult =
            self.driver
                .find_elements_by_class_name(GooglePage.searchResultsClass)
        return searchResult
```

Описание файла GoogleSteps.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from support.PageObjects.GooglePage import GooglePage

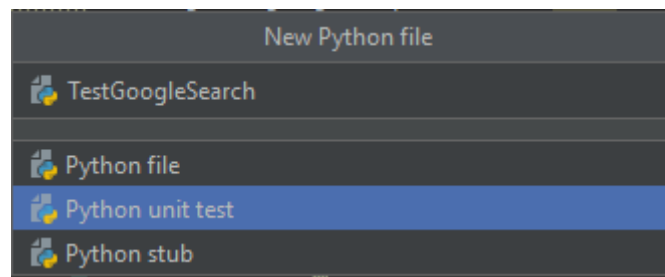
class GoogleSteps:
    driver = 0

    def __init__(self, webdriver):
        self.driver = webdriver

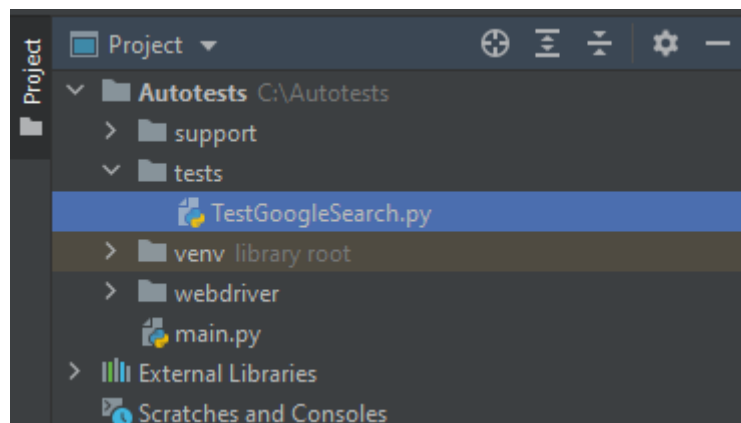
    def setValueInSearch(self, value):
        searchField = GooglePage.getInputSearch(self)
        searchField.send_keys(value)
        searchField.send_keys(Keys.ENTER)

    def getCountResultSearch(self):
        resultElements = GooglePage.getListResultsSearch(self)
        return len(resultElements)
```


Теперь создадим пакет tests и в нем модульный тест TestGoogleSearch.py



Когда пакет и файл будут созданы, проект будет выглядеть так:



Описание файла TestGoogleSearch.py

```
import unittest

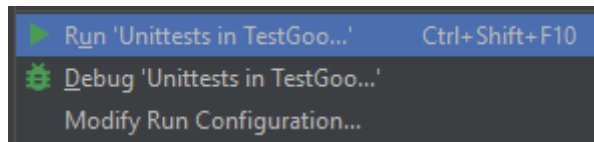
from selenium import webdriver
from support.PageObjects.GooglePage import GooglePage
from support.StepObjects.GoogleSteps import GoogleSteps

class TestGoogleSearch(unittest.TestCase):

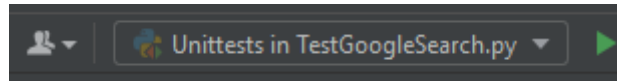
    def test_search(self):
        driver = webdriver
            .Chrome(executable_path="../../webdriver/chromedriver.exe")
        driver.get("https://www.google.com/")
        tester = GoogleSteps(driver)
        tester.setValueInSearch("GeForce 1650")
        result = tester.getCountResultSearch()
        print("Count: ", result)
        self.assertNotEqual(0, result)
        print("Tests finished: SUCCESS")
        driver.close()
        driver.quit()

if __name__ == '__main__':
    unittest.main()
```

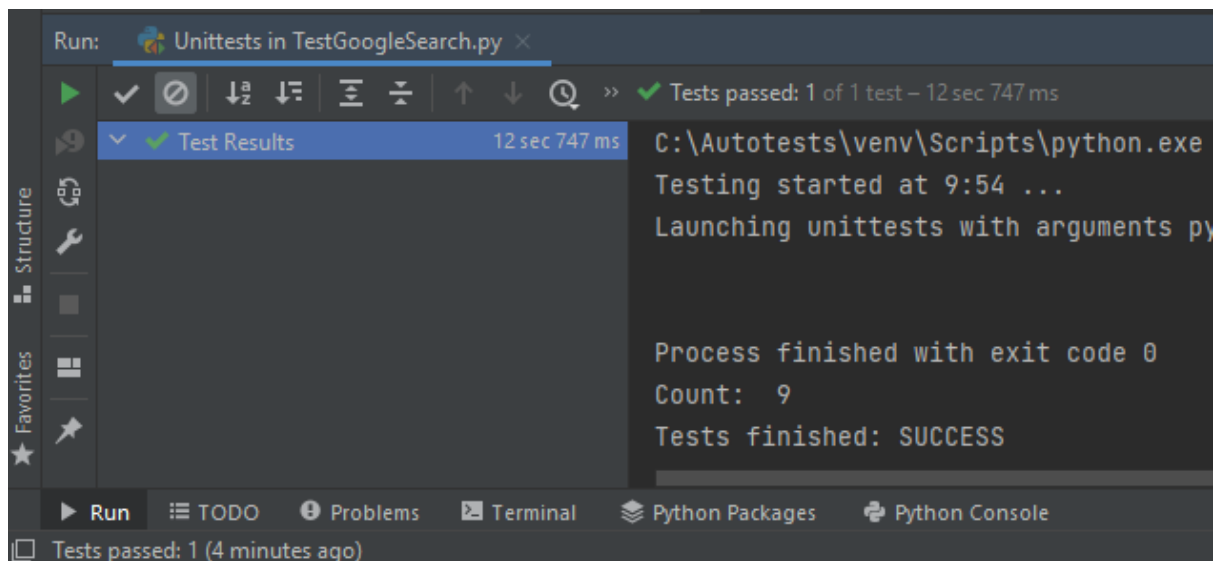
Запускаем автотест TestGoogleSearch.py в окне Project



или через панель инструментов



Если всё было сделано правильно, в результате мы получим сообщение об успешном прохождении тестирования.



```
C:\Autotests\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm
Community Edition
2021.1.3\plugins\python-ce\helpers\pycharm\_jb_unittest_runner.py" --path
C:/Autotests/tests/TestGoogleSearch.py
```

```
Testing started at 10:56 ...
Launching unittests with arguments python -m unittest
C:/Autotests/tests/TestGoogleSearch.py in C:\Autotests\tests
Count: 9
Tests finished: SUCCESS
```

Автотест можно запустить из консоли.

Для этого в переменной укажите точный адрес:

```
driver = webdriver.Chrome(executable_path="C:/Autotests/webdriver/chromedriver.exe")
```

Затем создайте файл run-test.bat в котором впишите строку

```
C:\Autotests\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm
Community Edition
2021.1.3\plugins\python-ce\helpers\pycharm\_jb_unittest_runner.py" --path
C:/Autotests/tests/TestGoogleSearch.py
```

Стек: Python + Robot

...

Стек: PHP + Selenium + Codeception + PHPUnit

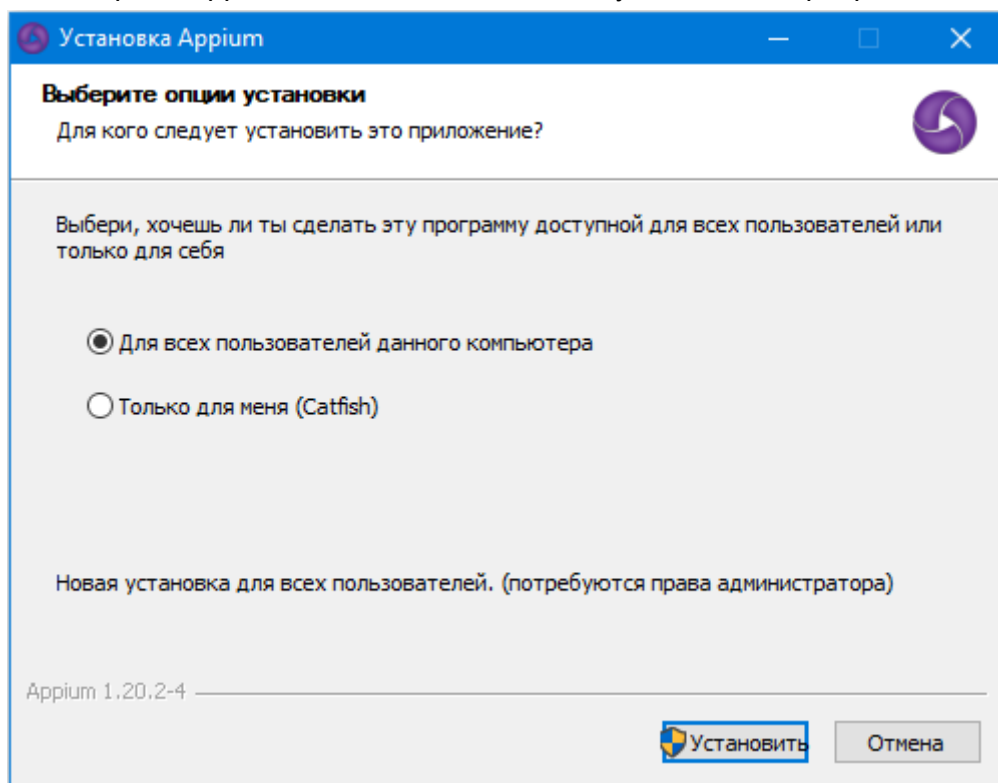
...

Стек: JavaScript + WebdriverIO

...

Appium установка и запуск

Необходимо скачать с официального сайта <http://appium.io/downloads.html>
<https://github.com/appium/appium-desktop/releases/tag/v1.20.2-4>
установочный файл Appium-windows-1.20.2-4.exe и установить сервер



После установки приложение будет находится по адресу:
C:\Users\...\AppData\Local\Programs\Appium\Appium.exe

Второй способ установки через NodeJS с помощью команды:

```
npm install -g appium
npm install -g appium@1.20.2
appium -v
where appium
```

```
Командная строка
Microsoft Windows [Version 10.0.19041.928]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>appium -v
1.20.0

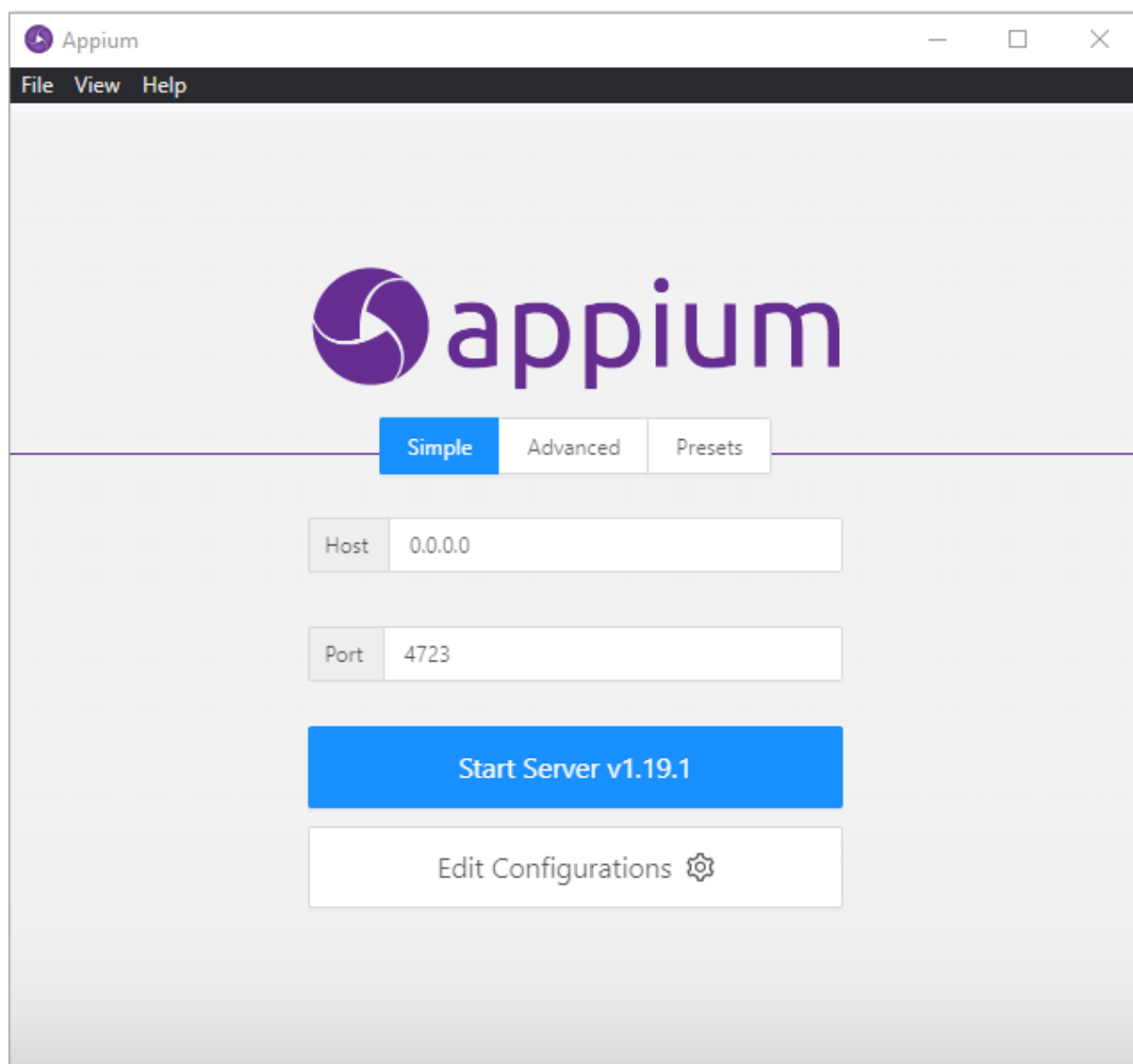
C:\Users\Catfish>where appium
C:\Users\Catfish\AppData\Roaming\npm\appium
C:\Users\Catfish\AppData\Roaming\npm\appium.cmd
```

Запуск appium из командной строки (остановка нажатием Ctrl+C и ввести Y)

```
Командная строка

C:\Users\Catfish>appium
[Appium] Welcome to Appium v1.20.0
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
[Appium] Received SIGINT - shutting down
[debug] [Appium] There are no active sessions for cleanup
[HTTP] Waiting until the server is closed
[HTTP] Received server close event
Завершить выполнение пакетного файла [Y(да)/N(нет)]? y
```

Запуск через GUI интерфейс



оставляем host 0.0.0.0 и port 4723 нажимаем кнопку Start Server

Дополнительные ссылки

Selenium	https://www.selenium.dev/
Appium	http://appium.io/
Chrome Driver (для Chrome)	https://chromedriver.chromium.org/
Gecko Driver (для FireFox)	https://github.com/mozilla/geckodriver/releases
Java SE Development Kit 8 (JDK)	https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

Java SE Runtime Environment 8 (JRE)	https://www.oracle.com/java/technologies/javase-jre8-downloads.html
Microsoft .NET Framework 3.5	https://www.microsoft.com/ru-ru/download/details.aspx?id=22
Visual C++ Redistributable	https://support.microsoft.com/ru-ru/topic/
Maven Repository (менеджер пакетов Java)	https://mvnrepository.com/
NodeJS	https://nodejs.org/
NPM (менеджер пакетов Node.js)	https://www.npmjs.com/
Composer (менеджер пакетов PHP)	https://getcomposer.org/
Python	https://www.python.org/
PHP	https://www.php.net/
IntelliJ IDEA (IDE Java)	https://www.jetbrains.com/idea/
PyCharm (IDE Python)	https://www.jetbrains.com/ru-ru/pycharm/
NetBeans (IDE PHP)	https://netbeans.apache.org/
Visual Studio Code (IDE JavaScript)	https://code.visualstudio.com/
Android Studio (Android SDK)	https://developer.android.com/studio
UiAutomator 2	https://github.com/appium/appium-uiautomator2-driver
Git	https://git-scm.com/
SmartGit (менеджер для Git)	https://www.syntevo.com/smartgit/
JUnit	https://junit.org/junit5/
TestNG	https://testng.org/
UnitTest	https://docs.python.org/3/library/unittest.html
PyTest	https://docs.pytest.org/en/6.2.x/
PHPUnit	https://phpunit.de/
Selendroid	http://selendroid.io/
Cucumber	https://cucumber.io/
Robot Framework	https://robotframework.org/
Codeception	https://codeception.com/
WebdriverIO	https://webdriver.io/

[illegible]