

Otomatisasi Pengujian Perangkat Lunak

(Software Testing Automation)



Lamhot JM Siagian

KATA PENGANTAR

Puji dan syukur penulis sampaikan kepada Tuhan yang Maha Esa atas penyertaannya, penulis bisa menyelesaikan buku yang berjudul “Otomatisasi Pengujian Perangkat Lunak”. Selain itu penulis juga ingin menyampaikan terimakasih kepada keluarga yang turut mendukung penulis. Terimakasih juga kepada teman yang memberikan dukungan (Rolastri, Mariyani, Wati, Ayu, Lestari, Frisca, Lasma, Siska, Melyana, Ulva, Trima, dan Olvi) yang turut membantu dalam proses bertukar pikiran dan memberikan saran membangun. Terimakasih juga untuk mentor saya Mas Ragapinilih untuk ilmu-ilmu baru dan sebagian saya tuliskan pada buku ini.

Automation Testing merupakan suatu teknologi yang sedang *booming* di dunia software testing. Tidak bisa dipungkiri bahwa *automation testing* sangat dibutuhkan untuk proses pengujian ulang seluruh fungsionalitas perangkat lunak. Untuk itu penulis tertarik mengambil topik mengenai *automation testing* dan terbilang jarang dibahas pada buku Bahasa Indonesia.

Buku ini akan membantu para software engineer in test atau QA engineer pemula untuk mengetahui lebih dalam mengenai automation testing. Penulis banyak membahas konsep *software testing*, *automation testing* web, *API testing*, *performance testing*, dan cara Konfigurasi *Continuous Integration* (CI) dengan menggunakan Travis. Penulis membuat buku dengan konsep sedikit teori, mini proyek setiap tools dan proses instalasi. Penulis mengharapkan setelah membaca buku ini, pembaca memahami dan mempraktikkan topik-topik berikut saat melakukan pengujian perangkat lunak.

1. Konsep Software Testing
2. Pemakaian Selenium IDE & Selenium Webdriver
3. *API automation testing* dengan Postman, Newman & Cucumber API
4. *Web Automation Testing* dengan Cucumber Ruby
5. *Performance Testing* dengan JMeter & Locust
6. Pengaturan *Continuous Integration* dengan Travis CI

Penulis menyadari masih terdapat kekurangan dalam buku ini untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diapresiasi. Semoga buku ini dapat memberi manfaat bagi yang tertarik dengan *automation testing* perangkat lunak, secara khusus dan bagi semua pihak yang membutuhkan.

Jakarta, 8 Agustus 2018

Lamhot JM Siagian

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	iii
DAFTAR TABEL	vii
DAFTAR GAMBAR	viii
DAFTAR LAMPIRAN.....	xii
BAB 1 Konsep Testing	1
1.1 Pengertian Testing	1
1.2 Istilah Pengujian	1
1.3 Metodologi Testing	4
1.3.1 Pengujian White-Box	5
1.3.2 Pengujian Black-Box	6
1.3.3 Pengujian Grey-Box.....	6
1.4 Proses Pengujian.....	6
1.5 Pelaporan Bug	8
1.6 Jenis Testing	10
1.6.1 Pengujian Fungsional	10
1.6.2 Pengujian Non-fungsional.....	15
BAB 2 Konsep Automation Testing	19
2.1 Mengapa Perlu Automation Testing?	19
2.2 Manual Testing vs Automation Testing	21
2.3 Best Practice Automation Testing	22
BAB 3 Selenium Web Automation Testing Tools.....	29
3.1 Pengenalan Selenium	29
3.2 Selenium IDE	30
3.2.1 Instalasi Selenium IDE di Chrome.....	30
3.2.2 Instalasi Selenium IDE di Firefox.....	32
3.2.3 Fitur Selenium IDE	34
3.2.4 Contoh Selenium IDE Tests.....	35
3.3 Cara Mengambil Locator <i>Element</i> HTML	42
3.3.1 Identifikasi melalui ID	42
3.3.2 Identifikasi melalui nama <i>element</i>	43
3.3.3 Identifikasi Melalui Link Text	44
3.3.4 Identifikasi Melalui Partial Link Text.....	44
3.3.5 Identifikasi Melalui TagName	44
3.3.6 Identifikasi Melalui Nama Class.....	44
3.3.7 Identifikasi Melalui CSS Selector	45
3.3.8 Identifikasi Melalui Xpath Selector	45

3.4	Inspect CSS dan Xpath <i>Element</i>	47
3.4.1	Inspect Elemen dengan ChroPath	47
3.4.2	Inspect Elemen dengan Chrome.....	48
3.5	Ruby Selenium Webdriver	49
3.5.1	Locators.....	50
3.5.2	Forms	50
3.5.3	Images	51
3.5.4	Checkboxes	53
3.5.5	Radio Buttons.....	54
3.5.6	Select Boxes	54
3.5.7	Tables	56
3.5.8	Javascripts	57
3.5.9	Frames	58
3.6	Menjalankan Mini Proyek Ruby Selenium Webdriver	59
BAB 4	Cucumber.....	62
4.1	Mengapa Cucumber	62
4.2	Bagaimana Cucumber Bekerja	63
4.3	Anotasi Cucumber	64
4.3.1	Feature.....	64
4.3.2	Background	65
4.3.3	Scenario.....	66
4.3.4	Scenario Outline.....	66
4.3.5	Given, When, Then, And, But.....	67
4.3.6	Data Tables.....	68
4.4	Tag Cucumber	68
4.5	Comments Cucumber	69
4.6	Hooks Cucumber	69
4.7	Perintah pada Cucumber	70
4.8	Laporan Cucumber	70
BAB 5	Cucumber Web Ruby.....	72
5.1	Gems Cucumber Web Ruby.....	72
5.2	Mini Proyek Cucumber Web Ruby	74
BAB 6	API Automation Testing.....	79
6.1	Pengenalan API	79
6.2	Pengenalan API Testing	81
6.3	Automation Testing dengan Postman.....	82
6.3.1	Instalasi	82
6.3.2	Mengirimkan Request API.....	84
6.3.3	Generate Kode Snippets.....	86
6.3.4	Collection, Environment & Variables.....	88
6.3.5	Menuliskan Test Script	93
6.3.6	Menjalankan Test	96
6.3.7	Integrasi dengan Newman.....	98

6.4	Cucumber API Ruby	103
6.4.1	Pengenalan	103
6.4.2	Instalasi	103
6.4.3	Konfigurasi.....	104
6.4.4	Penggunaan	104
6.5	Mini Proyek API Automation Testing	108
BAB 7	Performance Testing Testing.....	112
7.1	Pengenalan Performance Testing	112
7.1.1	Jenis Performance Testing	112
7.1.2	Jenis Permasalahan Performance Aplikasi.....	113
7.1.3	Performance Testing Process	114
7.1.4	Performance Testing Tools	115
7.2	Locust	115
7.2.1	Pengenalan Locust	115
7.2.2	Install Locust di Mac OS.....	116
7.2.3	Install Locust di Ubuntu.....	117
7.3	Mini Proyek Locust	117
7.4	JMeter	119
7.4.1	Pengenalan JMeter	119
7.4.2	Install HomeBrew	120
7.4.3	Install JMeter.....	120
7.4.4	Upgrade JMeter	120
7.4.5	Cara Menjalankan JMeter	120
7.4.6	Pengaturan JMeter.....	121
7.5	Mini Proyek JMeter	122
7.5.1	Test Plan.....	122
7.5.2	User Defined Variables	123
7.5.3	HTTP Header Manager	124
7.5.4	Thread Group	125
7.5.5	Config <i>Element</i>	126
7.5.6	Report.....	128
7.5.7	Menjalankan Test	129
7.5.8	Result	130
BAB 8	Continuous Integration (CI).....	133
8.1	Pengenalan Continuous Integration	133
8.1.1	Pengertian.....	133
8.1.2	Latar Belakang Penggunaan CI.....	133
8.1.3	Cara Kerja CI	133
8.1.4	Keuntungan Penggunaan CI.....	134
8.2	Pengenalan Travis	134
8.2.1	Kebutuhan	134
8.2.2	Memulai Travis CI	135
8.2.3	Memilih Bahasa Pemrograman	136
8.2.4	Pilihan infrastruktur (Opsional)	136
8.3	8.3 Mini Proyek Konfigurasi CI dengan Travis	137
BAB 9	Ringkasan.....	147

Profil Penulis.....	149
REFERENSI	150

DAFTAR TABEL

Tabel 1.1 Perbedaan Metodologi	4
Tabel 2.1 Tabel Perbandingan Manual dan Automation Testing	21
Tabel 3.1 Jenis Selenium.....	29
Tabel 3.2 Fungsi Opsi Toolbar Selenium IDE.....	34
Tabel 4.1 Anotasi Given, When, Then, And, But	67
Tabel 6.1 Status Code	80
Tabel 6.2 Contoh Variabel Postman	90
Tabel 7.1 JMeter Home.....	122
Tabel 7.2 Cellar Home JMeter	122

DAFTAR GAMBAR

Gambar 1.1 Perbandingan Box Testing	4
Gambar 1.2 Testing Process menurut Hambling (Hambling & Goethem, 2013: 22)..	7
Gambar 1.3 Bug Report Life Cycle (Black 2009: 160)	9
Gambar 3.1 Tautan <i>Download</i> Extention Selenium IDE Chrome	31
Gambar 3.2 Menambahkan Selenium IDE Extension pada Chrome	31
Gambar 3.3 Selenium IDE Chrome	32
Gambar 3.4 Tautan Untuk <i>Download</i> Plugin Selenium IDE	32
Gambar 3.5 Menambahkan Selenium IDE pada Firefox	33
Gambar 3.6 Tampilan Awal Selenium IDE	33
Gambar 3.7 Fitur Selenium IDE	34
Gambar 3.8 Button Record pada Selenium IDE	36
Gambar 3.9 Panel Skrip Test Selenium IDE	36
Gambar 3.10 Button Add new test suite	37
Gambar 3.11 Tampilan Test Suite pada Selenium IDE	37
Gambar 3.12 Tampilan Menu Membuat Test pada Selenium IDE	37
Gambar 3.13 Add New Test Selenium IDE	38
Gambar 3.14 Form Menambah Test Case pada Selenium IDE	38
Gambar 3.15 Menambahkan Test Suite Selenium IDE	38
Gambar 3.16 Form Menyimpan Test Selenium IDE	39
Gambar 3.17 Pilihan Selenium IDE pada halaman Web Browser	39
Gambar 3.18 Asert Title GUI Selenium IDE	40
Gambar 3.19 Command Assert Title Selenium IDE	40
Gambar 3.20 Run All Selenium IDE	41
Gambar 3.21 Panel log Selenium IDE	41
Gambar 3.22 Inspect <i>Element</i> Youtube	43
Gambar 3.23 Aktifkan Developer Mode	47
Gambar 3.24 Aktifkan XPath	47
Gambar 3.25 Absolute <i>Element</i> pada Chrome	48
Gambar 3.26 Copy XPath dari Chrome	48
Gambar 3.27 Tab Pencarian <i>Element</i> Chrome	49

Gambar 3.28 Menjalankan Proyek pada port 8000.....	60
Gambar 3.29 Tampilan Localhost:8000.....	60
Gambar 3.30 Eksekusi Ruby Selenium Web Driver.....	61
Gambar 4.1 Cara Kerja Cucumber.....	63
Gambar 4.2 Cucumber Help	70
Gambar 4.3 Jenkins Cucumber Report	71
Gambar 5.1 Tampilan Report proyek Cucumber Ruby Web.....	77
Gambar 6.1 Web API.....	79
Gambar 6.2 Tampilan Platform yang di Support Postman	82
Gambar 6.3 Tampilan Instalasi Postman pada MAC OS.....	83
Gambar 6.4 Jenis HTTP Request Postman	84
Gambar 6.5 Tampilan New Request	84
Gambar 6.6 Form Save Request ke Collection Postman	85
Gambar 6.7 Tampilan Menyimpan Request Postman.....	86
Gambar 6.8 Tampilan Send Request Postman.....	86
Gambar 6.9 Tampilan Link Code.....	87
Gambar 6.10 Tampilan Kode Snippets	88
Gambar 6.11 Tampilan New Collecton	88
Gambar 6.12 Form Create Collection	89
Gambar 6.13 Tampilan Collection pada Sidebar	89
Gambar 6.14 Tampilan Tombol New Pada Postman.....	90
Gambar 6.15 Tampilan Add Environtment.....	91
Gambar 6.16 Dropdownlist Environment.....	92
Gambar 6.17 Tampilan Enviroment Test.....	93
Gambar 6.18 Memanggil Variabel pada Postman	93
Gambar 6.19 Field Postman Token.....	94
Gambar 6.20 update Testing Environtment	94
Gambar 6.21 <i>Assign Response</i> ke Variabel.....	95
Gambar 6.22 Testing Environtment.....	95
Gambar 6.23 Auto Defined <i>Assertation</i> Test Postman	96
Gambar 6.24 Test Result Postman.....	96
Gambar 6.25 Button Runner	97
Gambar 6.26 Tampilan Menjalankan Collection Postman	97
Gambar 6.27 Tampilan Result Test Postman.....	98

Gambar 6.28 Tampilan npm install Newman	99
Gambar 6.29 Link Export Collection.....	100
Gambar 6.30 Button Export Collection	100
Gambar 6.31 Manage Environments.....	101
Gambar 6.32 Tampilan <i>Download</i> Enviroments	101
Gambar 6.33 Tampilan Menjalankan Newman	102
Gambar 6.34 Hasil Report Testing Newman	103
Gambar 6.35 Trace logging dari request dan <i>response</i> API	104
Gambar 7.1 Performance Testing Process	114
Gambar 7.2 Tampilan Menjalankan Locust dari Consol	118
Gambar 7.3 Halaman UtamaLocust.....	119
Gambar 7.4 GUI Dashboard Locust.....	119
Gambar 7.5 GUI JMeter.....	121
Gambar 7.6 Tampilan JMeter Console	121
Gambar 7.7 Tet Plan JMeter	123
Gambar 7.8 User Defined Variables	124
Gambar 7.9 HTTP Header Manager	125
Gambar 7.10 Halaman Thread Group.....	126
Gambar 7.11 JMeter HTTP Request.....	127
Gambar 7.12 User Defined Variables	127
Gambar 7.13 View Result Tree.....	129
Gambar 7.14 Aggregate Report	129
Gambar 7.15 Menjalankan JMeter	130
Gambar 7.16 JMeter View Result Tree.....	130
Gambar 7.17 JMeter Summary Report	131
Gambar 7.18 JMeter Aggregate Graph	131
Gambar 7.19 JMeter View Result in Table.....	132
Gambar 8.1 Continuous Integration Flow.....	134
Gambar 8.2 Daftar proyek sinkron pada Travis.....	135
Gambar 8.3 Form Menambah Proyek Baru di Github	137
Gambar 8.4 Daftar Proyek pada Travis CI.....	138
Gambar 8.5 Mengaktifkan Travis CI	138
Gambar 8.6 Sumber URL Clone Proyek.....	139
Gambar 8.7 Compare & pull request”.....	141

Gambar 8.8 Open Pull Request.....	141
Gambar 8.9 Triggering CI.....	142
Gambar 8.10 Pass CI.....	142
Gambar 8.11 Failed CI.....	143
Gambar 8.12 CI Details.....	144
Gambar 8.13 Halaman build automation di Travis.....	144
Gambar 8.14 Link Build Status.....	145
Gambar 8.15 Pop-up Link Build Status	145
Gambar 8.16 Tampilan Build Status.....	146

DAFTAR LAMPIRAN

LAMPIRAN 1 Instalasi Ruby & Cucumber	151
LAMPIRAN 2 Install Chrome & Firefox Driver.....	155

BAB 1

Konsep Testing

Pada Bab 1 akan dibahas mengenai pengertian testing, istilah dalam pengujian, metodologi testing, proses testing, pelaporan bug dan *level testing* (jenis testing).

1.1 Pengertian Testing

Menurut standar ANSI/IEEE 1059, *perangkat lunak testing* adalah proses menganalisa suatu entitas perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan (defects/error/bugs) dan mengevaluasi fitur-fitur dari entitas perangkat lunak.

Testing atau pengujian perangkat lunak merupakan proses inti dari jaminan kualitas perangkat lunak. Tujuan pengujian perangkat lunak¹:

1. Pengujian adalah proses eksekusi program dengan maksud menemukan kesalahan.
2. Sebuah kasus uji yang baik adalah salah satu dengan probabilitas tinggi untuk menemukan kesalahan yang belum ditemukan.
3. Sebuah tes yang sukses adalah salah satu yang menemukan kesalahan yang belum ditemukan.

1.2 Istilah Pengujian

Test plan → adalah dokumen yang berisi definisi tujuan dan sasaran pengujian dalam lingkup iterasi (atau proyek), item-item yang menjadi target pengujian, pendekatan yang akan diambil, sumber daya yang dibutuhkan dan poin untuk diproduksi. Dengan kata lain test plan dapat disebut sebagai perencanaan atau skenario untuk melakukan testing yang akan dilakukan baik oleh expert atau user umum.

Test case → adalah sekumpulan dari input tes, kondisi yang akan dieksekusi, dan hasil yang diharapkan (IEEE). Test case adalah kondisi saat testing melakukan suatu aksi pada sistem, tester memberikan data pada sistem yang diuji, dan sistem memberikan beberapa kondisi melalui output yang dapat dibandingkan dengan hasil yang diharapkan. Tindakan, data, dan hasil yang diharapkan adalah bagian utama dari testing dan menghasilkan kondisi tes. Test

¹ Presman, Rouger S, *Software Engineering*, 6th Edition, Mc. Graw Hill, 1997.

case adalah urutan langkah-langkah, sub dari langkah-langkah tersebut, dan tindakan lain yang dijalankan secara berurutan atau merupakan kombinasi dari konsekuensi yang menghasilkan kondisi tes sesuai harapan dan berguna untuk evaluasi lebih lanjut Black (2009:610).

Test plan → adalah sebuah dokumen yang berisi tentang penjelasan, strategi, metode dan juga kondisi-kondisi dalam melakukan test nantinya (Black, 2009:50).

Test tool → adalah tujuan umum dari hardware, perangkat lunak yang digunakan selama eksekusi test case untuk mengatur lingkungan, menciptakan kondisi pengujian, atau untuk mengukur hasil pengujian. Sebuah test tool terpisah dari test case itu sendiri (Black, 2009:612).

Bug report → adalah dokumen teknis yang menjelaskan berbagai tanda atau failure mode yang terkait dengan setiap bug. *Bug report* yang baik yaitu yang menyediakan informasi yang dibutuhkan tim proyek management untuk memutuskan kapan dan bagaimana cara memperbaiki bug tersebut. *Bug report* yang baik juga menyimpan informasi yang akan dibutuhkan seorang programmer untuk memperbaiki dan debug masalah (Black, 2009:146).

Test environment → adalah pengaturan di mana pengujian terjadi, termasuk tes, platform, infrastruktur tes, uji lab, dan fasilitas lainnya (Black, 2009:611)

Test result → seorang penguji harus mendokumentasikan hasil dari pengujiannya sehingga dapat diketahui hal apa yang telah tercapai ataupun yang tidak tercapai dalam sebuah hasil pengujian (Perry, 2006:463).

Bug → adalah suatu masalah yang hadir dalam sistem yang diuji yang menyebabkan gagal dalam memenuhi harapan atau biasa disebut juga kecacatan. Gejala dari bug adalah sebuah kegagalan (Black, 2009:601-602).

Test suite → adalah kumpulan test case yang dimaksudkan untuk digunakan untuk menguji program perangkat lunak untuk menunjukkan bahwa *perangkat lunak* memiliki beberapa set perilaku tertentu. Test suite sering berisi instruksi atau sasaran terperinci untuk setiap

kumpulan test case dan informasi tentang konfigurasi sistem yang akan digunakan selama pengujian. Sekelompok test case juga dapat berisi status atau langkah prasyarat, dan uraian pengujian berikutnya.² *Test suite* merupakan sebuah kerangka untuk melakukan eksekusi terhadap sekumpulan *test case*; suatu cara mengatur *test case*. Dalam sebuah *test suite*, *test case* dapat digabungkan untuk menjadi kondisi-kondisi pengujian yang unik. (Black, 2009: 611).

Test scenario → atau skenario pengujian adalah aktivitas pengujian perangkat lunak yang menggunakan skenario: cerita hipotetis untuk membantu penguji bekerja melalui masalah kompleks atau sistem pengujian. Tes skenario yang ideal adalah cerita yang kredibel, kompleks, menarik atau memotivasi yang hasilnya mudah dievaluasi³

Smoke testing → juga dikenal sebagai "*Build Verification Testing*", adalah jenis pengujian perangkat lunak yang terdiri dari serangkaian tes tidak lengkap yang bertujuan untuk memastikan bahwa fungsi yang paling penting berfungsi. Hasil dari smoke testing digunakan untuk memutuskan apakah sebuah build cukup stabil untuk dilanjutkan dengan pengujian lebih lanjut.

Istilah '*smoke testing*', muncul dari jenis pengujian perangkat keras, di mana perangkat lulus tes jika tidak terbakar (atau berasap/*smoked*) pertama kali dinyalakan.⁴

Verifikasi → merupakan istilah yang digunakan untuk mengetahui apakah produk atau perangkat lunak yang telah dibangun sudah benar. (Apakah produk yang dibangun sudah benar?)

Validasi → merupakan istilah yang digunakan untuk mengetahui apakah produk atau perangkat lunak yang telah dibangun sudah tepat. (Apakah produk yang dibangun sudah tepat?)

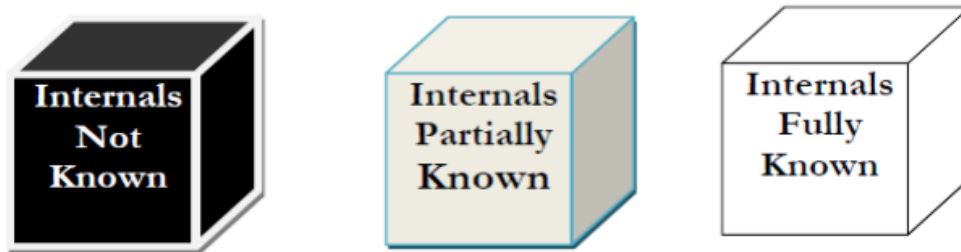
² <https://www.softwaretestinghelp.com/software-terms-complete-glossary>

³ Cem Kaner, *Introduction to Scenario Testing*, Florida Tech, June 2003.

⁴ <http://softwarefundamentals.com/smoke-testing>, Tanggal akses 1 Juni 2018

1.3 Metodologi Testing

Metodologi *testing* secara umum dibagi tiga berdasarkan *Box Testing* yaitu *black-box*, *white-box*, dan *grey-box testing*



Gambar 1.1 Perbandingan Box Testing ⁵

Berdasarkan Gambar 1.1, teknik pengujian tanpa memiliki pengetahuan tentang cara kerja interior aplikasi disebut pengujian *black-box*. Penguji tidak menyadari arsitektur sistem dan tidak memiliki akses ke kode sumber. Biasanya, ketika melakukan tes *black-box*, tester akan berinteraksi dengan antarmuka pengguna sistem dengan memberikan input dan memeriksa output tanpa mengetahui bagaimana dan di mana input bekerja. Detail perbedaan metodologi testing dapat dilihat pada Tabel 1.1

Tabel 1.1 Perbedaan Metodologi

No	Black Box Testing	Gray Box Testing	White Box Testing
1	Cara Kerja internal aplikasi tidak perlu diketahui	Pengetahuan tentang cara kerja internal diketahui	Tester memiliki pengetahuan penuh tentang cara kerja internal aplikasi
2	Dikenal sebagai pengujian kotak tertutup, pengujian yang didorong data, dan pengujian fungsional	Istilah lain adalah pengujian kotak samar, Penguji memiliki pengetahuan terbatas dari bagian dalam aplikasi	Juga dikenal sebagai pengujian kotak yang jelas, pengujian struktural atau pengujian berbasis kode
3	Dilakukan oleh	Dilakukan oleh pengguna	Biasanya dilakukan oleh

⁵ Acharya dkk, Bridge between Black Box and White Box – Gray Box Testing Technique, International Journal of Electronics and Computer Science Engineering, ISSN- 2277-1956.