Пишем автотесты

Цели: Познакомиться с тестовыми фреймворками

План урока:

- JUnit
- TestNG
- Аннотации
- Отчеты

JUnit

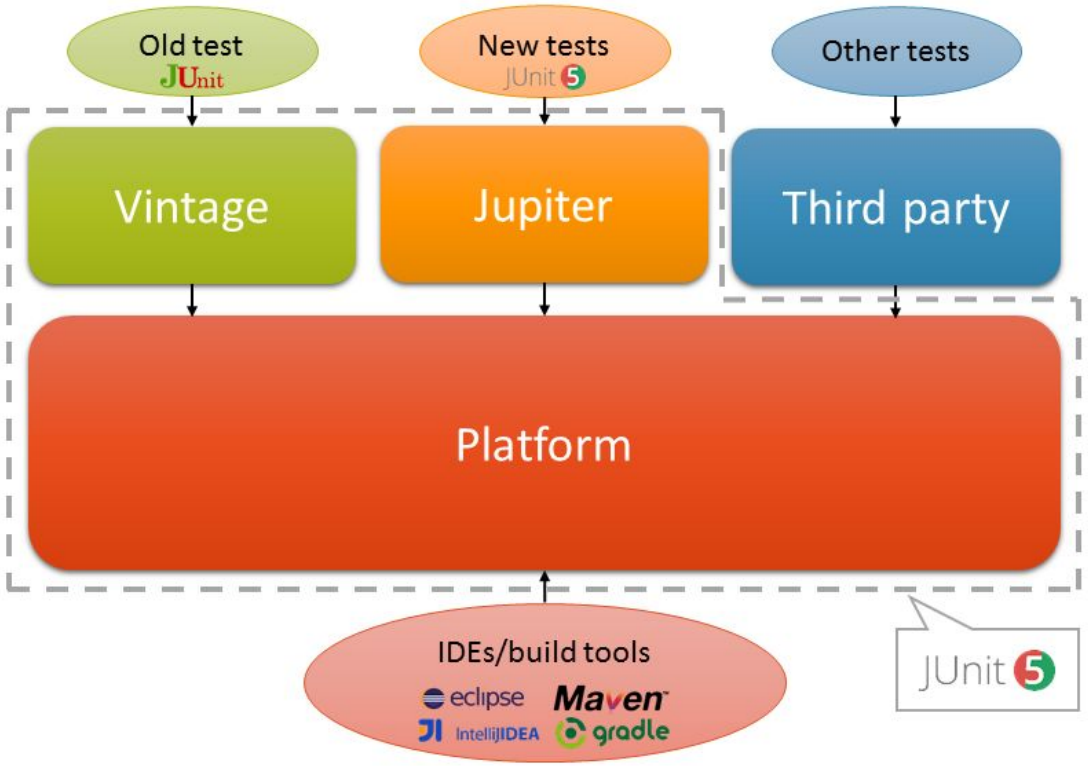
JUnit — фреймворк автоматического тестирования программного обеспечения на языке Java.

Созданный Кентом Беком и Эриком Гаммой, JUnit принадлежит семье фреймворков xUnit для разных языков программирования, берущей начало в SUnit Кента Бека для Smalltalk. JUnit породил экосистему расширений — JMock, EasyMock, DbUnit, HttpUnit и т. д.

Библиотека **JUnit** была портирована на другие языки, включая PHP (PHPUnit), C# (NUnit), Python (PyUnit), Fortran (fUnit), Delphi (DUnit), Free Pascal (FPCUnit), Perl (Test::Unit), C++ (CPPUnit), Flex (FlexUnit), JavaScript (JSUnit).

Архитектура JUnit5

JUnit 5 = Платформа JUnit + JUnit Jupiter + JUnit Vintage



- JUnit Jupiter: включает новые модели программирования и расширения для написания тестов. В нем есть все новые аннотации junit и TestEngine реализация для запуска тестов, написанных с этими аннотациями.
- Платформа JUnit: чтобы иметь возможность запускать тесты junit, IDE, инструменты сборки или плагины должны включать и расширять API платформы. Он определяет TestEngine API для разработки новых фреймворков тестирования, работающих на платформе. Он также предоставляет средство запуска консоли для запуска платформы из командной строки и создания подключаемых модулей для Gradle и Maven.
- [illegible]ать выполнение на платформе JUnit 5 тестов, написанных для JUnit 3 и JUnit 4. Это есть обратная совместимость.

JUnit аннотации

`@BeforeEach` - Аннотированный метод будет запускаться перед каждым тестовым методом в тестовом классе.

`@AfterEach` - Аннотированный метод будет запускаться после каждого тестового метода в тестовом классе.

`@BeforeAll` - Аннотированный метод будет запущен перед всеми тестовыми методами в тестовом классе. Этот метод должен быть статическим.

`@AfterAll` - Аннотированный метод будет запущен после всех тестовых методов в тестовом классе. Этот метод должен быть статическим.

`@Test` - Он используется, чтобы пометить метод как тест junit.

`@DisplayName` - Используется для предоставления любого настраиваемого отображаемого имени для тестового класса или тестового метода

`@Disable` - Он используется для отключения или игнорирования тестового класса или тестового метода из набора тестов.

`@Nested` - Используется для создания вложенных тестовых классов

`@Tag` - Пометьте методы тестирования или классы тестов тегами для обнаружения и фильтрации тестов.

`@TestFactory` - Отметить метод - это тестовая фабрика для динамических тестов.

Документация:

https://junit.org/junit5/docs/current/user-guide/#writing-tests-parameterized-tests

TestNG

TestNG — это среда тестирования, вдохновленная JUnit и NUnit, но представляющая некоторые новые функциональные возможности, которые делают ее более мощной и простой в использовании.

TestNG реализует core логику JUnit, который старше него и послужил вдохновением для первого. Но TestNG (NG означает Next Generation) изначально создавался для функционального и более высоких уровней тестирования, позиционировался как более простой в использовании, чем заслужил признание автоматизаторов.

Документация:

https://testng.org/doc/documentation-main.html

TestNG аннотации

@BeforeSuite - Метод с этой аннотацией будет выполнен до любых тестов, описанных в тестовом наборе testing.

@AfterSuite - Метод с такой аннотацией будет запущен после всех тестов из тестового набора TestNG.

@BeforeTest - Такие методы будут выполняться до каждой секции Test в наборе тестов.

@AterTest - Методы будут запускаться после каждой секции Test в тестовом наборе.

@BeforeGroups - Метод с аннотацией @BeforeGroup будет запускаться 1 раз до выполнения любого тестового метода указанной группы.

@AfterGroups - Такой метод будет запускаться 1 раз после запуска всех тестовых методов из указанной группы.

@BeforeClass - Помеченный метод выполняется однажды до запуска любого тестового метода в определенном тестовом классе.

@AfterClass - Такой метод запускается после выполнения всех тестовых методов тестового класса.

@BeforeMethod - Эти методы выполняются до запуска каждого тестового метода.

@AfterMethod - Такие методы запускаются после выполнения каждого тестового метода.

@DataProvider - Помечает метод как предоставляющий данные для тестового метода. Этот метод возвращает двумерный массив Object [ ] [ ] в качестве данных.

@Factory - Указывает что метод является фабрикой, возвращающей массив объектов Object[ ]. Эти объекты будут в последствии использованы как тестовые классы фреймворком TestNG. Такой механизм нужен для запуска набора тест-кейсов с различными значениями.

@Listeners - Применимо к тестовым классам. Определяет массив тестовых классов прослушивателей, реализующих org.testng.ITestNGListener. Предназначена для отслеживания статуса выполнения и логирования.

@Parameters - Аннотация используется для передачи параметров тестовому методу. Эти значения параметров подставляются из файла конфигурации testing.xml во время выполнения.

@Test - Отмечает класс или метод как тестовый. Если используется на уровне класса, все public методы этого класса помечаются как тестовые методы.

JUnit5 vs TestNG

	JUnit 5	TestNG
Аннотация теста	@Test	@Test
Название теста	@DisplayName	@Test(description = «test name»)
Запуск перед сьютом	-	@BeforeSuite
Запуск после сьюта	-	@AfterSuite
Запуск перед тестированием	-	@BeforeTest
Запуск после тестирования	-	@AfterTest
Запуск перед тестом из группы	-	@BeforeGroups
Запуск после теста из группы	-	@AfterGroups
Запуск перед классом	@BeforeAll	@BeforeClass
Запуск после класса	@AfterAll	@AfterClass
Запуск перед каждым тестовым методом	@BeforeEach	@BeforeMethod
Запуск после каждого тестового метода	@AfterEach	@AfterMethod
Игнорировать тест	@Disabled	@Test(enable=false)
Test factory для динамических тестов	@TestFactory	@Factory
Тегирование	@Tag	-
Таймаут	@Test(timeout = 1000)	@Test(timeout = 1000)

JUnit5 vs TestNG

https://habr.com/ru/company/otus/blog/544770/

https://www.lambdatest.com/blog/junit-5-vs-testng/

Allure

Allure Framework – популярный инструмент построения отчётов автотестов, упрощающий их анализ. Это гибкий и легкий инструмент, который позволяет получить не только краткую информацию о ходе выполнения тестов, но и предоставляет всем участникам производственного процесса максимум полезной информации из повседневного выполнения автоматизированных тестов.

Разработчикам и тестировщикам использование отчетов Allure позволяет сократить жизненный цикл дефекта: падения тестов могут быть разделены на дефекты продукта и дефекты самого теста, что сокращает затраты времени на анализ дефекта и его устранение. Также к отчету могут быть прикреплены логи, обозначены тестовые шаги, добавлены вложения с разнообразным контентом, получена информация о таймингах и времени выполнения тестов. Кроме того, Allure-отчеты поддерживают взаимодействие с системами непрерывной интеграции и баг-трекинговыми системами, что позволяет всегда держать под рукой нужную информацию о прохождении тестов и дефектах.

Документация:

https://docs.qameta.io/allure/

Вопросы?