Java-Testframeworks

Kurze Definition, was ein Java-(Unit-)Testframework ist

standardisiertes, ausgefeiltes und erweiterbares Mittel zum Erstellen einer Software

Inhalt sind Pakete mit vorgefertigten Codes und/oder Bibliotheken, Compiler oder ähnliche Tools, welche die Sicherheit und Effizienz zu ermitteln erlauben, sowie deren Anpassung

Auflistung der Vorstellung

- Karate
- JBehave
- JUnit
- Serenity (auch gen. Thucydides)
- TestNG
- Selenide
- Gauge
- Gab
- Spock
- HttpUnit
- JWebUnit

Karate (BDD testing framework)

Vorteile:

Art der Testfallbeschreibung sehr leicht → einfache Testszenarien-Erstellung

parallele Ausführung von Testfällen zum Vergleichen möglich

kann um JavaScript erweitert werden

Nachteile:

unterstützt nicht den üblichen Cucumber Sprachansatz, ist also differenziert davon

JBehave (BDD)

Vorteile:

für Java entwickelt → Besonders detailliert angepasst und intuitiv

Bietet die Integration zwischen verschiedenen Entwicklerteams und Projekten an

Semi-formale Sprachen → Leichtes Verständnis insgesamt

Nachteile

Kommunikation muss manuell erfolgen (wenn man das einen Nachteil nennen kann)

Einschätzung von der Quelle:

JBehave - das Testing-Framework für Java funktioniert ähnlich wie Frameworks wie Serenity. Wenn Sie jedoch vorhaben, die automatisierten Akzeptanztests effektiv zu erstellen, ist es besser, JBehave zusammen mit Serenity einzubinden, um eine ideale Testerfahrung zu nutzen. Das gesamte Konzept von Serenity basiert auf der BDD-

Entwicklung und ermöglicht es den Anwendern auch, einen umfangreichen und leistungsstarken Testbericht zu schreiben.

JUnit (BDD)

Vorteile:

für Regressionstests und wiederholende Testfälle (schreiben und implementieren)

Leichte Bug-Erkennung + Lesefreundlich

Unterstützt viele IDE´s wie Eclipse, IntelliJ oder NetBenas

viele Javaversionen unterstützt

Nachteile

Keine Abhängigkeitstests → Ausweichen auf TestNG möglich

Hinweis

Codeänderung → Testfälle neustarten um Probleme direkt feststellen zu können

Einschätzung von der Quelle:

JUnit ist ein weit verbreitetes Open-Source-Framework, das auf Java basiert. Sowohl TestNG als auch JUnit machen einen ähnlichen Job und ihre Haupteigenschaften sind in gewisser Weise die gleichen, abgesehen von den Möglichkeiten des Abhängigkeitstests. Auch die Vorgehensweise bei der Ausführung von parametrisierten Tests ist in beiden Frameworks unterschiedlich.

Aufgrund der Tatsache, dass JUnit schon seit längerer Zeit eingesetzt wird, gibt es eine große Unterstützung durch die Community. Durch die Integration von Selenium WebDriver für Java-basierte Anwendungen wurde es auch als Standard Java Unit Testing Frameworks definiert. Obwohl TestNG nur wenige Benutzer hat, wächst die Community stetig. Wir schließen also mit dem Vorschlag, dass die Wahl zwischen JUnit oder TestNG für ein Java-Testframework ganz von der Art und den Notwendigkeiten der spezifischen Anwendungen abhängt.

Serenity (BDD)

Vorteile:

großer Wert auf Akzeptanzkriterien (sauber, strukturiert und handhabbar)

Erweiterung von Selenium WebDriver und JUnit

explizite Unterstützung für REST Dienste

Automatisierungsframework wird „mitgeliefert“

hohe Kompatibilität mit anderen Frameworks

Nachteile:

KnowHow in Java benötigt für die Verwendung und Maven 3.0 oder höher muss da sein (+JDK5 oder höher)

Absprachen im Team sind für einen effektiven Ablauf von nöten

Einschätzung von der Quelle:

Dieses Java-Testing-Framework wird hauptsächlich für das Schreiben und Berichten von Akzeptanzkriterien verwendet. Die mit Serenity geschriebenen Berichte sind sehr effektiv, nützlich, vorteilhaft, reichhaltig und sehr

informativ im Vergleich zu anderen BDD-Frameworks wie JBehave. Es unterstützt die Programmierer beim Schreiben von qualitativ hochwertigen Testfallszenarien für die Automatisierung. Serenity unterstützt auch Selenium WebDriver und Rest Assured, was es für die QA-Ingenieure schneller und einfacher macht, automatisierte Akzeptanztestkriterien zu schreiben.

TestNG (NG=Nächste generation)

Vorteile:

- Hat Funktionen, die es JUnit überlegen machen sollen
- Deckt alle Softwaretests ab
- Erstellen und Ausführen paralleler Tests möglich (Auch HTML Berichterstellung unterstützt
- Priorisierung möglich

Nachteile:

- Neueste Version von JDK und spezialisiert auf Eclipse (was nicht jeder mag)
- Einrichtung dauert länger

Einschätzung von der Quelle:

Wie wir bereits wissen, erfüllt das Test-Framework JUnit die gleichen Funktionen wie TestNG, hat aber einige Nachteile. JUnit ist erstaunlich, wenn der Test isoliert durchgeführt wird; wenn es jedoch eine Art von Abhängigkeit gibt, haben Sie keine Kontrolle darüber, welcher Test früher durchgeführt wird. TestNG-Frameworks für Java helfen Ihnen, indem Sie Testfälle so ausführen lassen, wie Sie es möchten.

Außerdem erhalten Sie zusätzliche Unterstützung für die Parametrisierung in TestNG. Auch wenn diese besondere Eigenschaft bereits in JUnit 4.5 zu finden ist, ist das TestNG-Framework noch hilfreicher. Wenn ein Projekt komplex ist und Sie vielleicht 100 oder mehr Testfälle benötigen, ist es besser, sich etwas Zeit zu nehmen und ein TestNG-Framework einzurichten, als sich komplett auf andere Frameworks wie JUnit zu verlassen.

Selenide (basiert auf Selenium WebDriver)

Mehr für Apps geeignet, aber auch eine Erweiterung von JUnit

Gauge (ebenfalls von den SeleniumSchöpfern)

Vorteile:

- Einfache und flexible Syntax
- Viele IDS`s unterstützt
- parallele Test möglich
- ausgelegt auch für große Datenmengen

Nachteile:

- Neu, vllt noch nicht alle Kinderkrankheiten ausgemerzt
- Probleme mit IntelliJ und Specs

Einschätzung von der Quelle:

Das Java-Testing-Framework Gauge eignet sich hervorragend für die Durchführung von Cross-Browser-Tests. Dieses Framework hat die Testautomatisierungsverfahren unglaublich revolutioniert und ermöglicht es auch nicht-technischen Anwendern, mit Leichtigkeit ein Testautomatisierungs-Framework zu schreiben. Es löst echte Anomalien bei der Testwartung - dank seiner wachsenden Community und dem Beitrag von Early Adopters.

Gab (basierend auf Groovy und ebenfalls Selenium WebDriver)

Vorteile:

für Browser optimiert

Testautomatisierung für Screen Scraping, Webtests, …

Integriert TestNG und JUnit

Der englischen Sprache sehr ähnlich (wenn man das einen Vorteil nennen möchte)

Nachteile:

LambdaTest-Authentifizierungs- und Validierungsmeldedaten nötig

Sollte nur für große Projekte verwendet werden (aufwendig und in der Ausführung Datenlastig

Einschätzung von der Quelle:

Das Java-Testing-Framework Geb bietet intelligente Integrationsmodule für Frameworks wie Cucumber, TestNG, JUnit und Spock. Allerdings ist Spock eine großartige Ergänzung zu Geb. Die Verwendung von Geb mit Spock kann prägnante, klare und verständliche Testspezifikationen mit weniger Aufwand liefern.

Spock (in Groovy geschrieben)

Vorteile:

bequeme Nutzung (Lesbarkeit, Stubbing und Mocking eingebaut, kurze Parametrisierung und hat eine einfache und kommunikative domänenspezifische Sprache)

Nachteile:

Kenntnisse in Groovy von nöten

Einschätzung von der Quelle:

Ja, viele halten es für das beste Java-Testframework und setzen es an die erste Stelle. Spock ist das beste Testframework für Behaviour-Driven Development, weil es eine leistungsstarke DSL, eine einfache Bedienung und eine intuitive Benutzeroberfläche bietet. Vor allem, wenn Sie einige der Anwendungen haben, die auf der JVM basieren. Spock ist inspiriert von RSpec, jMock, JUnit, Mockito, Vulcans, Groovy, Scala und anderen fesselnden Lebensformen.

HttpUnit (quelloffen)

Vorteil und Nachteil:

eher zum Testen von Webseiten ohne Webbrowserabhängigkeit und komplex

JWebUnit (ebenfalls eher für Webapplikationen gedacht)

Vorteile:

eine bevorzugte JUnit-Erweiterung für Integrations-, Regressions- und Funktionstests

Einschätzung und Empfehlung:

Ich würde persönlich für den Einsatz beim Programmieren des Spiels „*Mecha War*“ (welches auf RoboRally basiert) folgende zwei Testframeworks empfehlen:

Das schon oftmals, wie in Eclipse, implementierte JUnit, da wir einerseits dort die Javasyntax ohne großartiges Umgewöhnen effektiv nutzen können und andererseits der Umfang für unsere Zwecke mehr als ausreichen sollte.

Als zweites Framework fällt meine Empfehlung auf Serenity, da es eine Erweiterung von JUnit ist, ihm sehr ähnlich und soweit ich weitere Quellen gelesen habe auch sehr Aussage/Informationsfreundlich. Es bietet sehr ausführliche Auswertungen

Quellen:

https://www.lambdatest.com/blog/top-10-java-testing-frameworks/

https://www.computerweekly.com/de/video/Fuenf-Java-Test-Frameworks-die-JDK-Entwickler-kennen-muessen

https://entwickler.de/online/javascript/test-framework-karate-579924613.html

https://intuit.github.io/karate/ & https://github.com/intuit/karate

[diverse YT Videos, um zu erkennen, was Frameworks genau machen -> Bsp.:

https://www.youtube.com/watch?v=CK8shWNtizk]

http://www.thucydides.info/#/