

DevOps 2TIN Chapter 7

Integrated Testing



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be



Integrated testing

- Soorten testen
 - Unittesten
 - Integratie testen
 - Functionele testen
 - Niet functionele testen
 - Test automatisatie
 - Testing in pipelines

Wat is een Pipeline?

Definitie “deployment pipeline”:

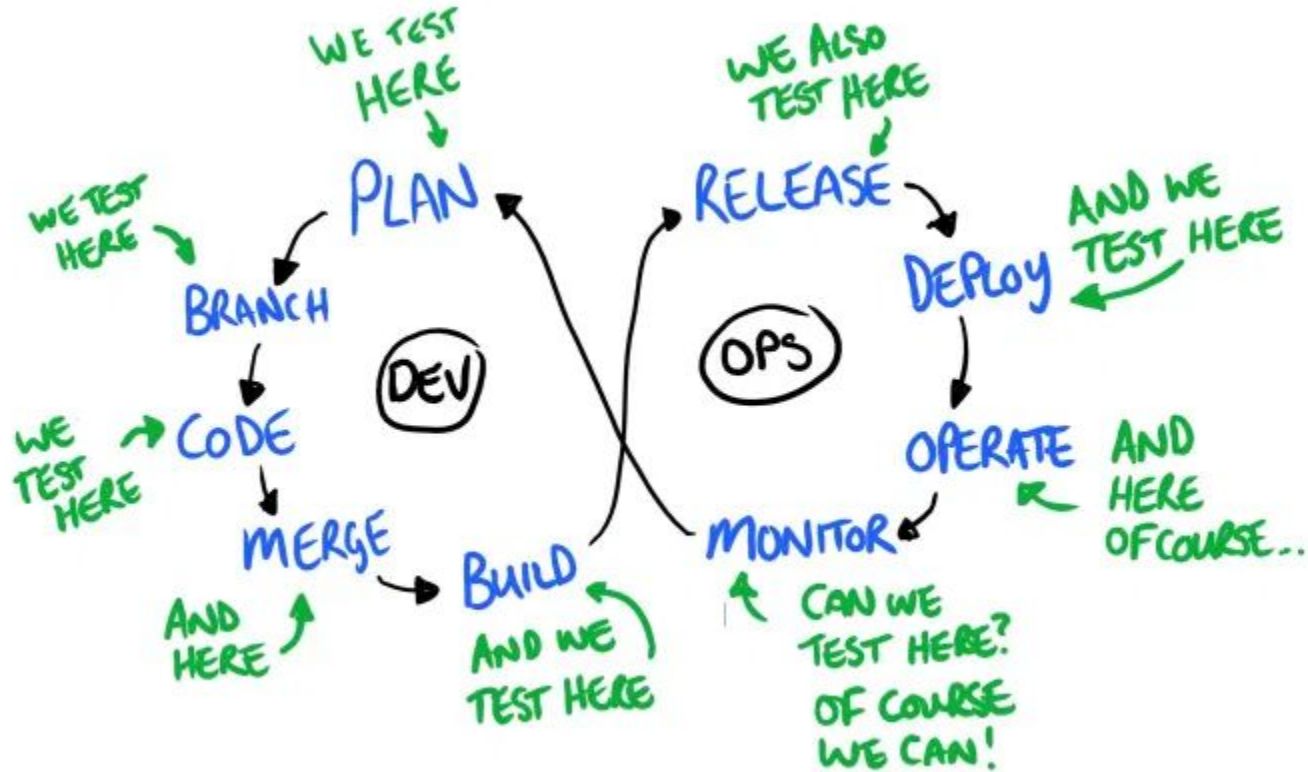
(first defined by Jez Humble and David Farley in their book *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*)

It ensures that all code checked in to version control is automatically built and tested in a production-like environment.

Sleutelwoorden:

- Alle Code
- Versiebeheer
- Automatisch gebouwd
- **Automatisch getest**
- Productie-waardige omgeving

Integrated testing



RECAP - Testing in de 3 ways

Test **ALLES** vanaf het moment dat het kan!

Test **automatisch** om Continuous Integration te bekomen

Definitie *continuous integration*:

- **A comprehensive and reliable set of automated tests that validate we are in a deployable state.**
- **A culture that “stops the entire production line” when our validation tests fail. (=digital andon cord)**

Ideal vs. Non-Ideal Testing Pyramids

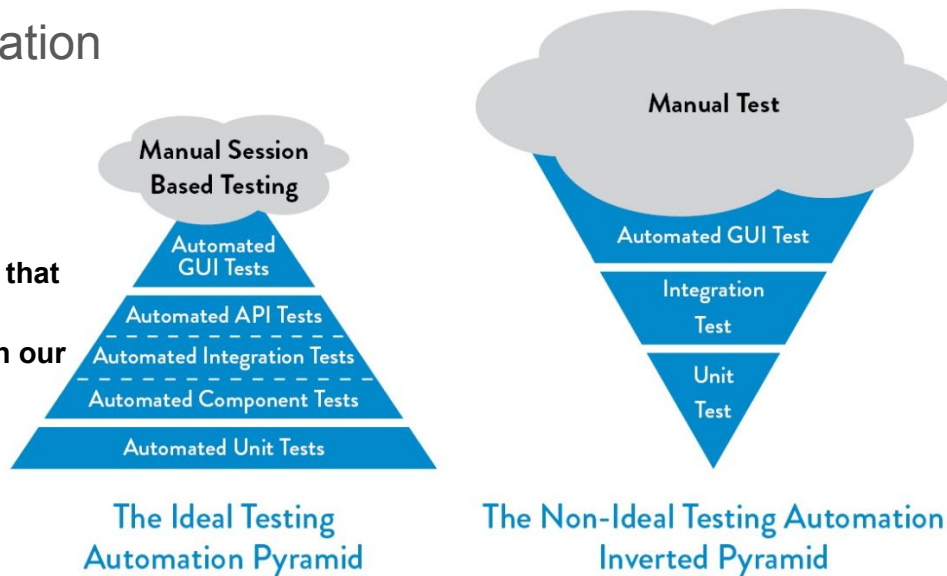


Figure 14: The ideal and non-ideal automated testing pyramids (Source: Martin Fowler, “TestPyramid.”)

Maar hoe gaan we daarmee om?

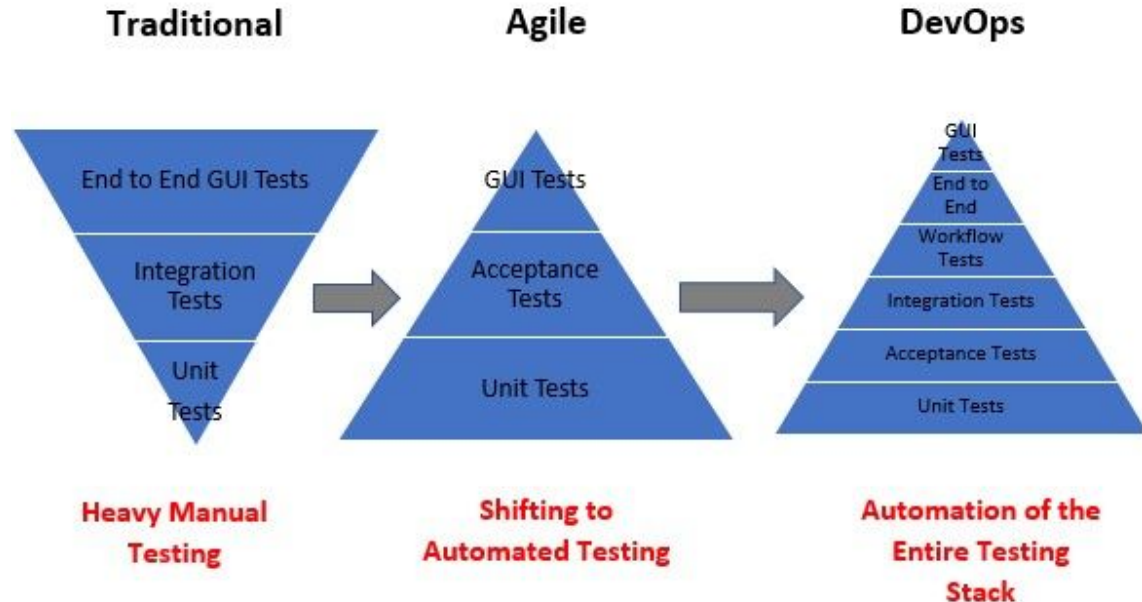
Overall testen is goed in devops-> Zo snel mogelijk feedback!

Maar, testen kost tijd, dus hoe lossen we dat op?

We LEREN en optimaliseren waar nodig.

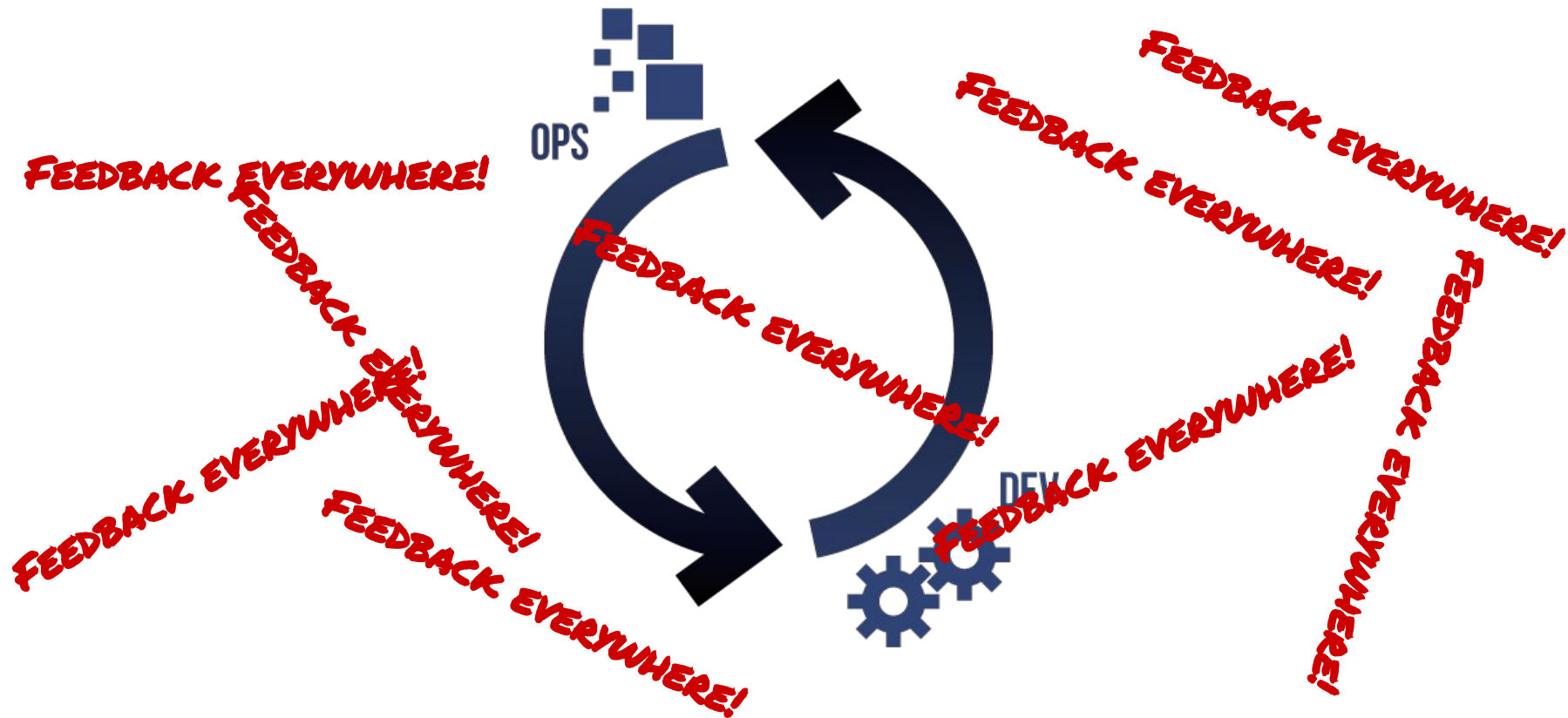
Begin gerust manueel, maar evolueer!

Flipping/Inverting the Testing Pyramid



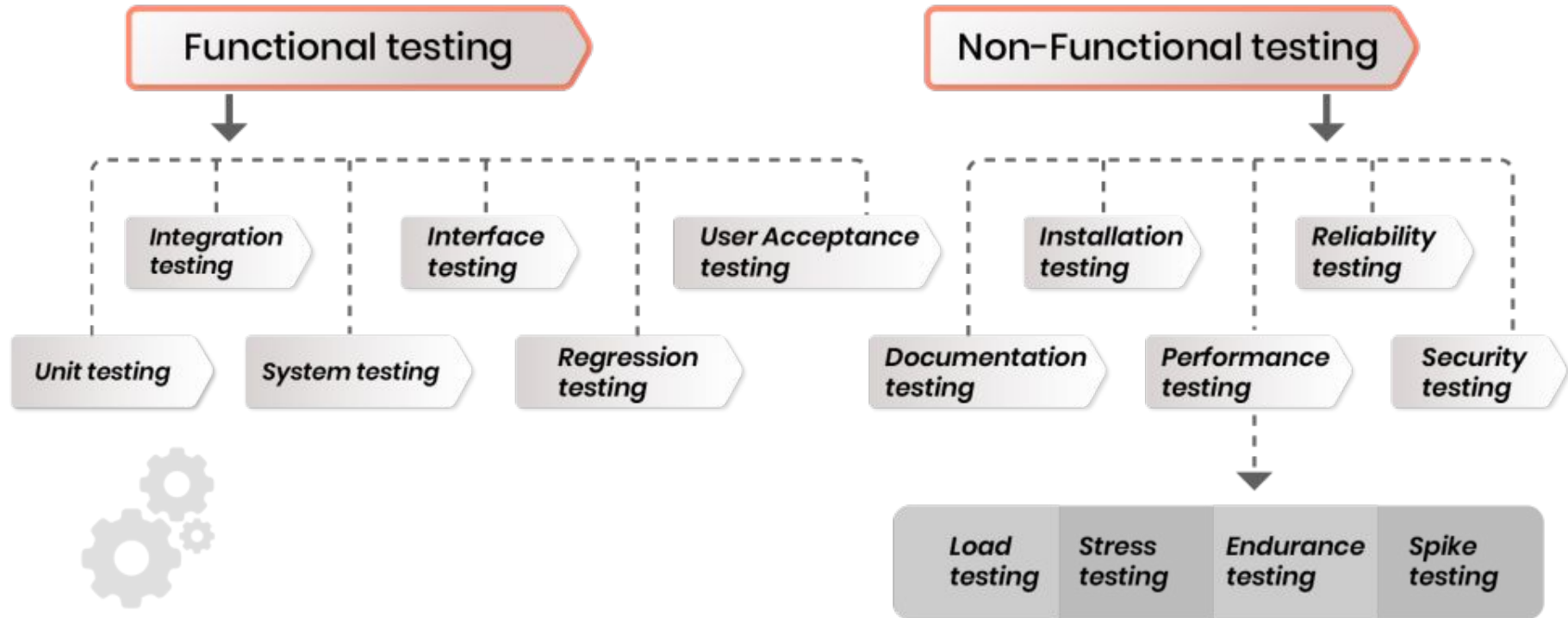
THE SECOND WAY

AMPLIFY FEEDBACK LOOPS



Soorten Testen

TYPES OF SOFTWARE TESTING



Soorten testen - Voorbeeld

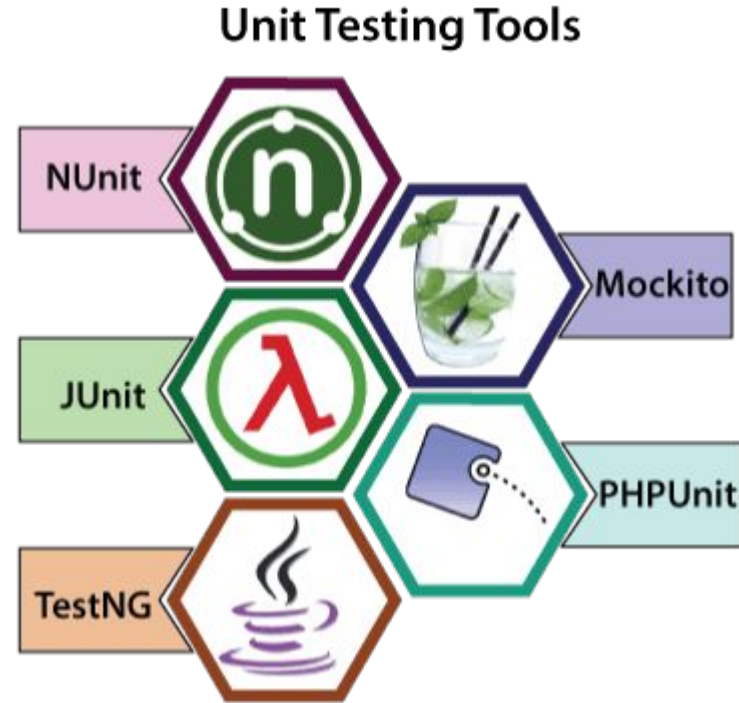
Voor een functionele **smartphone** zijn de belangrijkste onderdelen die nodig zijn "batterij" en "simkaart".

- **Unit testing** - de batterij wordt gecontroleerd op levensduur, capaciteit en andere parameters. Simkaart wordt gecontroleerd op activering.
- **Integration Testing** - batterij en simkaart zijn geïntegreerd, dwz samengesteld om de mobiele telefoon te starten.
Opgeslagen nummers op de simkaart kunnen uitgelezen worden door de telefoon.
De geactiveerde simkaart maakt het mogelijk om met de telefoon een (nood)nummer te bellen.
- **Functioneel testen** - de functionaliteit van de mobiele telefoon wordt gecontroleerd op basis van zijn functies en ook op batterijgebruik en simkaartfaciliteiten.

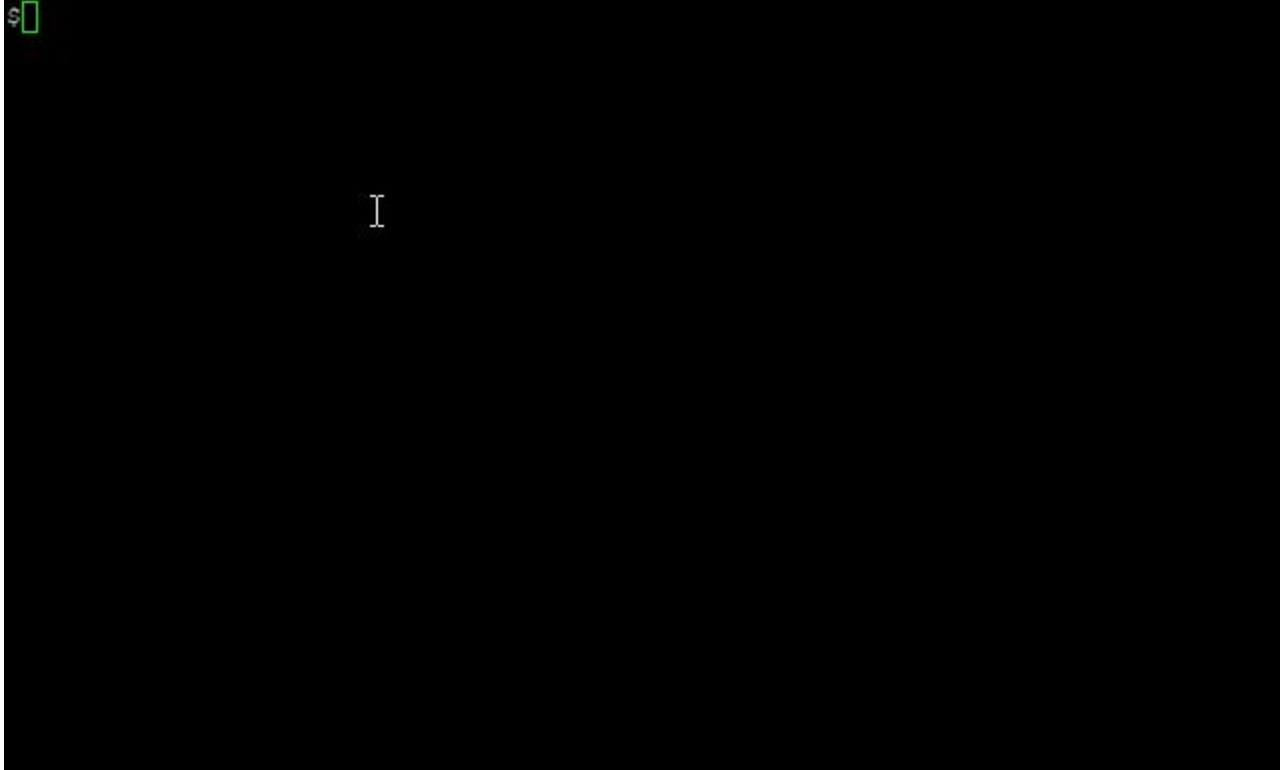
Unit testen

- Testen van software componenten los van Groter geheel
- Kwaliteitscontrole op code-niveau
- Bewijs dat code doet wat het zou moeten Doen

<https://www.javatpoint.com/unit-testing-tools>



Unit testen



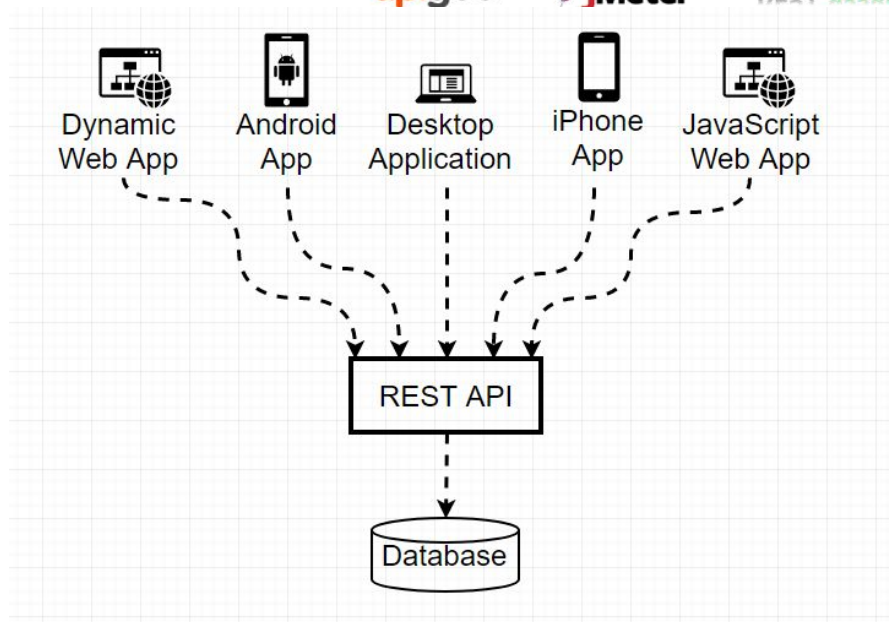
Integratie testen

API Testing Tools

Integraties met andere software



-> Meestal via API's



Integratie testen

Opbouw API (In java)

- Programma code
- Jetty webserver

```
1 <web-app
2   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5     http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6   version="3.1">
7
8   <servlet>
9     <servlet-name>PersonServlet</servlet-name>
10    <servlet-class>PersonServlet</servlet-class>
11  </servlet>
12
13  <servlet-mapping>
14    <servlet-name>PersonServlet</servlet-name>
15    <url-pattern>/people/*</url-pattern>
16  </servlet-mapping>
17
18 </web-app>
```

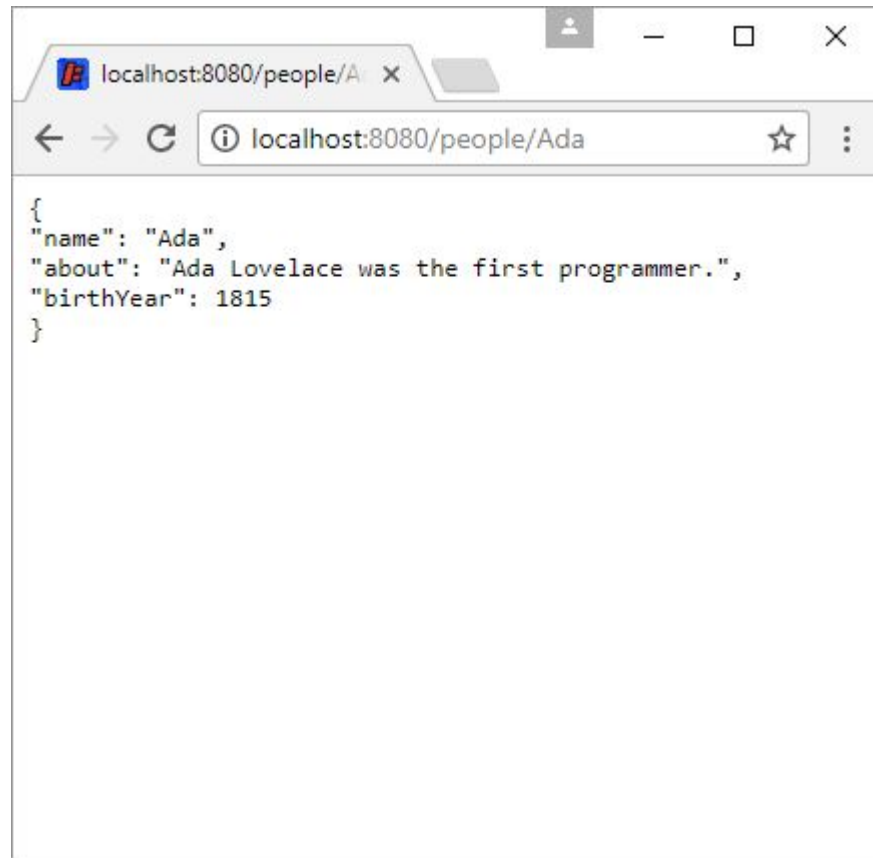
```
1 import java.io.IOException;
2
3 import javax.servlet.ServletException;
4 import javax.servlet.http.HttpServlet;
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7
8 import org.json.JSONObject;
9
10 public class PersonServlet extends HttpServlet {
11
12     @Override
13     public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
14
15         String requestUrl = request.getRequestURI();
16         String name = requestUrl.substring("/people/".length());
17
18         Person person = DataStore.getInstance().getPerson(name);
19
20         if(person != null){
21             String json = "{\n";
22             json += "\"name\": " + JSONObject.quote(person.getName()) + ",\n";
23             json += "\"about\": " + JSONObject.quote(person.getAbout()) + ",\n";
24             json += "\"birthYear\": " + person.getBirthYear() + "\n";
25             json += "}";
26             response.getOutputStream().println(json);
27         }
28         else{
29             //That person wasn't found, so return an empty JSON object. We could also return an error.
30             response.getOutputStream().println("{}");
31         }
32     }
33
34
35
36     @Override
37     public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
38
39         String name = request.getParameter("name");
40         String about = request.getParameter("about");
41         int birthYear = Integer.parseInt(request.getParameter("birthYear"));
42
43         DataStore.getInstance().putPerson(new Person(name, about, birthYear, password));
44     }
45 }
```

Integratie testen

Output in webbrowser:

Hoe testen we dit?

- Manueel met Browser of wget (CLI)?
- Automatisch!



Integratie testen

- Postman test collection
 - Bundeling van requests op url's met headers / content / body /
 - Automatische inventarisatie van de response
 - Mogelijk om de volledige suite in één keer te runnen
 - Eenvoudige import / export features

<https://learning.postman.com/docs/getting-started/creating-the-first-collection/>

Integratie testen



Postman interface showing a REST client request setup and response.

Request Details:

- Method: GET
- URL: http://localhost:8080/people/ada
- Environment: No Environment

Query Params:

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response:

```
{
  "name": "Ada",
  "about": "Ada Lovelace was the first programmer.",
  "birthYear": 1815
}
```

Integratie testen



Postman

File Edit View Help

+ New Import Runner

My Workspace Invite

No Environment

Launchpad GET get GET https://w... GET http://loc... POST Devops... X

History Collections APIs

+ New Collection Trash

test 2 requests

GET get

POST Devops-POST-TestCase1

Devops-POST-TestCase1

POST https://postman-echo.com/post?PXL=Digital&Course=DevOps

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> PXL	Digital	Departement		
<input checked="" type="checkbox"/> Course	DevOps	Vak		
Key	Value	Description		

Body Cookies (1) Headers (7) Test Results

Status: 200 OK Time: 398 ms Size: 993 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "args": {
3     "PXL": "Digital",
4     "Course": "DevOps"
5   },
6   "data": "Deze data zit in de body van de boodschap die ik upload",
7   "files": {},
8   "form": {},
9   "headers": {
10    "x-forwarded-proto": "https",
11    "x-forwarded-port": "443",
12    "host": "postman-echo.com",
13    "x-amzn-trace-id": "Root=1-5fb433a2-4073656864c79bde37560844",
14    "content-length": "55",
```

Integratie testen

=> Suites zijn uitbreidbaar met test scripts

- Diepere controle op de gemaakte requests en responses
 - Welke statuscode krijg ik terug
 - Welke waardes krijg ik terug, zit waarde x daarin
 - Gebruik van variabelen doorheen de test suite
- Pass / fail net zoals bij andere soorten testen

<https://learning.postman.com/docs/writing-scripts/test-scripts/>

<https://learning.postman.com/docs/writing-scripts/script-references/test-examples/>

Integratie testen

Postman ECHO No Environment, just now

Run SummaryRun AgainNew

All TestsPassed (4)Failed (1)

Iteration 1

GET GET Request postman-echo.com/get?record={{name_detail}}

200 OK227 ms860 B

PassStatus code is 200

PassStatus code is 200

POST POST Request postman-echo.com/post

200 OK226 ms880 B

PassStatus code is 200

PassStatus code is 200

FailResponse time is less than 200ms | AssertionError: expected 226 to be below 200

Desktop AgentRunnerTrash?

Integratie testen - Postman test scripts

- Testen op response code

The screenshot shows the Postman interface for a GET request to `https://postman-echo.com/get?foo1=bar1&foo2=bar2`. The 'Tests' tab is selected, displaying a JavaScript test script:

```
1 pm.test("Status test", function () {  
2   pm.response.to.have.status(200);  
3 });
```

The right sidebar contains a 'Cookies' section and a 'SNIPPETS' section with the following items:

- Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)
- Get an environment variable
- Get a global variable
- Get a variable
- Set an environment variable
- Set a global variable
- Clear an environment variable

Integratie testen - Postman test scripts

- Testen op response code + body



Integratie testen - Postman test scripts

- Testen op inhoud response



```
pm.test("Person is Jane", () => {  
  const responseJson = pm.response.json();  
  pm.expect(responseJson.name).to.eql("Jane");  
  pm.expect(responseJson.age).to.eql(23);  
});
```


- Gebruik van variabelen

EDIT COLLECTION

Name

Examen 2de zit

Description

Authorization

Pre-request Scripts

Tests

Variables

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	id	5ff04333f8d4f34c5c0ab82f	5ff04333f8d4f34c5c0ab82f			
<input checked="" type="checkbox"/>	addedID		61956d0743ade59430bb7c7f			
	Add a new variable					

Integratie testen - Postman test scripts

- Gebruik van variabelen

- Waarde opslaan:

```
const responseJson = pm.response.json();  
pm.collectionVariables.set("addedID", responseJson._id);
```

- Waarde opvragen

```
pm.test("Body contains _id with newly added", () => {  
  pm.expect(pm.response.text()).to.include(pm.collectionVariables.get("addedID"));  
});
```

Integratie testen - Postman CLI

- Postman = grafische applicatie
 - Niet integreerbaar in CI pipelines
- Newman = CLI applicatie
 - Integreerbaar in CI pipelines als npm package
 - Kan geëxporteerde postman test suites / scripts uitvoeren
 -

<https://learning.postman.com/docs/running-collections/using-newman-cli/command-line-integration-with-newman/>

Functionele (E2E) testen

- Testen van UI
 - Niet meer op code niveau
- Applicatie moet *functioneel* zijn
- Applicatie wordt getest als (sub)geheel, niet meer als losse code-componenten

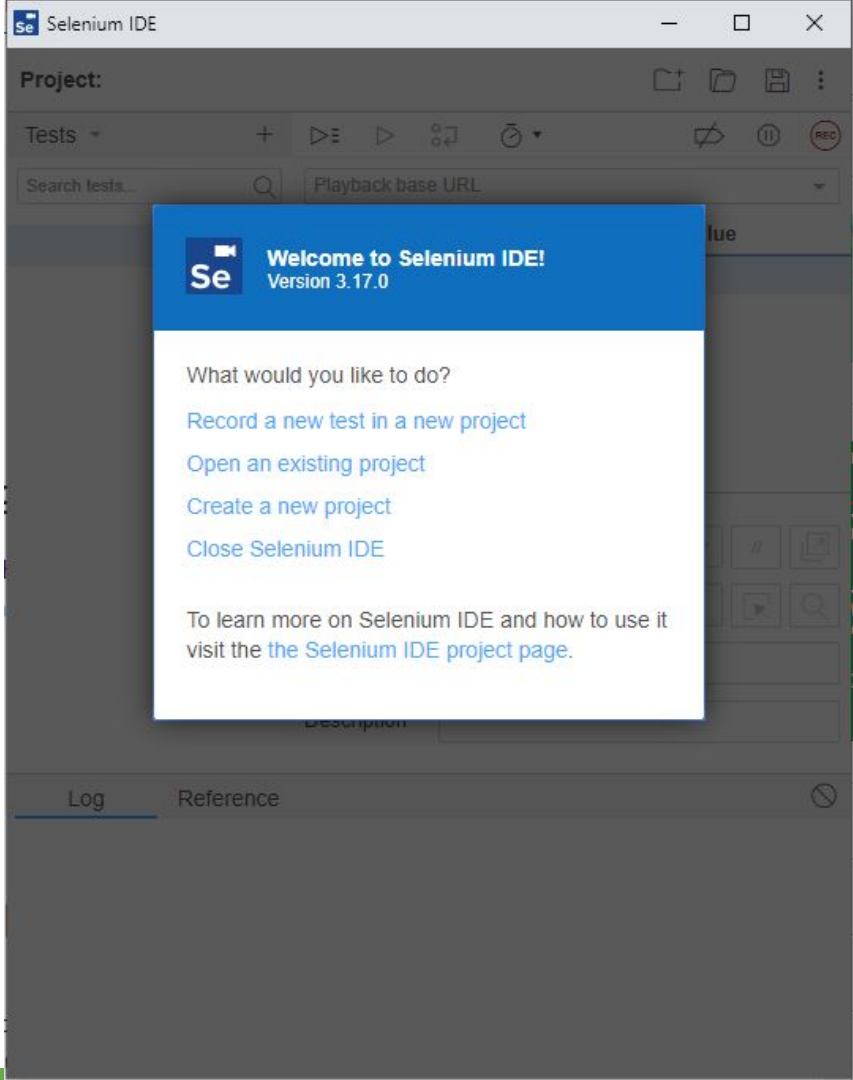


Functionele (E2E) testen

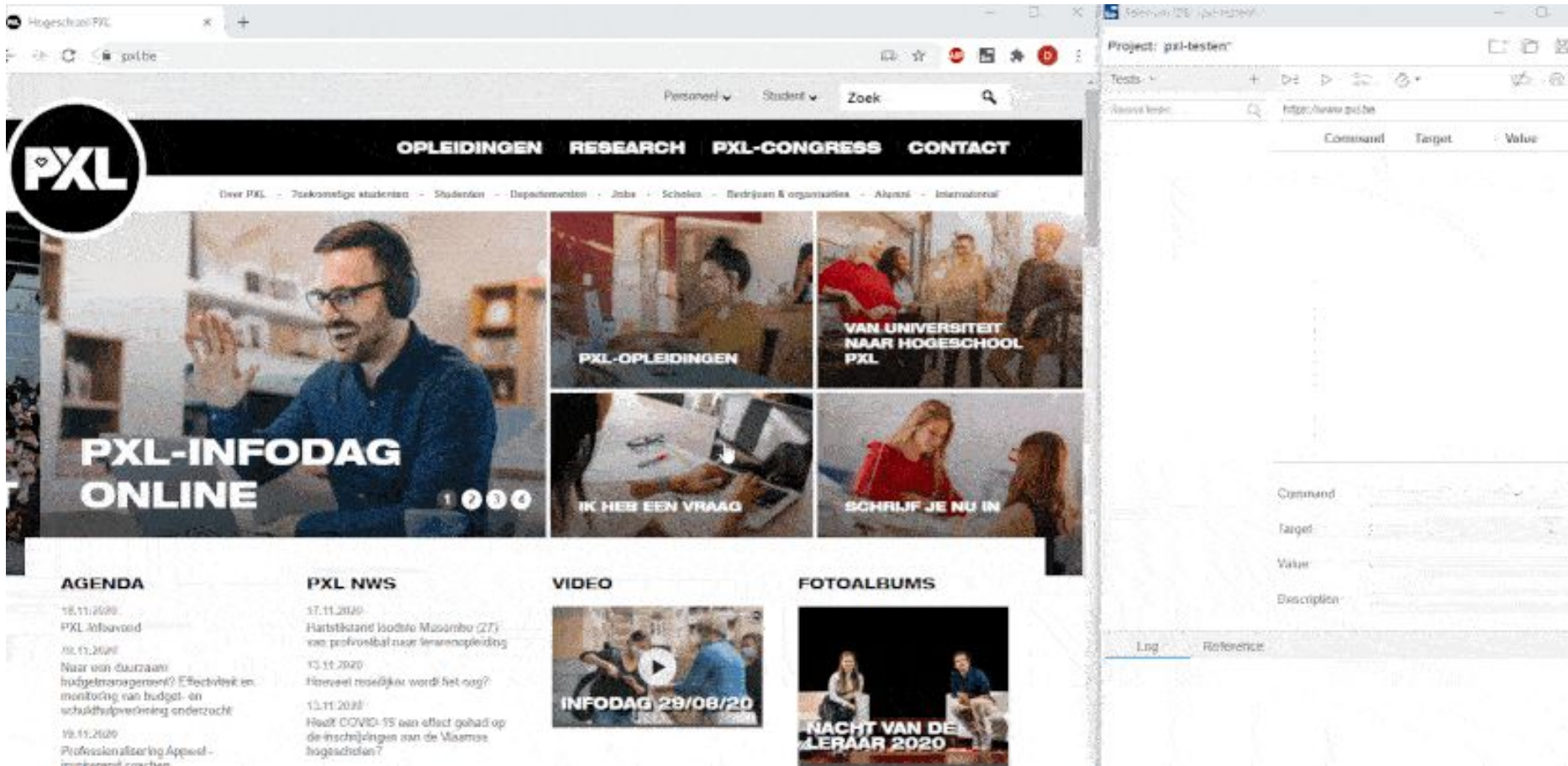
- Selenium IDE
 - Chrome en/of firefox plugin
- Lightweight add-on om E2E testen te schrijven
- .side files

```
pxl-testen - Kladblok
Bestand Bewerken Opmaak Beeld Help

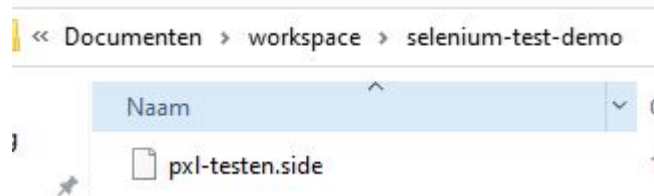
{
  "target": "/",
  "targets": [],
  "value": ""
}, {
  "id": "8c9e4ca0-9532-4689-ac3b-9fd029959c6a",
  "comment": "",
  "command": "setWindowSize",
  "target": "1187x867",
  "targets": [],
  "value": ""
}, {
  "id": "65e4c567-9f85-4828-a19c-cd1fb3d1231a",
  "comment": "",
  "command": "click",
  "target": "css=.nav_list > li:nth-child(1) > a",
  "targets": [
    ["css=.nav_list > li:nth-child(1) > a", "css:finder"],
    ["xpath=//a[contains(text(),'Opleidingen')]","xpath:link"],
    ["xpath=//div[@id='nav1_container']/div/div[2]/ul/li/a","xpath:idRelative"],
    ["xpath=//a[contains(@href, '/Opleidingen.html')]","xpath:href"],
    ["xpath=//div[2]/div/div/div[2]/ul/li/a","xpath:position"]
  ],
  "value": ""
}
```



Functionele (E2E) testen



Selenium IDE



The screenshot shows a Notepad++ window titled 'pxl-testen - Kladblok'. The menu bar includes 'Bestand', 'Bewerken', 'Opmaak', 'Beeld', and 'Help'. The text area contains Selenium IDE test commands in JSON format. The commands are: 'setWindowSize' with target '1187x867', and 'click' with target 'css=.nav_list > li:nth-child(1) > a'. The 'click' command has a list of targets including CSS selectors, XPath expressions, and positions. The status bar at the bottom right indicates 'Ln 1, Col 1'.

```
"target": "/",
"targets": [],
"value": ""
}, {
  "id": "8c9e4ca0-9532-4689-ac3b-9fd029959c6a",
  "comment": "",
  "command": "setWindowSize",
  "target": "1187x867",
  "targets": [],
  "value": ""
}, {
  "id": "65e4c567-9f85-4828-a19c-cd1fb3d1231a",
  "comment": "",
  "command": "click",
  "target": "css=.nav_list > li:nth-child(1) > a",
  "targets": [
    ["css=.nav_list > li:nth-child(1) > a", "css:finder"],
    ["xpath=//a[contains(text(),'Opleidingen')]", "xpath:link"],
    ["xpath=//div[@id='nav1_container']/div/div[2]/ul/li/a", "xpath:idRelative"],
    ["xpath=//a[contains(@href, '/Opleidingen.html')]"]],
    ["xpath=//div[2]/div/div/div[2]/ul/li/a", "xpath:position"]
  ],
  "value": ""
}, {
  "id": "f233f1fb-fd19-4599-afcb-a56a3ab9d829",
  "comment": "",
  "command": "click",
  "target": "css=.filter_title1",
  "targets": [
    ["css=.filter_title1", "css:finder"],
    ["xpath=//div[@id='content']/div/div/span", "xpath:idRelative"],
    ["xpath=//div[2]/div/div/div[2]/ul/li/a", "xpath:position"]
  ],
  "value": ""
}
```


Selenium IDE

- Custom commands
 - Controle of elementen (niet) aanwezig zijn
 - Waar die aanwezig moeten zijn
 - ...
- Testen falen of slagen
- Bundeling van testen = test suite

Project: pxl-testen

Tests +

Search tests...

pxl-digital

https://www.pxl.be

	Command	Target	Value
1	open	/	
2	set window size	1187x867	
3	click	css=.nav_list > li:nth-child(1) > a	
4	click	css=.filter_title1	
5	click	name=option2	
6	click	css=.filter_title1	

Command: click

Target: css=.filter_title1

Value:

Description:

Log Reference

1.	open on / OK	12:28:39
2.	setWindowSize on 1187x867 OK	12:28:39
3.	click on css=.nav_list > li:nth-child(1) > a OK	12:28:39
4.	click on css=.filter_title1 OK	12:28:42
5.	click on name=option2 OK	12:28:44
6.	click on css=.filter_title1 OK	12:28:44
'pxl-digital' completed successfully		12:28:44

Selenium IDE

The image shows a Selenium IDE window with a project named 'pxl-testen'. The test suite 'pxl-zoek-algemeen' contains five steps:

Step	Command	Target	Value
1	open	/	
2	setWindowRect	1187x807	
3	click	id=q	
4	type	id=q	labafreek
5	click	css=somplekstend	

The bottom panel shows the execution log:

Log	Reference	Time
Running 'pxl-zoek-algemeen'		12:58:56
1. open on / OK		12:58:58
2. setWindowRect on 1187x807 OK		12:58:59
3. click on id=q OK		12:59:00
4. type on id=q with value labafreek OK		12:59:02
5. click on css=somplekstend OK		12:59:02
'pxl-zoek-algemeen' completed successfully		12:59:02

Continuous quality in een pipeline

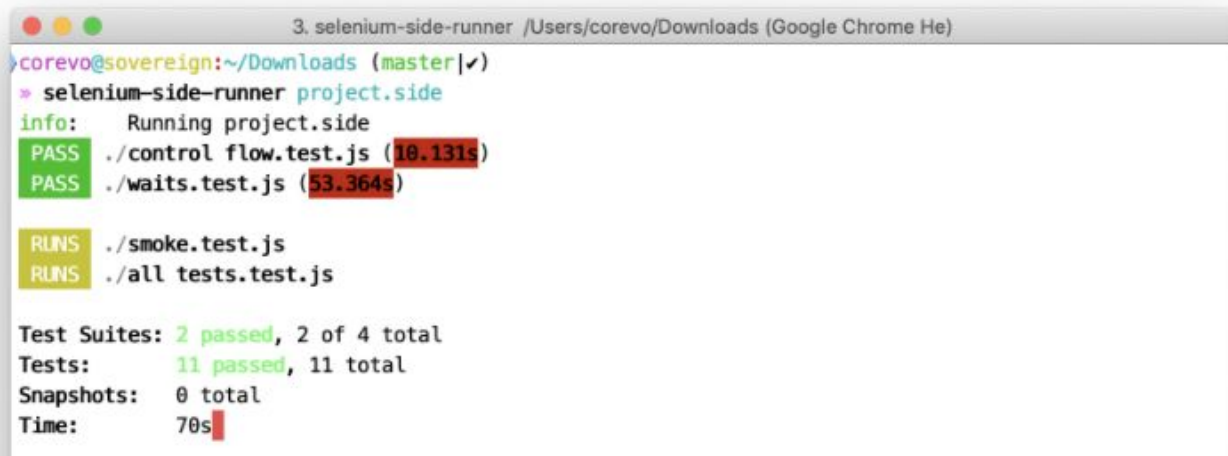
- Testen runnen vanuit een pipeline
 - CLI omgevingen
 - Integratie met externe systemen (APIs?)
- Functionele testen
 - Selenium CLI

E2E testen in een pipeline

Command-line Runner

You can now run all of your Selenium IDE tests on any browser, in parallel, and on a Grid without needing to write any code.

There's just the small matter of installing the Selenium IDE command line runner, getting the necessary browser drivers (if running your tests locally), and launching the runner from a command prompt with the options you want.

A terminal window titled '3. selenium-side-runner /Users/corevo/Downloads (Google Chrome He)' shows the execution of 'selenium-side-runner project.side'. The output includes status messages for individual test files: 'PASS ./control flow.test.js (10.131s)' and 'PASS ./waits.test.js (53.364s)', followed by 'RUNS ./smoke.test.js' and 'RUNS ./all tests.test.js'. A summary at the bottom states: 'Test Suites: 2 passed, 2 of 4 total', 'Tests: 11 passed, 11 total', 'Snapshots: 0 total', and 'Time: 70s'.

```
3. selenium-side-runner /Users/corevo/Downloads (Google Chrome He)
corevo@sovereign:~/Downloads (master|✓)
> selenium-side-runner project.side
info: Running project.side
PASS ./control flow.test.js (10.131s)
PASS ./waits.test.js (53.364s)

RUNS ./smoke.test.js
RUNS ./all tests.test.js

Test Suites: 2 passed, 2 of 4 total
Tests: 11 passed, 11 total
Snapshots: 0 total
Time: 70s
```

<https://www.selenium.dev/selenium-ide/docs/en/introduction/command-line-runner>

E2E testen in een pipeline - benodigdheden

Prerequisites

The following dependencies are needed for the command line runner to work:

- `node` (the Node.js programming language) version `8` or `10`
- `npm` (the NodeJS package manager) which typically gets installed with `node`
- `selenium-side-runner` (the Selenium IDE command line runner)
- and the browser driver we want to use (more on that in the next section)

```
> brew install node  
> npm install -g selenium-side-runner
```

E2E testen in een pipeline - benodigdheden

- Browser nodig - Op een server? - Headless!
- Chrome webdriver

<https://chromedriver.chromium.org/>

Continuous quality in Jenkins

- Manuele tool installatie!

Install Chromedriver before use!

```
wget https://chromedriver.storage.googleapis.com/86.0.4240.22/chromedriver_linux64.zip
unzip chromedriver_linux64.zip
sudo mv ./chromedriver /usr/local/bin/chromedriver
```

Install Chrome before use!

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo dpkg -i google-chrome-stable_current_amd64.deb
```

<https://chromedriver.chromium.org/>

Continuous quality in Jenkins

- Global tool configuration!

NodeJS

NodeJS installations

Add NodeJS

NodeJS

Name

node-selenium

☒ Install automatically

Install from nodejs.org

Version

NodeJS 10.19.0

☐ Force 32bit architecture

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

Global npm packages to install

selenium-side-runner

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

72

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

Idem voor newman



Continuous quality in Jenkins

- Bouwen van de pipeline
 - Binnenhalen .side testfiles
 - Selenium side runner CLI
 - Headless chrome
 - Genereren van rapporten


```
stage ('fetch tests'){
  steps{
    git 'https://github.com/d-ries/test-selenium.git'
  }
}
stage('Run automated tests') {
  steps {
    nodejs('node-selenium') {
      sh "npm -v"
      sh "selenium-side-runner -c \"goog:chromeOptions.args=[disable-infobars, headless] browserName=chrome\" --output-directory=seleniumReports --output-format=junit *.side"
      sh "ls -lah ."
    }
  }
}
```


Continious quality in Jenkins


```
selenium-side-runner -c \"goog:chromeOptions.args=[disable-infobars, headless] browserName=chrome\" --output-directory=seleniumReports --output-format=junit *.side
```


- Selenium-side-runner CLI omgeving
- **-c configuration:** geeft aan dat we de testen draaien in een headless chrome omgeving
- Configuratie van de output van de testresultaten
 - In de map seleniumReports
 - In JUNIT formaat
- Uit te voeren test scripts (.side files)

Continuous quality in Jenkins


 **Jenkins**


 **1**


 **Dries Swinnen**


 **log out**


Dashboard > **integrated-testing-selenium** > **#10**


 **Back to Project**


 **Status**


 **Changes**


 **Console Output**


 **Edit Build Information**


 **Delete build '#10'**


 **Git Build Data**


 **No Tags**

 **Restart from Stage**


 **Replay**


 **Pipeline Steps**


 **Workspaces**


 **Previous Build**



 Next Build


 **Build #10 (Oct 28, 2020 1:58:35 PM)**




 **add description** Started 19 days ago
Took 7.1 sec

 **Build Artifacts**

 **calculator-app-selenium.xml** 379 B  **view**

 Started by user [Dries Swinnen](#)

 **git**

Revision: df6e45fcc8cf0705ad8248a59929a81f9f09f4cf

- refs/remotes/origin/master

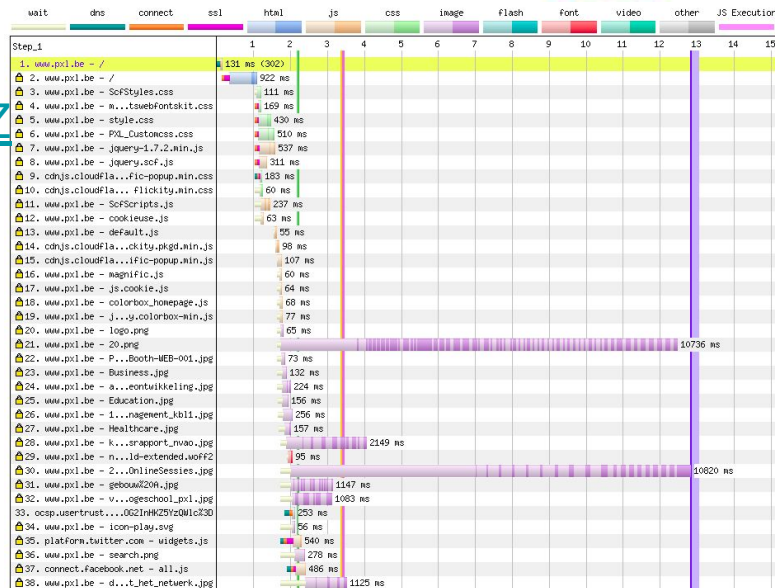
Non-functional testen

Performance testing tools

Front-end testing web apps:

Lighthouse voorbeeld: <https://bit.ly/3kFrfNy>

Webpagetest voorbeeld: <https://bit.ly/2K5XHvZ>



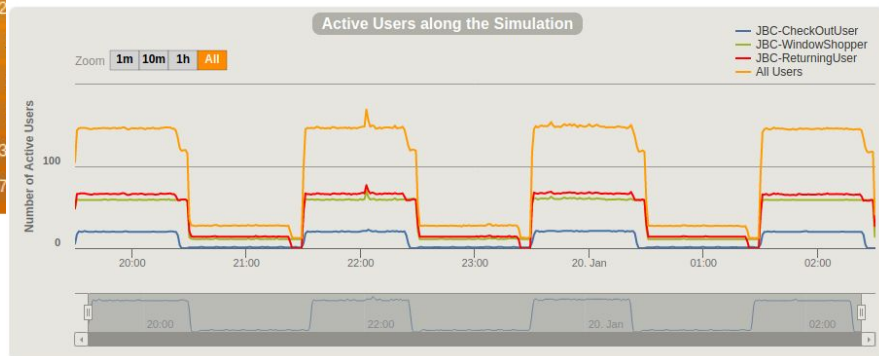
Non-functional testen



Performance testing: End-to-end Enterprise testing

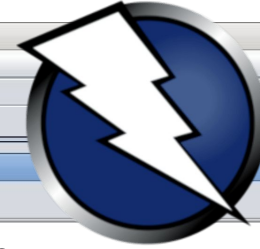
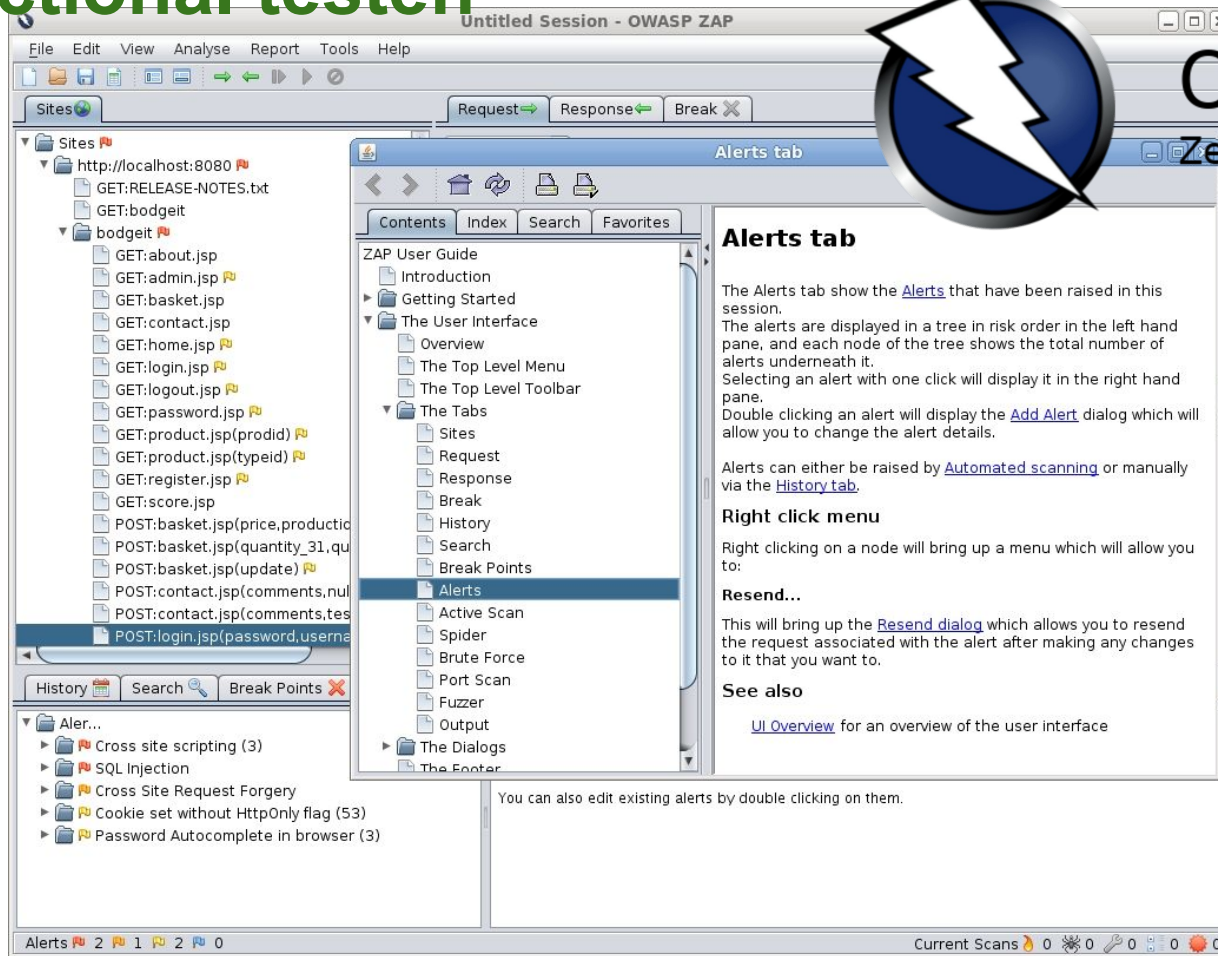


STATISTICS Expand all groups Collapse all groups													
Requests ^	Executions				Req/s ^	Response Time (ms)							
	Total ^	OK ^	KO ^	% KO ^		Min ^	50th pct ^	75th pct ^	95th pct ^	99th pct ^	Max ^	Mean ^	Std Dev ^
Global Information	627257	627209	48	0%	24.864	62	151	177	344	1060	42140	196	314
▶ Homepage	94750	94750	0	0%	3.756	112	151	179	321	831	18219	190	249
▶ Login	36800	36800	0	0%	1.459	460	835	1199	1997	2878	26238	1030	637
▶ ProductDetail	149950	149949	1	0%	5.944	106	160	186	336	812	20913	194	204
▶ Register	2750	2750	0	0%	0.109	340	534	782	1528	2100	18219	190	249
▶ Category	94750	94748	2	0%	3.756	99	150	177	356	1060	42140	196	314
▶ Search	94750	94750	0	0%	3.756	98	141	169	299	831	18219	190	249
▶ StaticPage	92000	91998	2	0%	3.647	94	134	154	278	812	20913	194	204
▶ Cart	2750	2750	0	0%	0.109	751	1020	1264	2310	2878	26238	1030	637
▶ Checkout	2750	2707	43	2%	0.109	2612	3417	3624	4780	5100	18219	190	249



Non-functional testen

Security testing tools



OWASP
Zed Attack Proxy

Non-functional testen

Security testing tools

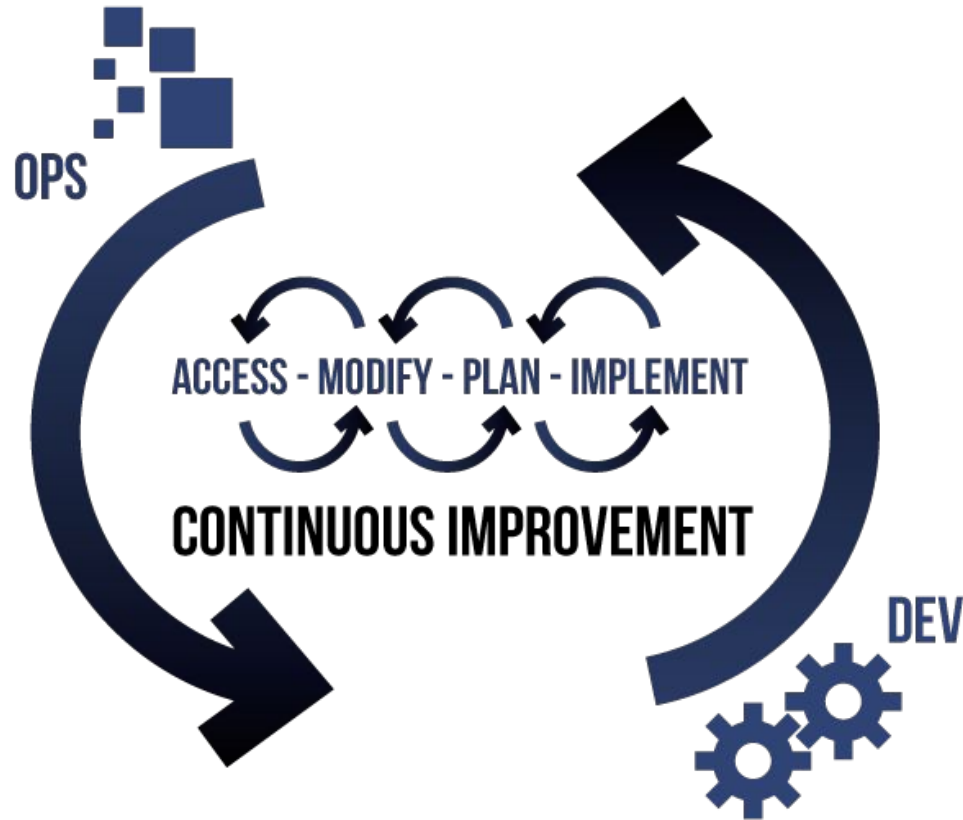
The screenshot displays the 'Scan Results' window of Acunetix. It shows a scan of 'http://testphp.vulnweb.com:80/' with a status of 'Finished (213 alerts)'. A tree view on the left lists various vulnerabilities, including 'Blind SQL Injection (34)', 'CRLF injection/HTTP response splitting (verified)', 'Cross site scripting (2)', 'Cross site scripting (verified) (34)', 'Directory traversal (verified) (2)', 'HTTP parameter pollution (2)', 'Script source code disclosure (1)', 'Server side request forgery (2)', 'SQL injection (4)', and 'SQL injection (verified) (38)'. The 'SQL injection (verified) (38)' category is expanded, showing sub-items like '/AJAX/infoartist.php (1)', '/AJAX/infocateg.php (1)', '/AJAX/infotitle.php (1)', '/artists.php (2)', and '/cart.php (5)'. The '/cart.php (5)' item is further expanded to show 'addcart (3)', which is then expanded to show 'variant 1', 'variant 2', and 'variant 3'. Other items like 'del (1)', 'login (1)', '/guestbook.php (1)', and '/listproducts.php (4)' are also visible.

www.SoftwareTestingHelp.com

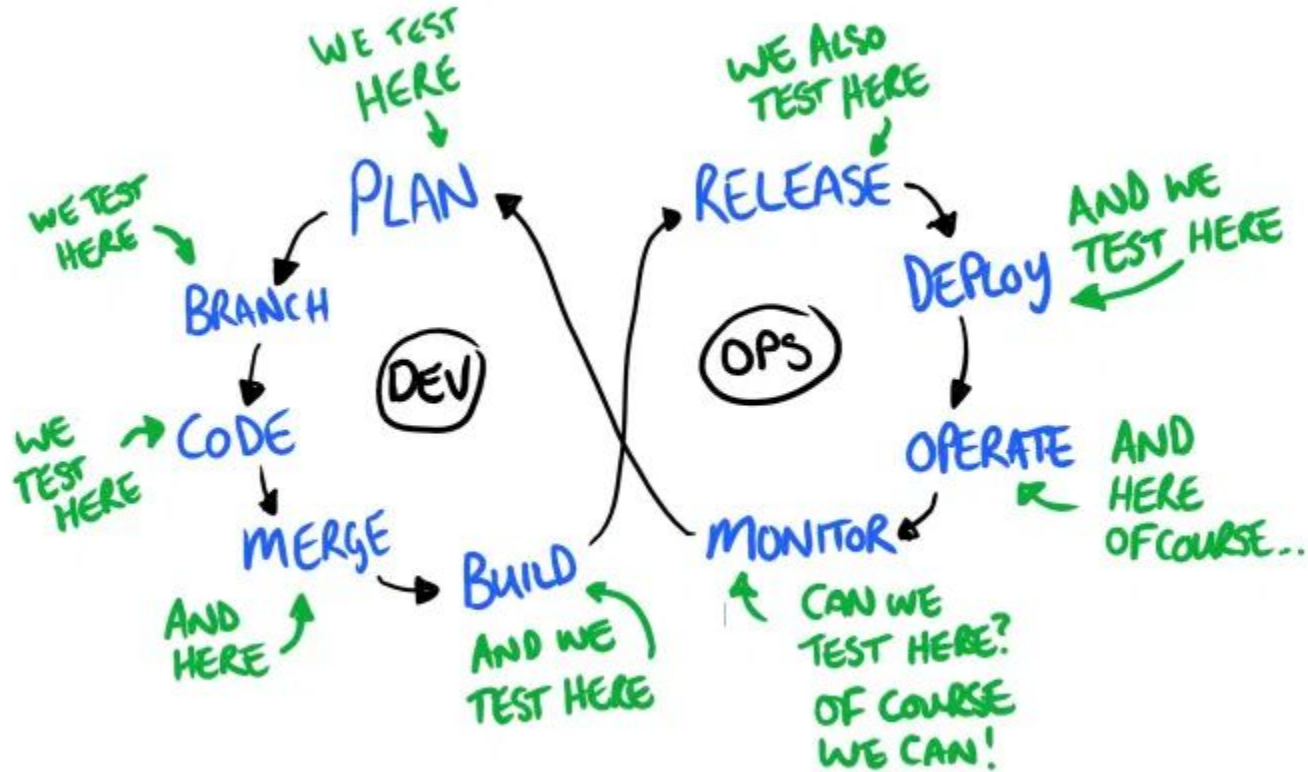
The screenshot shows the 'Vulnerability description' for a verified SQL injection. The title is 'SQL injection (verified)'. The description states: 'This script is possibly vulnerable to SQL Injection attacks. SQL injection is a vulnerability that allows an attacker to alter back-end SQL statements by manipulating the injection occurs when web applications accept user input that is directly placed into a SQL statement and does dangerous characters. This is one of the most common application layer attacks currently being used on the Internet. Despite the fact that it is easy to protect against, there is a large number of web applications vulnerable. This vulnerability affects /cart.php. Discovered by: Scripting (Sql_injection.script). AcuSensor TECHNOLOGY. Vulnerability details: Source file: /hj/var/www/cart.php line: 81. Additional details: SQL query: SELECT * FROM carts WHERE cart_id='10ebceb64152145d987c385c96080bea' AND item=1ACUSTART''8JFMGACUEND. "mysql_query" was called. Attack details: URL encoded POST input addcart was set to 1ACUSTART''8JFMGACUEND. Actions: View HTTP headers, View HTML response, Launch the attack with HTTP Editor, Retest alert(s), Mark this alert as a false positive. The impact of this vulnerability: An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity and/or expose sensitive information. Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system compromise. It may be possible to not only manipulate existing queries, but to UNION in arbitrary data, use sub additional queries. In some cases, it may be possible to read in or write out to files, or to execute shell commands on the underlying operating system.

THE THIRD WAY

LEARN & EXPERIMENT CONTINUOUSLY



Integrated testing





PLURALSIGHT

Title: [Creating Automated Browser Tests with Selenium in C#](#)

Detailed course on writing durable browser tests in c#. [4h04mins]

Title: [Automated Web Testing with Selenium and WebDriver Using Java](#)

Detailed course on writing durable browser tests in Java. [3h14mins]

Title: [Postman fundamentals](#)

Introduction to all the features of the postman tool. [2h38mins]



Assignments