Sakuli 2.x

Was war, was ist, was sein wird

Simon Hofmann
Software Engineer

28.05.2019

Agenda

- Intro
- Was war: Sakuli v1.x
- Was ist: Sakuli v2.x
  - Technologiestack
  - Architektur
  - Sakuli Lifecycle
  - Plugin System
- Was sein wird: Sakuli Roadmap
  - Short Term
  - Mid Term
  - Long Term

Intro

Intro

Simon Hofmann

Software Engineer

ConSol Consulting & Solutions GmbH

Maintainer / Entwickler von Sakuli

Fokusthema Testautomatisierung

 simon.hofmann@consol.de

 @s1hofmann

 s1hofmann

Was war: Sakuli v1.x

Was war: Sakuli v1.x

- 16. Mai 2014: Erster Release auf GitHub (v0.4.0)
  - Initiale Entwicklung von Tobias Schneck und Simon Meggle
- Sakuli mit OMD Anbindung
- 1847 Commits auf master seit dem ersten public Release
- 13 Veröffentlichungen zum Thema Sakuli und Containerized E2E Testing / Monitoring
- “Containerized E2E Testing”

Was war: Sakuli v1.x

Was war: Sakuli v1.x

- Sakuli besitzt zwei Kernkomponenten:
  - Sahi: Web
  - Sikuli: Desktop
- Sahi OS wurde 2015 eingestellt
  - Sahi stellt den Testrunner in Sakuli bereit
  - Ohne eigene Modifikationen keine Weiterentwicklung möglich
- Entscheidung für Eigenentwicklung für mehr Flexibilität

Was ist: Sakuli v2.x

Sakuli v2.x - Technologiestack

v1.x

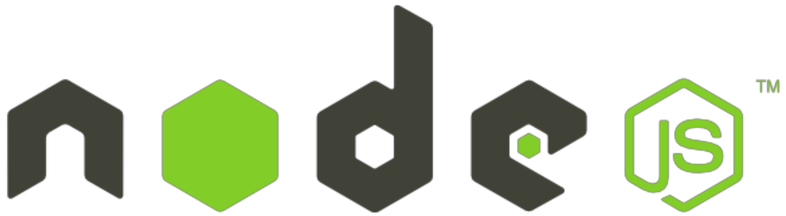
- Runtime: JVM
- Web Testing: Sahi (mittels Proxy)
- Native Testing: Sikuli
- Programmiersprache
  - Framework: Java
  - Tests: SahiScript
- Installation: Native Installer

v2.x

- Runtime: Node.js
- Web Testing: Webdriver
- Native Testing: nut.js
- Programmiersprache
  - Framework: JavaScript
  - Tests: JavaScript
- Installation: Per npm CLI

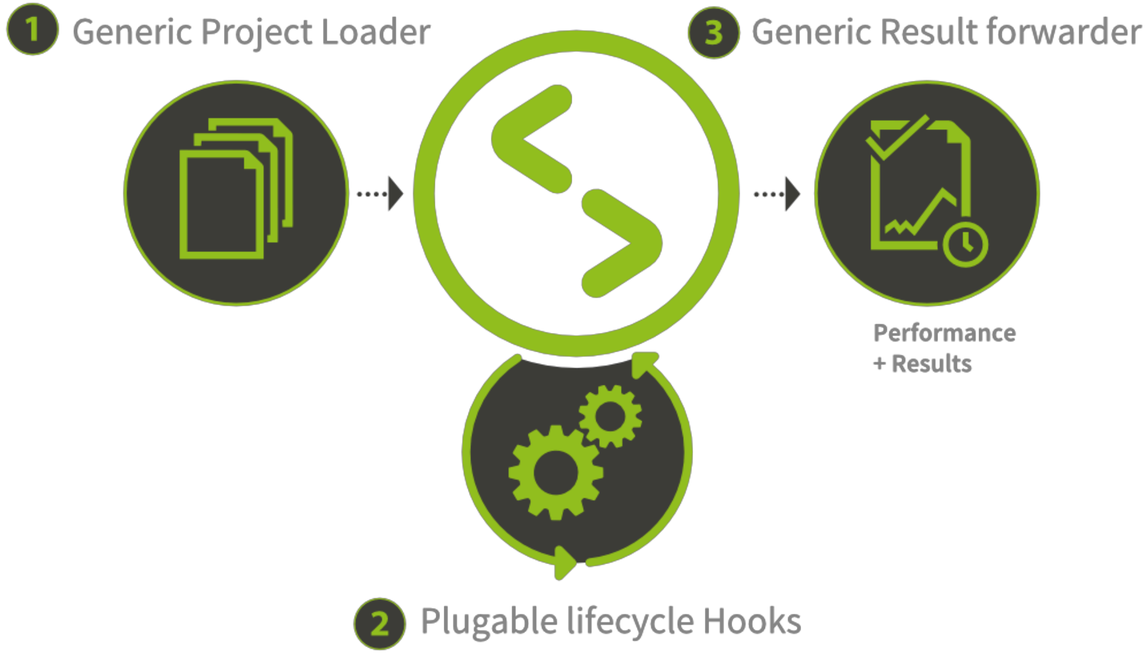
Sakuli v2.x - Technologiestack

- Kein Eingriff in Netzwerkkommunikation durch Wegfall des Sahi Proxy
- Konsistente Nutzung der selben Programmiersprache für Framework und Tests
- Tests werden in modernem Javascript geschrieben
- Möglichkeit der Nutzung des gesamten node Ökosystems, auch in Tests
- Installation über etablierte Node.js Tools möglich



Sakuli v2.x - Architektur

- Fokus von Sakuli 2.x liegt auf Flexibilität und Erweiterbarkeit
- Kern des Frameworks ist ein generischer Testrunner
- Testrunner verarbeitet verschiedene Kombinationen von
  - Project Loader
  - Context Provider
  - Result Forwarder
- Erlaubt flexible Konfiguration und eigene Erweiterungen



Sakuli v2.x - Project Loader

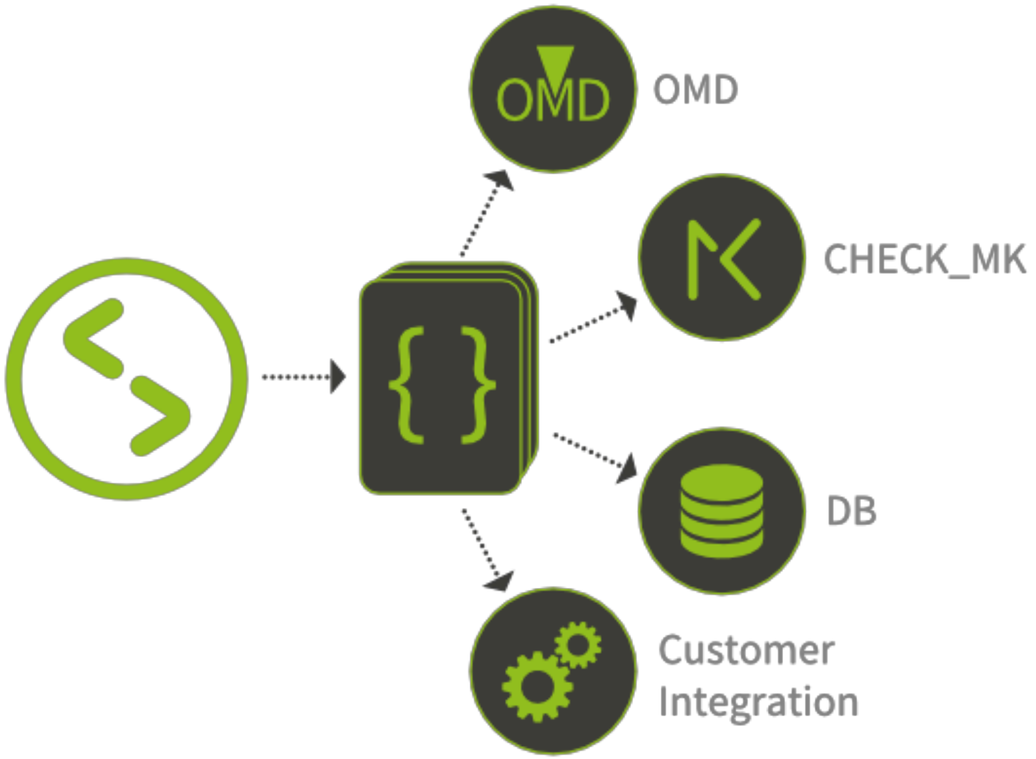
- Project Loader stellen abstrakte “Projekte” bereit
- Über Project Loader wird definiert, wie
  - Testdateien geladen werden
  - Properties geladen werden
- Eigene Implementierung eines Project Loaders erlaubt einfache Anbindung an Drittsysteme

Sakuli v2.x - Context Provider

- Context Provider stellen die Laufzeitumgebung eines Tests
- Definieren, welche Komponenten zur Laufzeit verfügbar sind
- Im Legacy Context z.B bekannte Klassen aus Sakuli 1.x:
  - Environment
  - Region
  - TestCase
  - etc.
- Können auf Lifecycle Events reagieren

Sakuli v2.x - Result Forwarder

- Sakuli stellt Testergebnisse als Datenobjekt bereit
- Plugins steht eine definierte Schnittstelle bereit
- Forwarder implementieren Logik zur Weitergabe der Daten
- Automatische Ausführung durch das Sakuli Plugin System



Sakuli v2.x - Lifecycle

- Testausführung erfolgt in einem definierten Zyklus
- Context Provider können auf Events in diesem Zyklus reagieren
- Verfügbare “Lifecycle Hooks”:
  - ***onProject***: Projekt wurde erfolgreich geladen und steht zur Verfügung
  - ***beforeExecution***: Testsuite Ausführung startet, noch keine Test ausgeführt
  - ***beforeRunFile***: Testfile Ausführung startet, noch kein Test ausgeführt
  - ***readFileContent***: Testfile wird gelesen, Input für die Testausführung
  - ***afterRunFile***: Testcode in einer Datei wurde erfolgreich ausgeführt
  - ***afterExecution***: Alle Tests eines Projekts wurden ausgeführt

Sakuli v2.x - Plugin System

- Sakuli Plugins werden als Dependency installiert (npm install …)
- Plugins:
  - Registrieren sich selbständig an der Sakuli Plugin Registry
  - Lesen eigenständig die benötigten Properties aus der jeweiligen Quelle
  - Werden zur Laufzeit automatisch über die Registry aufgerufen

Sakuli v2.x - Enterprise Support

- Gewartete VNC Docker Images
- Monitoring Forwarder
- Gestaffelte Angebote
  - Umfang
  - Anzahl der Instanzen
  - Support Zeiten

Was sein wird: Roadmap

Sakuli Roadmap - Short Term

- Hypercare Phase
  - Kontinuierliche Verbesserung / Stabilisierung von Sakuli v2.x
  - Feedback von Kunden und der Community
- Community Auf- / Ausbau und Pflege

Sakuli Roadmap - Mid Term

- Neue API
  - Alternative zur bestehenden API, welche aus Sakuli 1.x übernommen wurde
- Testrecorder
  - Web
  - Native
  - Snapshot Tests
- Sakuli Grid
  - Remote Ausführung von Sakuli Tests

Sakuli Roadmap - Long Term

- Sakuli Studio
  - Voll integrierte Lösung mit
    - Testrecorder
    - Testmanagement
    - Testeditor
    - Remote Execution

Sakuli - Links

https://sakuli.io

https://labs.consol.de/sakuli/

https://github.com/sakuli

https://twitter.com/sakuli_e2e

Demo

Noch Fragen?

Vielen Dank!

ConSol
Consulting & Solutions Software GmbH

St.-Cajetan-Straße 43
D-81669 München
Tel.: +49-89-45841-100
info@consol.de
www.consol.de
Twitter: @consol_de