AUTOMATISATION DES TESTS FONCTIONNELS

Frameworks & Bonnes Pratiques

LIVRE BLANC CO-ÉCRIT ET PUBLIÉ PAR

All4Test

CHRYSOCODE

Licence

Date de publication : 28 août 2020

Table des matières

Objectifs

Le livre que nous avons ici le plaisir de partager avec vous est un livre évolutif, dédié aux **frameworks d'automatisation de test**, c'est-à-dire aux *cadres de travail méthodologiques et technologiques pour mener des projets de développement de test automatisé*. Nous vous proposons dans cette version de découvrir quelques types de framework d'automatisation de test, et de nous suivre pour profiter des prochaines parutions, qui chacune ajouteront un nouveau chapitre spécifique de l'un des nombreux frameworks de test.

Version	Frameworks de test	Date
0.1	● Framework par script linéaire, avec Selenium IDE ● Framework de test piloté par mots d'action (*Keyword-Driven Testing*), avec Robot Framework ● Framework de test fondé sur des composants (*Component-Based Testing*), avec le *Page Object Model* et Selenium WebDriver	28/08/2020
0.2	Nouveautés : ● Framework de test piloté par les comportements (*Behavior-Driven Development* (BDD)), avec Robot Framework Mises à jour : ● Implémentation du framework de test piloté par mots d'action (*Keyword-Driven Testing*) avec Robot Framework	02/11/2020
0.3	Nouveautés : ● Framework de test piloté par les comportements (*Behavior-Driven Development* (BDD)), avec Cucumber.js et Protractor, en TypeScript	01/02/2021
0.4	À paraître.	01/03/2021

Auteurs

Nous avons écrit ce livre à six mains. Si la dernière itération n'est pas parfaite, la prochaine sera meilleure.

Xavier
PIGEON
xavierpigeon.com

Expert Méthode & Qualité en Stratégie IT, Coach Agile / Craftsmanship / Test
Fondateur & CEO de Chrysocode

chrysocode.io
contact@chrysocode.io

Auteur de GOST
gearsoftesting.org

Développeur de
TestAsYouThink
testasyouthink.org

Contributeur de
Wikipédia

Xavier Pigeon est Ingénieur Logiciel, et évolue aujourd'hui en tant qu'Expert Méthode & Qualité en Stratégie IT. Il est aussi l'auteur du framework méthodologique GOST dédié à une approche holistique de gestion de la qualité logicielle et à la conception de stratégies adaptatives de test.
Xavier se consacre au coaching organisationnel et technique, en associant étroitement Agilité, Test et Software Craftsmanship (artisanat du code et compagnonnage logiciel). Il accompagne notamment les équipes dans leur cheminement vers l'**excellence ingénierique**, en les guidant dans leur appropriation de méthodes et pratiques d'ingénierie alignées avec un monde VUCA (volatile, incertain, complexe et ambigu).

En 2019, Xavier a fondé la marque **CHRYSOCODE** et sa société Chrysocode IT, un cabinet de conseil dont il est le dirigeant.

Contribution aux contenus du livre :

- Définition des types de framework d'automatisation de test pour tous les chapitres.
- Développement de l'application SPA en JavaScript avec le framework web Aurelia, utilisée pour automatiser les tests.
- Chapitre "Framework de test piloté par mots d'action", dont son implémentation avec Robot Framework.
- Chapitre "Framework de test piloté par les comportements" : bonnes pratiques du framework et implémentation avec Robot Framework.
- Bonnes pratiques du chapitre "Framework de test fondé sur des composants".

CEO d'All4Test

Co-auteur du livre Les Tests Logiciels en Agile

all4test.fr
contact@all4test.com

Julien
VAN QUACKEBEKE

Julien est passionné par le sujet du test et de la qualité et aime partager ses connaissances sur le sujet. A ce titre il anime la communauté "test et qualité" de Telecom Valley (animateur du numérique en PACA), il organise une fois par an la "Soirée du test logiciel", et co-rédacteur du livre du CFTL en 2019 Les tests logiciels en Agile.

Julien Van Quackebeke est également le fondateur de la société **ALL4TEST**, un des principaux "pure player" du test logiciel en France depuis 2006.

Consultant & Formateur Test /
Test lead Automation

all4test.fr

Zied
HANNACHI

Zied Hannachi est un référent technique en automatisation de test, dont l'éventail des technologies de programmation ou de test lui permettent d'intervenir dans de nombreux contextes d'entreprise. Il consacre ses activités à implémenter des stratégies de test et des solutions d'automatisation. Zied éprouve un attrait particulier pour les méthodes de *Shift-Left Testing* comme *Behavior-Driven Development*. C'est ainsi qu'il a publié de nombreux articles dont il fait profiter la communauté du test en France.
Zied a rejoint All4Test en 2017 où il a pu évoluer et participer à de nombreuses publications sur l'automatisation des tests.

Contribution aux contenus du livre :

- Chapitre "Framework par script linéaire" : bonnes pratiques et implémentation avec Selenium IDE.
- Chapitre "Framework de test fondé sur des composants" : implémentation du POM avec Selenium WebDriver.

Tawfik
HOSNI

Ingénieur QA / Automatisation Test

All4Test

all4test.fr

Tawfik Hosni est un ingénieur en automatisation de test, il possède une vaste expérience dans la création de solutions d'automatisation de test web et d'API.
Tawfik a intégré All4Test en 2019 où il a pu participer à des publications sur l'automatisation des tests.

Contribution aux contenus du livre :

- Implémentation BDD avec Cucumber.js et Protractor en TypeScript

Pourquoi un livre blanc dédié aux frameworks de test automatisé ?

Combien d'entre vous n'ont jamais connu de problèmes pour automatiser leurs tests dans leurs projets informatiques ?

Savez-vous que par expérience, on constate que plus de 50 % des projets d'automatisation de test fonctionnel sont des échecs ?!
Comme ces projets sont souvent trop compliqués et coûteux à maintenir, les causes d'échec sont nombreuses. Afin d'éviter ces pièges, mieux vaut bien sûr mettre en place une véritable stratégie d'automatisation de test. Mais il faut également être capable de comprendre l'**utilité d'un framework d'automatisation de test**, *et savoir comment choisir le bon framework par rapport à ses besoins et ses contraintes techniques et humaines.*

C'est pour apporter des réponses concrètes à ces questions que nous avons rédigé ce livre blanc gratuit.
Pour avoir une vision complète du sujet, **All4Test**, "pure player" du Test logiciel en France, et **Chrysocode**, expert de l'Ingénierie du Code et des Développements Agiles, ont collaboré à la rédaction de cet opus.

Suite à cette première version, nous mettrons à jour ce document via la présentation d'autres outils et frameworks d'automatisation de test très régulièrement.
Alors restez connecté et suivez cela de près !

Convaincus par le rôle incontournable des tests dans la création de logiciels de qualité, nous voulons démocratiser les tests et leur automatisation, en rendant accessibles les choix technologiques structuraux et les bonnes pratiques d'automatisation de test.
Avec le déploiement de l'agilité dans les entreprises, puis par la suite la mise en place du DEvOps, les méthodes de travail ont évoluées et la fréquence de livraison d'un logiciel a augmenté.
La question n'est donc plus : est-ce utile ou rentable d'automatiser les tests fonctionnels de mon logiciel ?
En effet, il n'est simplement plus possible de garantir les rythmes de livraison actuels sans automatiser ces tests, au moins en partie.
L'enjeu est plutôt de savoir comment faire pour parvenir à automatiser ces tests avec une approche "industrielle" et garantir que l'automatisation des tests fonctionnera durant toute la vie du produit.
Mais le contexte peut varier : entre automatiser un historique de TNR (legacy) vieux de plusieurs années et automatiser les tests de nouvelles fonctions "à la volée" dans une équipe agile, le besoin n'est pas le même.
Automatiser des tests d'applications mobile, web ou un ERP là aussi les contraintes techniques sont différentes.
Il est donc nécessaire de bien définir le périmètre de votre projet d'automatisation en début de projet, qui aura la charge de superviser le projet, mais aussi de le maintenir.

À qui ce livre est-il destiné ?

La vocation de ce livre est d'apporter une contribution utile à la communauté francophone des praticiens du test logiciel. Son ambition est d'apporter des réponses concrètes à des questions qui nous paraissent fondamentales dans l'automatisation des tests.

Nous destinons ce livre à tous ceux qui, néophytes ou experts du test, souhaitent prendre du recul par rapport aux frameworks d'automatisation de test et aux opportunités qu'ils représentent selon les contextes. Les novices y trouveront des définitions utiles à leur compréhension, concrétisées par des implémentations avec leur code source, afin qu'elles les guident dans leurs projets. Les experts quant à eux y trouveront des bonnes pratiques pour enrichir leur stratégie d'automatisation de test, ainsi qu'une utilisation avancée des technologies de test choisies dans les exemples d'implémentation.

Comment lire ce livre ?

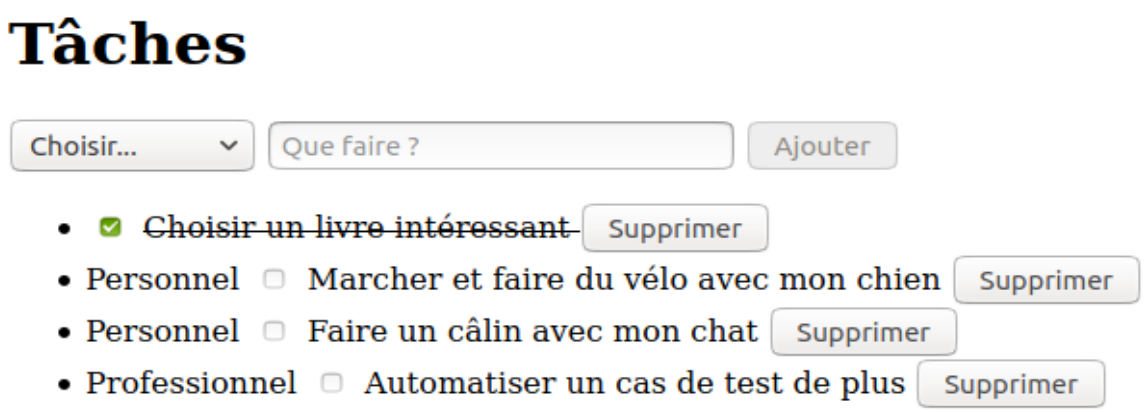
Les chapitres qui suivent sont chacun consacrés à un type de framework d'automatisation de test. Après en avoir donné une définition, nous vous proposons trois bonnes pratiques à essayer dans votre contexte et qui peuvent faire une différence dans votre stratégie de test. S'ensuit un exemple d'implémentation spécifique avec une technologie de test, et qui se rapporte seulement au framework présenté.

À des fins d'illustration des concepts relatifs aux frameworks de test, nous allons donc tester de différentes façons une application web de type "Todo List", à travers quelques cas de test très simples.

- Je peux ajouter une tâche.
- Je peux terminer une tâche.
- Je peux supprimer une tâche.
- Je peux catégoriser des tâches.

Le code source est disponible en ligne : https://github.com/livre-blanc-adtf2020. Le README de chaque dépôt Git fournit les instructions pour rejouer les tests chez vous.

Toutes les implémentations fournies sont en Anglais par souci d'homogénéité du code source, afin de faciliter la comparaison des tests automatisés avec les différentes technologies employées. L'application satisfait elle-même des prérequis de testabilité en boîte noire, notamment pour assurer la robustesse et la maintenabilité des tests automatisés.



Application "Tout Doux" de gestion de tâches

Capture vidéo des tests

Vos retours nous sont précieux !

Souhaitez-vous nous faire un retour après avoir lu ce livre, voire pendant votre lecture ? N'hésitez pas à nous écrire à l'adresse électronique suivante :

retour_lecteur_adtf2020@chrysocode.io

Nous nous ferons un plaisir de vous répondre !

Framework par script linéaire

Un framework par script linéaire, ou *linear scripting framework* en anglais, consiste à enregistrer une première fois le déroulement manuel d'un script manuel de test, puis à rejouer automatiquement et à l'identique la séquence des interactions homme-machine précédemment enregistrées. Ce type de framework est aussi appelé framework "Record & Playback" (parfois aussi nommé "Record & Replay").

3 bonnes pratiques à essayer

Sélectionnez le localisateur approprié. La sélection d'un localisateur stable est l'étape la plus importante pour avoir un script de test maintenable. Le localisateur d'élément peut changer fréquemment dans le DOM, ce qui entraîne un test impossible à maintenir. Lors de l'enregistrement du scénario de test, le développeur doit choisir le localisateur qui ne changera probablement pas lors de la modification du formulaire. Est une bonne pratique pour stocker le localisateur de l'élément souvent utilisé dans des variables. Si le localisateur est alors affecté par une modification, il sera plus facile de définir le nouveau localisateur dans le script de test. De cette manière, le scénario de test devient également plus lisible.

Gardez le test maintenable. Évitez les doublons dans les scripts de test. Modulez les scénarios de test et les suites de tests et réutilisez les mêmes scénarios de test dans plusieurs suites de tests. Créer un nouveau extensions ou Rollup pour la séquence de commandes fréquemment utilisée.
Gardez le cas de test et les suites de tests petits. Chaque suite de tests doit tester un seul cas d'utilisation. Séparez la suite de tests en plusieurs scénarios de test, chaque scénario de test est une seule unité de test.
Refactoriser le test lors de la refactorisation du code ou de la modification du formulaire.
Documentez les tests. Utilisez les conventions de dénomination appropriées et enregistrez la suite de tests et les cas de test dans une structure de dossiers organisée.
Utilisez les fonctions setUp et tearDown pour rendre le test répétable. Évitez d'utiliser d'autres cas de test comme setUp ou tearDown, si le cas de test du setUp échoue, la configuration ne sera pas correcte.

Insérer des commentaires pour plus de lisibilité. Lorsque les scripts commencent à grossir, ils deviennent difficiles à lire. Un excellent moyen de s'assurer qu'ils seront mieux compris à l'avenir est d'inclure des commentaires pour fournir des informations sur ce qui est fait.
Vous pouvez ajouter manuellement des commentaires aux scripts directement dans le code source via des commentaires HTML normaux.

Implémentation avec Selenium IDE

Le code ci-après est généré par Selenium IDE après exécution manuelle et capture du script de test.

IcanaddatodoTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class IcanaddatodoTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
```

```
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  public void SuiteSetup() {
    System.out.println("`set speed` is a no-op in code export, use
`pause` instead");
    driver.get("http://localhost:9090/");
    driver.manage().window().setSize(new Dimension(600, 300));
  }
  @Test
  public void icanaddatodo() {
    SuiteSetup();
    TestSetup();
    {
      List<WebElement> elements =
driver.findElements(By.xpath("//input[@type=\'text\']"));
      assert(elements.size() > 0);
    }

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Ado
pter de bonnes pratiques de test");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector("ul .au-target:nth-child(2)"));
      assert(elements.size() > 0);
    }

assertFalse(driver.findElement(By.xpath("//input[@type=\'checkbox\']"
)).isSelected());
    TestTeardown();
    SuiteTeardown();
  }
}
```

IcanselectatodoTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class IcanselectatodoTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
```

```
  public void SuiteSetup() {
    System.out.println("`set speed` is a no-op in code export, use
`pause` instead");
    driver.get("http://localhost:9090/");
    driver.manage().window().setSize(new Dimension(600, 300));
  }
  @Test
  public void icanselectatodo() {
    SuiteSetup();
    TestSetup();

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Ado
pter de bonnes pratiques de test");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Com
prendre le Keyword-Driven Testing");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Aut
omatiser des cas de test avec Robot Framework");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector("ul > .au-target:nth-child(1) >
.au-target:nth-child(2)"));
      assert(elements.size() > 0);
    }
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector(".au-target:nth-child(2) >
.au-target:nth-child(2)"));
      assert(elements.size() > 0);
    }
    {
      List<WebElement> elements =
```

```
driver.findElements(By.cssSelector(".au-target:nth-child(3) >
.au-target:nth-child(2)"));
      assert(elements.size() > 0);
    }
    driver.findElement(By.cssSelector(".au-target:nth-child(1) >
.au-target:nth-child(1)")).click();

assertTrue(driver.findElement(By.xpath("//input[@type=\'checkbox\']")
).isSelected());

assertFalse(driver.findElement(By.xpath("(//input[@type=\'checkbox\']
)[2]")).isSelected());

assertFalse(driver.findElement(By.xpath("(//input[@type=\'checkbox\']
)[3]")).isSelected());
    TestTeardown();
    SuiteTeardown();
  }
}
```

IcanremoveatodoTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
```

```
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class IcanremoveatodoTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  public void SuiteSetup() {
    System.out.println("`set speed` is a no-op in code export, use
`pause` instead");
    driver.get("http://localhost:9090/");
    driver.manage().window().setSize(new Dimension(600, 300));
  }
  @Test
  public void icanremoveatodo() {
    SuiteSetup();
    TestSetup();

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Cho
isir le bon type de framework de test");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements = driver.findElements(By.cssSelector
```

```
("ul .au-target:nth-child(3)"));
      assert(elements.size() > 0);
    }
driver.findElement(By.cssSelector("ul.au-target:nth-child(3)")).click
();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector("ul .au-target:nth-child(2)"));
      assert(elements.size() == 0);
    }
    TestTeardown();
    SuiteTeardown();
  }
}
```

IcancategorizesometodoTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
```

```
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class IcancategorizesometodosTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  public void SuiteSetup() {
    System.out.println("`set speed` is a no-op in code export, use
`pause` instead");
    driver.get("http://localhost:9090/");
    driver.manage().window().setSize(new Dimension(600, 300));
  }
  @Test
  public void icancategorizesometodos() {
    SuiteSetup();
    TestSetup();
    driver.findElement(By.cssSelector(".au-target:nth-child(2) >
.au-target:nth-child(2)")).sendKeys("Choisir un livre intéressant");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.xpath("//span[contains(.,\'Personnel\')]"));
      assert(elements.size() == 0);
    }
    {
      List<WebElement> elements =
driver.findElements(By.xpath("//span[contains(.,\'Professionnel\')]")
```

```
);
      assert(elements.size() == 0);
    }
    {
      WebElement dropdown = driver.findElement(By.cssSelector("label
> .au-target"));
      dropdown.findElement(By.xpath("//option[. =
'Personnel']")).click();
    }

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Mar
cher et faire du vélo avec mon chien");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector(".au-target:nth-child(2) >
.au-target:nth-child(3)"));
      assert(elements.size() > 0);
    }
    {
      List<WebElement> elements =
driver.findElements(By.xpath("//span[contains(.,\'Personnel\')]"));
      assert(elements.size() > 0);
    }

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Fai
re un câlin avec mon chat");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector(".au-target:nth-child(2) >
.au-target:nth-child(3)"));
      assert(elements.size() > 0);
    }
    {
      List<WebElement> elements =
driver.findElements(By.xpath("//span[contains(.,\'Personnel\')]"));
```

```
      assert(elements.size() > 0);
    }
    {
      WebElement dropdown = driver.findElement(By.cssSelector("label
> .au-target"));
      dropdown.findElement(By.xpath("//option[. =
'Professionnel']")).click();
    }

driver.findElement(By.xpath("//input[@type=\'text\']")).sendKeys("Aut
omatiser un cas de test de plus");

driver.findElement(By.xpath("//button[@type=\'submit\']")).click();
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector(".au-target:nth-child(4) >
.au-target:nth-child(1)"));
      assert(elements.size() > 0);
    }
    {
      List<WebElement> elements =
driver.findElements(By.cssSelector(".au-target:nth-child(4) >
.au-target:nth-child(3)"));
      assert(elements.size() > 0);
    }
    TestTeardown();
    SuiteTeardown();
  }
}
```

SuiteSetupTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
```

```
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class SuiteSetupTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void suiteSetup() {
    System.out.println("`set speed` is a no-op in code export, use
`pause` instead");
    driver.get("http://localhost:9090/");
    driver.manage().window().setSize(new Dimension(600, 300));
  }
```

```
}
```

TestSetupTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class TestSetupTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
```

```
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void testSetup() {
    {
      WebDriverWait wait = new WebDriverWait(driver, 30);

wait.until(ExpectedConditions.presenceOfElementLocated(By.id("heading
")));
    }
    assertThat(driver.getTitle(), is("Tout Doux"));
  }
}
```

TestTeardownTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
```

```
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class TestTeardownTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void testTeardown() {
  }
}
```

SuiteTeardownTest.java

```
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
```

```
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class SuiteTeardownTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void suiteTeardown() {
    driver.close();
  }
}
```

Le dépôt du code source est disponible ici.

Framework de test piloté par mots d’action

Un framework de test piloté par mots d’action, ou *Keyword-Driven Testing framework* en anglais, consiste à automatiser un cas de test de façon littérale, au moyen de phrases verbales brèves et invariantes. Plutôt que de phrases verbales, on parle en fait de mots d’action, ou mots-clés : chacun d’eux est constitué d’un sujet facultatif, puis d’un verbe obligatoire, puis d’un complément si nécessaire, le tout placé au début de chaque ligne. Un mot d’action peut recevoir des arguments placés à sa droite. Pour distinguer un mot d’action et ses arguments d’une part, et les arguments entre eux d’autre part, on utilise un séparateur, qui peut être un nombre constant d’espaces, une tabulation ou un caractère spécial. Du fait de cette structure, le code de test peut être lu soit sous forme de texte, soit sous forme tabulaire. C’est pourquoi on rencontre parfois les termes d’*Action Word-Based Testing* ou de *Table-Driven Testing* pour désigner le *Keyword-Driven Testing* (KDT).

3 bonnes pratiques à essayer

Organisez vos mots d’action entre eux en les découpant en largeur et en profondeur. Un découpage en largeur consiste à identifier des étapes homogènes de test dans un ordre chronologique, tandis qu’un découpage en profondeur permet de diviser pour régner sur les problématiques traitées à l’automatisation des tests et de casser la complexité de mise en œuvre. Au plus haut niveau de préoccupation, vous devriez trouver les mots d’action qui servent à définir un cas de test : ils sont peu réutilisables, seulement à l’échelle d’une suite de test. Au plus bas niveau de préoccupation, vous devriez trouver les interactions de base avec une interface homme-machine : ces mots d'action sont très fortement réutilisables, à l'échelle de toutes les suites du référentiel de test.

Distillez le métier, ses concepts et ses intentions autant que vous le pouvez. Ne passez à côté d’aucune opportunité d’enrichir le sens de vos tests. Le fait d’exploiter le langage du métier facilite énormément l’identification de nouveaux mots d’action factorisables et la réutilisation du code de test. Vous éviterez par la même occasion les chevauchements de responsabilité et les duplications de code.

Ne sous-estimez pas le pouvoir d’une convention de nommage forte appliquée aux mots d’action (langue, structure grammaticale, casse de caractère, conjugaison). De très nombreux problèmes peuvent être ainsi gérés proprement.

- Résoudre un conflit entre mots d’action personnalisés, voire entre un mot d’action personnalisé et un mot d’action natif ou externe.
- Différencier de manière formelle les niveaux de préoccupation des mots d’action entre porcelaine (côté utilisateur) et plomberie (côté système).
- Identifier les mots d'action mutualisés à réutiliser ou pas dans les suites de tests.

Implémentation avec Robot Framework

Robot Framework est surtout un framework de *Keyword-Driven Testing*, mais c'est aussi un framework hybride, puisqu'il supporte les tableaux de données du *Data-Driven Testing* et la syntaxe Gherkin de *Behavior-Driven Development*. Il fournit un cadre d'automatisation open source générique pour les tests d'acceptation, le développement piloté par les tests d'acceptation (ATDD) et l'automatisation des processus robotiques (RPA). Il a une syntaxe de données de test tabulaire facile à utiliser et il repose sur une approche de test fondée sur des mots d'action.
Robot Framework peut être utilisé pour :

- les tests d'application web avec Selenium WebDriver ;
- les tests d'application mobile avec Appium ;
- les tests d'interface graphique avec SikuliX ou AutoIt ;
- les tests d'API HTTP avec Requests ;
- et bien d'autres tests encore[1].

Ses capacités de test peuvent être étendues par des bibliothèques de test implémentées en Python ou Java, et les utilisateurs peuvent créer de nouveaux mots d'action de niveau supérieur à partir de mots d'action existants en utilisant la même syntaxe que celle utilisée pour créer des cas de test.

Robot Framework vient avec plusieurs éditeurs de code ou IDE. Il s'accompagne notamment d'un IDE nommé RED (*Robot EDitor*), fondé sur Eclipse IDE.

Si nous automatisons les 4 cas de test proposés pour l'application "Tout Doux" avec Robot Framework, nous obtenons le code source ci-après.

todos.robot

```
*** Settings ***
Library  SeleniumLibrary
Resource  setup_teardown.resource
Suite Setup  Prepare the test suite
Test Setup  Prepare the test case
Test Teardown  Clean up the test case
Suite Teardown  Clean up the test suite

*** Variables ***
&{CATEGORIES}
...  private=Personnel
...  professional=Professionnel
```

[1] Robot Framework, Libraries, https://robotframework.org/#libraries

```
*** Test Cases ***
I can add a todo
  Page Should Contain Element  data-id:input.text.description
  Submit a todo  Adopter de bonnes pratiques de test
  A new todo should be created  1  Adopter de bonnes pratiques de test

I can complete one of several todos
  Submit a todo  Adopter de bonnes pratiques de test
  Submit a todo  Comprendre le Keyword-Driven Testing
  Submit a todo  Automatiser des cas de test avec Robot Framework
  A new todo should be created  1  Adopter de bonnes pratiques de test
  A new todo should be created  2  Comprendre le Keyword-Driven Testing
  A new todo should be created  3  Automatiser des cas de test avec Robot Framework
  Complete a todo  2
  The todo should be completed  2
  The todo should be uncompleted  1
  The todo should be uncompleted  3

I can remove a todo
  Submit a todo  Choisir le bon type de framework de test
  Remove a todo  1
  The todo should be deleted  1

I can categorize some todos
  Submit a todo  Choisir un livre intéressant
  The todo should not be categorized  1
  Submit a todo  Marcher et faire du vélo avec mon chien  ${CATEGORIES.private}
  The todo 2 should be private
  Submit a todo  Faire un câlin avec mon chat
  The todo 3 should be private
  Submit a todo  Automatiser un cas de test de plus  ${CATEGORIES.professional}
  The todo 4 should be professional

*** Keywords ***
Submit a todo
  [Arguments]  ${description}  ${category}=${None}
  Run Keyword Unless  '${category}' == '${None}'
  ...  Select From List By Value  data-id:select.category  ${category}
  Input Text  data-id:input.text.description  ${description}
  Submit Form
```

```
A new todo should be created
  [Arguments]  ${number}  ${description}
  Element Should Contain  todo:${number}  ${description}
  Checkbox Should Not Be Selected  data-id:input.checkbox.done-${number}

Complete a todo
  [Arguments]  ${number}
  Select Checkbox  data-id:input.checkbox.done-${number}

The todo should be completed
  [Arguments]  ${number}
  Checkbox Should Be Selected  data-id:input.checkbox.done-${number}

The todo should be uncompleted
  [Arguments]  ${number}
  Checkbox Should Not Be Selected  data-id:input.checkbox.done-${number}

Remove a todo
  [Arguments]  ${number}
  Page Should Contain Button  data-id:button.remove_todo-${number}
  Click Button  data-id:button.remove_todo-${number}

The todo should be deleted
  [Arguments]  ${number}
  Page Should Not Contain Element  todo:${number}

The todo should not be categorized
  [Arguments]  ${number}
  Page Should Not Contain Element  data-id:category-${number}

The todo ${number} should be ${category}
  Element Text Should Be  data-id:category-${number}  ${CATEGORIES.${category}}
```

setup_teardown.resource

```
*** Settings ***
Library  SeleniumLibrary

*** Keywords ***
Prepare the test suite
  Open Browser  url=http://localhost:9090  browser=chrome
```

```
  Set Window Size  width=600  height=300  inner=true
  Set Selenium Timeout  10 seconds
  ${selenium_speed} =  Get Variable Value  ${SPEED}  0
  Set Selenium Speed  ${selenium speed} seconds
  Add Location Strategy  strategy_name=data-id  strategy_keyword=test id locator strategy
  Add Location Strategy  strategy_name=todo  strategy_keyword=todo locator strategy

Test id locator strategy
  [Arguments]  ${browser}  ${locator}  ${tag}  ${constraints}
  ${element}=  Execute Javascript  return window.document.querySelector('[data-id="${locator}"]');
  [Return]  ${element}

Todo locator strategy
  [Arguments]  ${browser}  ${locator}  ${tag}  ${constraints}
  ${element}=  Execute Javascript  return window.document.querySelector('ul > li[data-id="todo-${locator}"]');
  [Return]  ${element}

Prepare the test case
  Wait Until Page Contains Element  id:heading
  Title Should Be  Tout Doux

Clean up the test case
  Reload Page

Clean up the test suite
  Close All Browsers
```

Le dépôt du code source est disponible ici.

Après avoir exécuté les tests, Robot génère le rapport ci-après.

LOG

Todos Report

Generated
20201101 20:43:49 UTC+01:00
54 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20201101 20:43:34.327
End Time:	20201101 20:43:49.581
Elapsed Time:	00:00:15.254
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:12	
All Tests	4	4	0	00:00:12	

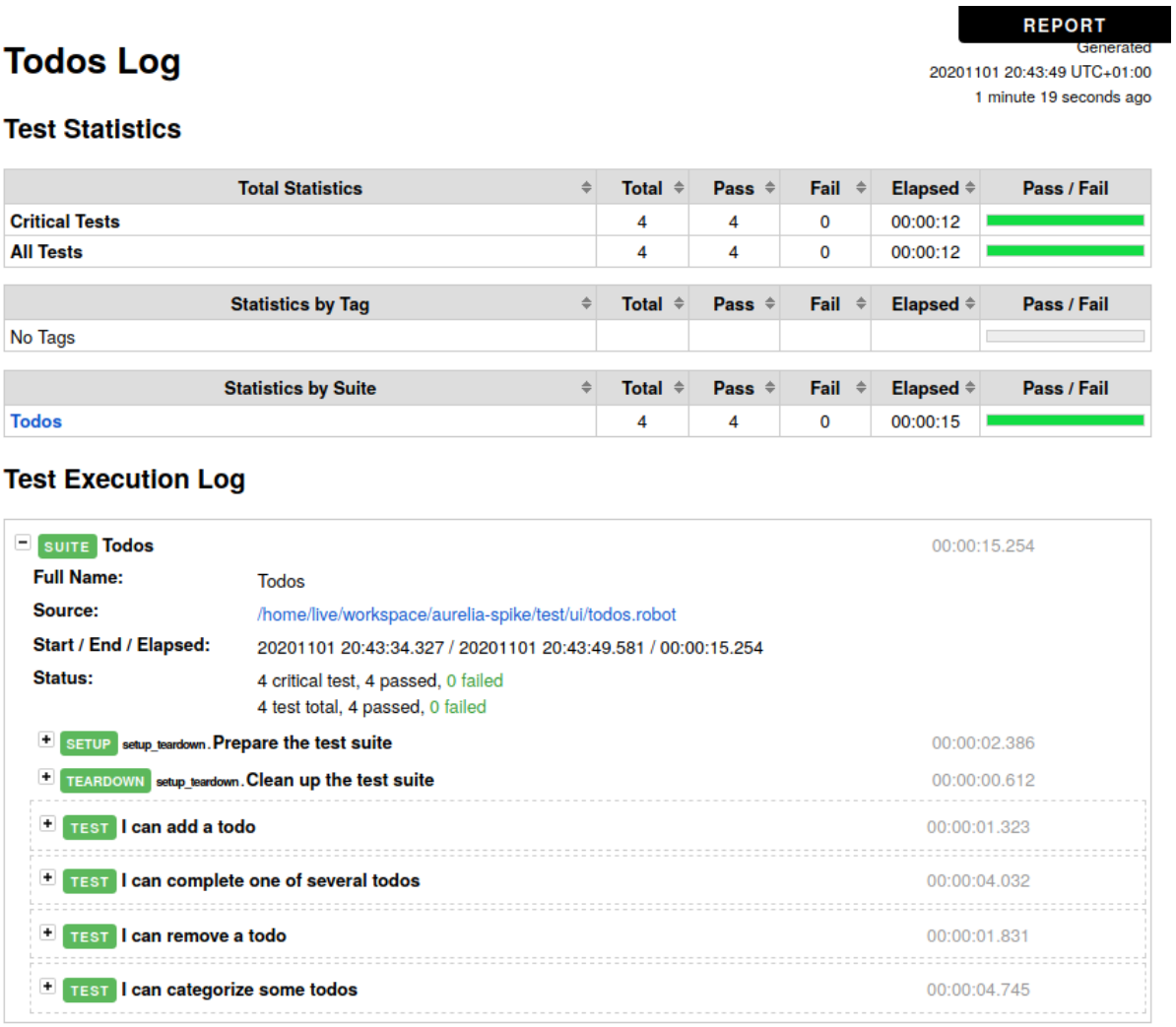
Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:15	

Test Details

Totals | Tags | Suites | Search

Type:
- Critical Tests
- All Tests



Todos Log

REPORT
Generated
20201101 20:43:49 UTC+01:00
1 minute 19 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:12	
All Tests	4	4	0	00:00:12	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:15	

Test Execution Log

SUITE Todos 00:00:15.254

Full Name: Todos
Source: /home/live/workspace/aurelia-spike/test/ui/todos.robot
Start / End / Elapsed: 20201101 20:43:34.327 / 20201101 20:43:49.581 / 00:00:15.254
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

SETUP setup_teardown . Prepare the test suite 00:00:02.386
TEARDOWN setup_teardown . Clean up the test suite 00:00:00.612
TEST I can add a todo 00:00:01.323
TEST I can complete one of several todos 00:00:04.032
TEST I can remove a todo 00:00:01.831
TEST I can categorize some todos 00:00:04.745

Et il est facile de consulter le niveau de détail souhaité simplement en dépliant les mots d'action dans le rapport.

REPORT
Generated
20201101 20:43:49 UTC+01:00
1 minute 19 seconds ago

Todos Log

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:12	
All Tests	4	4	0	00:00:12	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:15	

Test Execution Log

SUITE **Todos** 00:00:15.254

Full Name: Todos
Source: /home/live/workspace/aurelia-spike/test/ui/todos.robot
Start / End / Elapsed: 20201101 20:43:34.327 / 20201101 20:43:49.581 / 00:00:15.254
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

SETUP setup_teardown.**Prepare the test suite** 00:00:02.386

TEARDOWN setup_teardown.**Clean up the test suite** 00:00:00.612

TEST **I can add a todo** 00:00:01.323

Full Name: Todos.I can add a todo
Start / End / Elapsed: 20201101 20:43:37.009 / 20201101 20:43:38.332 / 00:00:01.323
Status: PASS (critical)

SETUP setup_teardown.**Prepare the test case** 00:00:00.035

KEYWORD SeleniumLibrary.**Page Should Contain Element** data-id:input.text.description 00:00:00.011

KEYWORD **Submit a todo** Adopter de bonnes pratiques de test 00:00:00.819

Start / End / Elapsed: 20201101 20:43:37.056 / 20201101 20:43:37.875 / 00:00:00.819

KEYWORD BuiltIn.**Run Keyword Unless** '${category}' == '${None}', Select From List By Value, data-id:select.category, ${category} 00:00:00.000

Documentation: Runs the given keyword with the given arguments if `condition` is false.
Start / End / Elapsed: 20201101 20:43:37.057 / 20201101 20:43:37.057 / 00:00:00.000

KEYWORD SeleniumLibrary.**Input Text** data-id:input.text.description, ${description} 00:00:00.706

KEYWORD SeleniumLibrary.**Submit Form** 00:00:00.108

KEYWORD **A new todo should be created** 1, Adopter de bonnes pratiques de test 00:00:00.077

Start / End / Elapsed: 20201101 20:43:37.875 / 20201101 20:43:37.952 / 00:00:00.077

KEYWORD SeleniumLibrary.**Element Should Contain** todo:${number}, ${description} 00:00:00.050

KEYWORD SeleniumLibrary.**Checkbox Should Not Be Selected** data-id:input.checkbox.done-${number} 00:00:00.025

TEARDOWN setup_teardown.**Clean up the test case** 00:00:00.378

TEST **I can complete one of several todos** 00:00:04.032

TEST **I can remove a todo** 00:00:01.831

TEST **I can categorize some todos** 00:00:04.745

Framework de test fondé sur des composants

Un framework de test fondé sur des composants, ou *component-based testing framework* en anglais, consiste à automatiser un cas de test en s'appuyant sur des composants de test qui modélisent chacun des parties de l'application à tester. Ces composants développés à seule fin de test sont capables d'interagir avec l'application, pour restituer ou modifier son état. Ils ont l'avantage d'être réutilisables, au moins pour automatiser un même cas de test, voire à l'échelle de plusieurs cas de test. Ils apportent un degré d'abstraction et de découplage supplémentaire entre les cas de test et les interactions directes avec l'application. Si l'application est modifiée, on met à jour le composant de test, sans que cela n'affecte le reste du code de test.

3 bonnes pratiques à essayer

Pour modéliser votre application du point de vue des tests, **choisissez le modèle de conception de test qui vous correspond le mieux**, en fonction de votre niveau de programmation. Vous avez notamment le choix entre le *Page Object Model* (POM), centré sur les écrans et relativement simple à mettre en place, et *Screenplay*, centré sur les acteurs, plus avancé en matière de conception logicielle et beaucoup plus fin dans sa modélisation de l'application à tester.

Hiérarchisez vos composants de test. Ces derniers vous permettent de modéliser des parties plus ou moins grandes, plus ou moins petites de votre application. Dès lors, les premiers sont modélisés de façon composite en agrégeant les seconds. Les premiers peuvent servir de façade vis-à-vis des seconds. Vous augmentez ainsi le découplage entre les cas de test et les interactions directes avec l'application. Et surtout, vous augmentez la réutilisabilité de vos composants de test !

Combinez le framework de test fondé sur des composants avec un autre type de framework, par exemple avec le framework de test piloté par les comportements (BDD). Vous obtiendrez ainsi un framework hybride qui combinera les forces de ces deux frameworks. Dans une démarche BDD, privilégiez *Screenplay* plutôt que le *Page Object Model* si possible : vous ferez plus facilement le lien entre les tests d'acceptation des histoires utilisateur et l'automatisation d'un référentiel de test.

Implémentation du POM avec Selenium WebDriver

Setup.java

```
import static org.hamcrest.CoreMatchers.is;
import static org.junit.Assert.assertThat;
```

```
import java.io.IOException;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Dimension;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Parameters;

public class Setup extends BasePage {

    public Setup() throws IOException {
        super();

    }

    @Parameters({ "browser" })
    @BeforeMethod
    public void setupTest(String browser) throws IOException {

        initialization(browser);
        driver.get(prop.getProperty("URL"));
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,
TimeUnit.MILLISECONDS);
        assertThat(driver.getTitle(), is("Tout Doux"));
        Dimension d = new Dimension(600, 300);
        driver.manage().window().setSize(d);
    }

    @AfterMethod
    public void cleanTest() {

        driver.navigate().refresh();
        driver.quit();
    }
}
```

AddTodoPageTest.java

```
import java.io.IOException;
import org.junit.Assert;
import org.testng.annotations.Test;
import com.pages.TodosPage;
import com.todo.base.Setup;

public class AddTodoPageTest extends Setup {

    public AddTodoPageTest() throws IOException {
        super();
    }

    TodosPage todosPage;

    @Test()
    public void iCanAddTodo() throws IOException {

        todosPage = new TodosPage();
        boolean elementText =
todosPage.isElementDisplayed(TodosPage.inputText);
        Assert.assertTrue(elementText);
        todosPage.submitTodo(prop.getProperty("TODO1"));
        String elementTodo =
todosPage.checkElementContain(TodosPage.element1);

Assert.assertTrue(elementTodo.contains(prop.getProperty("TODO1")));
        boolean checkbox =
todosPage.isCheckboxSelected(TodosPage.checkbox);
        Assert.assertFalse(checkbox);
    }
}
```

SelectTodoPageTest.java

```java
import java.io.IOException;

import org.junit.Assert;
import org.testng.annotations.Test;

import com.pages.TodosPage;
import com.todo.base.Setup;

public class SelectTodoPageTest extends Setup {

    public SelectTodoPageTest() throws IOException {
        super();
    }

    TodosPage todosPage;

    @Test()
    public void iCanSelectTodo() throws IOException {

        todosPage = new TodosPage();
        todosPage.submitTodo(prop.getProperty("TODO1"));
        todosPage.submitTodo(prop.getProperty("TODO2"));
        todosPage.submitTodo(prop.getProperty("TODO3"));
        String elementTodo1 =
todosPage.checkElementContain(TodosPage.element1);

Assert.assertTrue(elementTodo1.contains(prop.getProperty("TODO1")));
        String elementTodo2 =
todosPage.checkElementContain(TodosPage.element2);

Assert.assertTrue(elementTodo2.contains(prop.getProperty("TODO2")));
        String elementTodo3 =
todosPage.checkElementContain(TodosPage.element3);

Assert.assertTrue(elementTodo3.contains(prop.getProperty("TODO3")));
        todosPage.clickCheckbox();
        boolean checkbox =
todosPage.isCheckboxSelected(TodosPage.checkbox);
        Assert.assertTrue(checkbox);
```

```
        boolean checkbox2 =
todosPage.isCheckboxSelected(TodosPage.checkbox2);
        Assert.assertFalse(checkbox2);
        boolean checkbox3 =
todosPage.isCheckboxSelected(TodosPage.checkbox3);
        Assert.assertFalse(checkbox3);
    }
}
```

RemoveTodoPageTest.java

```
import java.io.IOException;

import org.junit.Assert;
import org.testng.annotations.Test;

import com.pages.TodosPage;
import com.todo.base.Setup;

public class RemoveTodoPageTest extends Setup {

    public RemoveTodoPageTest() throws IOException {
        super();
    }

    TodosPage todosPage;

    @Test()
    public void iCanRemoveTodo() throws IOException {

        todosPage = new TodosPage();
        todosPage.submitTodo(prop.getProperty("TODO4"));
        boolean buttonRemove =
todosPage.isElementDisplayed(TodosPage.buttonRemove);
        Assert.assertTrue(buttonRemove);
        todosPage.removeTodo();
        String element = todosPage.getPageSource();
```

```
Assert.assertFalse(element.contains(prop.getProperty("TODO4")));

    }
}
```

CategorizeTodoPageTest.java

```
import java.io.IOException;

import org.junit.Assert;
import org.testng.annotations.Test;

import com.pages.TodosPage;
import com.todo.base.Setup;

public class CategorizeTodoPageTest extends Setup {

    public CategorizeTodoPageTest() throws IOException {
        super();
    }

    TodosPage todosPage;

    @Test()
    public void iCanAddTodo() throws IOException {

        todosPage = new TodosPage();
        todosPage.submitTodo(prop.getProperty("TODO5"));
        String elementTodo1 =
todosPage.checkElementContain(TodosPage.todo1);

Assert.assertFalse(elementTodo1.contains(prop.getProperty("PRIVATE"))
);

Assert.assertFalse(elementTodo1.contains(prop.getProperty("PROFESSION
NAL")));
        todosPage.selectCategory(prop.getProperty("PRIVATE"));
```

```
        todosPage.submitTodo(prop.getProperty("TODO6"));
        String elementTodo2 =
todosPage.checkElementContain(TodosPage.todo2);

Assert.assertTrue(elementTodo2.contains(prop.getProperty("PRIVATE")))
;
        todosPage.submitTodo(prop.getProperty("TODO7"));
        String elementTodo3 =
todosPage.checkElementContain(TodosPage.todo3);

Assert.assertTrue(elementTodo3.contains(prop.getProperty("PRIVATE")))
;
        todosPage.selectCategory(prop.getProperty("PROFESSIONNAL"));
        todosPage.submitTodo(prop.getProperty("TODO8"));
        String elementTodo4 =
todosPage.checkElementContain(TodosPage.todo4);

Assert.assertTrue(elementTodo4.contains(prop.getProperty("PROFESSIONN
AL")));
    }
}
```

TodosPage.java

```
import java.io.IOException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.Select;

import com.todo.base.BasePage;

public class TodosPage extends BasePage {

    public TodosPage() throws IOException {
        PageFactory.initElements(driver, this);
```

```
    }

    /* Locators */

    final static String INPUT_TEXT = "//input[@type='text']";
    final static String BOUTTON_SUBMIT = "//button[@type='submit']";
    final static String ELEMENT_TODO_1 = "/html/body/ul/li/span";
    final static String CHECKBOX = "//input[@type='checkbox']";

    final static String ELEMENT1 = "/html/body/ul/li[1]/span";
    final static String ELEMENT2 = "/html/body/ul/li[2]/span";
    final static String ELEMENT3 = "/html/body/ul/li[3]/span";
    final static String CHECKBOX2 =
"//input[@data-id='input.checkbox.done-2']";
    final static String CHECKBOX3 =
"//input[@data-id='input.checkbox.done-3']";

    final static String BOUTTON_REMOVE = "ul
.au-target:nth-child(3)";
    final static String DROPDOWN =
"//select[@data-id='select.category']";
    final static String TODO_1 = "//li[@data-id='todo-1']";
    final static String TODO_2 = "//li[@data-id='todo-2']";
    final static String TODO_3 = "//li[@data-id='todo-3']";
    final static String TODO_4 = "//li[@data-id='todo-4']";

    /* @FindBy */

    @FindBy(how = How.XPATH, using = INPUT_TEXT)
    public static WebElement inputText;
    @FindBy(how = How.XPATH, using = BOUTTON_SUBMIT)
    public static WebElement submitBoutton;
    @FindBy(how = How.XPATH, using = ELEMENT_TODO_1)
    public static WebElement elementTodo1;
    @FindBy(how = How.XPATH, using = CHECKBOX)
    public static WebElement checkbox;

    @FindBy(how = How.XPATH, using = ELEMENT1)
    public static WebElement element1;
```

```
    @FindBy(how = How.XPATH, using = ELEMENT2)
    public static WebElement element2;
    @FindBy(how = How.XPATH, using = ELEMENT3)
    public static WebElement element3;
    @FindBy(how = How.XPATH, using = CHECKBOX2)
    public static WebElement checkbox2;
    @FindBy(how = How.XPATH, using = CHECKBOX3)
    public static WebElement checkbox3;

    @FindBy(how = How.CSS, using = BOUTTON_REMOVE)
    public static WebElement buttonRemove;
    @FindBy(how = How.XPATH, using = DROPDOWN)
    public static WebElement dropDown;
    @FindBy(how = How.XPATH, using = TODO_1)
    public static WebElement todo1;
    @FindBy(how = How.XPATH, using = TODO_2)
    public static WebElement todo2;
    @FindBy(how = How.XPATH, using = TODO_3)
    public static WebElement todo3;
    @FindBy(how = How.XPATH, using = TODO_4)
    public static WebElement todo4;

    /* Methods */

    public void submitTodo(String todo) {

        writeText(inputText, todo);
        click(submitBoutton);
    }

    public void clickCheckbox() {

        click(checkbox);
    }

    public void removeTodo() {

        click(buttonRemove);
    }
```

```
    public Boolean isElementDisplayed(WebElement element) {

        Boolean isElementTextDisplayed = element.isDisplayed();
        return isElementTextDisplayed;
    }

    public Boolean isCheckboxSelected(WebElement element) {

        Boolean isCheckboxSelected = element.isSelected();
        return isCheckboxSelected;
    }

    public String checkElementContain(WebElement element) {

        String elementTodo = element.getText();
        return elementTodo;
    }

    public String getPageSource() {

        String element = driver.getPageSource();
        return element;
    }

    public void selectCategory(String category) {

        WebElement element = dropDown;
        Select select = new Select(element);
        select.selectByValue(category);
    }

}
```

Le dépôt du code source est disponible ici.

Framework de test piloté par les comportements

Un framework de développement piloté par les comportements, ou *Behavior-Driven Development framework* en anglais, consiste à automatiser un cas de test de façon littérale, au moyen de phrases verbales brèves et paramétrables, selon une syntaxe étendue nommée Gherkin, souvent connue pour son triplet *Given-When-Then*. Plutôt que de phrases verbales, on parle en fait d'étapes de test : chacune d'elles est constituée d'un sujet obligatoire, puis d'un verbe obligatoire, puis d'un complément si nécessaire, le tout placé sur une même ligne. Une étape de test peut recevoir un ou des arguments, intégrés à sa structure grammaticale. Gherkin permet également d'alimenter une étape de test par un tableau de données : chaque colonne correspond à une donnée reçue en argument de l'étape, et le cas de test sera exécuté autant de fois qu'il y a de lignes dans le tableau. Cette facilité permet d'exécuter un même scénario en plusieurs cas de test, sans dupliquer le scénario pour chaque jeu de données.

3 bonnes pratiques à essayer

Dans la définition des cas de test, focalisez-vous sur les comportements, étape par étape, c'est-à-dire sur les intentions des utilisateurs finaux, et non sur les détails d'IHM[2] (exemples : éléments graphiques comme les onglets et menus, les boutons, les listes déroulantes, les boutons radio, les cases à cocher, etc). L'IHM doit pouvoir évoluer, sans que le cas de test et son scénario n'aient besoin d'être modifiés. Le cas échéant, il suffira de mettre à jour l'implémentation d'une étape BDD, sans autre coût de maintenance des tests. De plus, si le test échoue, son scénario vous apportera la prise de recul nécessaire et vous saurez dire pourquoi. Sinon il y a des chances que le test soit abandonné, et finalement supprimé du référentiel, faute de savoir quoi en faire.
Exemple :

Ce qu'il NE faut PAS faire
Scénario : Je peux créer un élément dans la liste. Quand le texte "Adopter BDD & Gherkin" est placé dans le champ de saisie Et que le bouton "Ajouter" reçoit un clic Alors l'élément "Adopter BDD & Gherkin" est affiché dans la liste
Ce qu'il faut faire
Scénario : Je peux ajouter une tâche. Quand j'ajoute une tâche "Adopter BDD & Gherkin" Alors une nouvelle tâche 1 "Adopter BDD & Gherkin" est créée

Le scénario à éviter occulte ce qui est vraiment important par quantité de détails secondaires, il décrit ce qui est apparent et trivial : c'est un scénario impératif. Le scénario

[2] IHM : Interface Homme-Machine

bien écrit énonce clairement à quoi sert le sujet de test, il délivre une connaissance : c'est un scénario déclaratif.

Faites de vos scenarii de test des exemples concrets de l'utilisation qui peut être faite du sujet de test. Pour être pertinent, un scénario de test doit reposer sur des données concrètes. A contrario, des informations vagues ou l'absence de données concrètes sont inefficaces pour caractériser le comportement d'un logiciel, car elles ne permettent pas d'identifier ni de lever des ambiguïtés.
Exemple :

Ce qu'il NE faut PAS faire
Scénario : Je peux ajouter une tâche. Quand j'ajoute une tâche Alors une nouvelle tâche est créée
Ce qu'il faut faire
Scénario : Je peux ajouter une tâche. Quand j'ajoute une tâche "Adopter BDD & Gherkin" Alors une nouvelle tâche 1 "Adopter BDD & Gherkin" est créée

Le scénario à éviter est incomplet et vague : on ne sait pas ce qu'est exactement une tâche, ni ce que devient la tâche après ajout. Même si on peut faire des suppositions, rien n'est vraiment clair, et il serait facile de mal interpréter l'aboutissement du scénario. Au contraire, le scénario bien écrit ne laisse pas de place au doute : une tâche est une chaîne de caractères constituée de plusieurs mots formant une phrase, et la tâche créée est retournée et ordonnée.

Dans un scénario, concentrez-vous sur un seul comportement à la fois. En limitant la portée d'un scénario, vous en réduisez la complexité : dès lors, le comportement attendu est plus facile à comprendre, à tester et à implémenter. En cas de régression, il sera également plus aisé d'identifier la cause de l'échec.
Exemple :

Ce qu'il NE faut PAS faire
Scénario : Je peux ajouter une tâche. Quand j'ajoute une tâche "Adopter BDD & Gherkin" Alors une nouvelle tâche 1 "Adopter BDD & Gherkin" est créée Quand je supprime la tâche 1 Alors la tâche 1 devrait être supprimée
Ce qu'il faut faire
Scénario : Je peux ajouter une tâche. Quand j'ajoute une tâche "Adopter BDD & Gherkin"

Alors une nouvelle tâche 1 “Adopter BDD & Gherkin” est créée

Scénario : Je peux supprimer une tâche.

Étant donné qu'il existe une tâche 1 “Adopter BDD & Gherkin”
Quand je supprime la tâche 1
Alors la tâche 1 devrait être supprimée

Le scénario à éviter met bout à bout deux comportements, ce qui laisse supposer à tort qu'il existe une suite logique entre les deux et que le résultat final est l'aboutissement de cette suite. En fait, il s'agit bien de deux cas de test distincts, dont les résultats sont indépendants. Au contraire, les scenarii bien écrits rendent compte immédiatement de cette indépendance entre les deux cas de test en les séparant par leur définition.

Implémentation avec Robot Framework

En tant que framework hybride, Robot Framework supporte partiellement la syntaxe Gherkin de *Behavior-Driven Development* (BDD), en proposant de préfixer chaque étape d’un cas de test par *Given*, *When* ou *Then* selon qu’un mot d’action est invoqué respectivement à des fins de préparation du contexte initial, ou pour interagir avec le sujet de test, ou encore afin de vérifier les résultats obtenus[3]. Les mots-clés *Given-When-Then* peuvent aussi être remplacés au choix par *And* ou *But*, si *Given*, *When* ou *Then* sont répétés plusieurs fois à la suite. À noter qu’accessoirement, il n’existe pas d’alternative Gherkin dans d’autres langues que l’anglais.

De plus, pour définir nos étapes de test conformément à Gherkin, nous avons recouru aux arguments embarqués, c’est-à-dire la possible d’insérer des arguments à l’intérieur des étapes de test[4]. Cette fonctionnalité de Robot Framework est inspirée de la définition des étapes par Cucumber. Les paramètres peuvent être associés à des expressions régulières. Même si ces dernières sont facultatives, il est fortement recommandé de les fournir, pour éviter toute erreur d’interprétation à l’exécution des tests.

Par ailleurs, nous utilisons le mot d’action natif “Test Setup” au lieu du mot-clé “Background” propre à Gherkin[5]. Pour une meilleure compréhension des correspondances entre Robot Framework et Gherkin, vous trouverez les mots d’action de Robot Framework qui répondent au même besoin que les mots-clés de Gherkin dans le tableau ci-dessous. Les trois styles de cas de test[6] de Robot Framework y sont représentés.

---

[3] Guide de l’Utilisateur de Robot Framework, Behavior-Driven Style : http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#behavior-driven-style
[4] Guide de l’Utilisateur de Robot Framework, Embedding arguments into keyword name : http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#embedding-arguments-into-keyword-name
[5] Gherkin reference : https://cucumber.io/docs/gherkin/reference/
[6] Guide de l’Utilisateur de Robot Framework, Different test case styles : http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#different-test-case-styles

Gherkin	Robot Framework	Style de cas de test avec Robot Framework
Feature	Documentation	Keyword-Driven Style
Rule	Documentation	Keyword-Driven Style
Example	Test Case	Keyword-Driven Style
Scenario	Test Case	Keyword-Driven Style
Given	Given	Behavior-Driven Style
When	When	Behavior-Driven Style
Then	Then	Behavior-Driven Style
And	And	Behavior-Driven Style
But	But	Behavior-Driven Style
Background	Test Setup	Keyword-Driven Style
Scenario Outline Scenario Template	Template Test Template	Data-Driven Style
Examples	Template Test Template	Data-Driven Style

Ci-après, nous avons implémenté un framework BDD avec Robot Framework pour automatiser les cas de test de l'application web "Tout Doux".

todos.robot

```
*** Settings ***
Library  SeleniumLibrary
Resource  setup_teardown.resource
Suite Setup  Prepare the test suite
Test Setup  Prepare the test case
Test Teardown  Clean up the test case
Suite Teardown  Clean up the test suite

*** Variables ***
&{CATEGORIES}
...  private=Personnel
...  professional=Professionnel
```

```
*** Test Cases ***
I can add a todo
  When I submit a todo "Adopter de bonnes pratiques de test"
  Then a new todo #1 should be created being "Adopter de bonnes pratiques de test"

I can complete one of several todos
  Given an existing todo #1 being "Adopter de bonnes pratiques de test"
  And an existing todo #2 being "Comprendre le Keyword-Driven Testing"
  And an existing todo #3 being "Automatiser des cas de test avec Robot Framework"
  When I complete the todo #2
  Then the todo #2 should be completed
  And the todo #1 should be uncompleted
  And the todo #3 should be uncompleted

I can remove a todo
  Given an existing todo #1 being "Choisir le bon type de framework de test"
  When I remove a todo #1
  Then the todo #1 should be deleted

I can categorize some todos
  When I submit a todo "Choisir un livre intéressant"
  Then the todo #1 should not be categorized
  When I submit a private todo "Marcher et faire du vélo avec mon chien"
  Then the todo #2 should be private
  When I submit a todo "Faire un câlin avec mon chat"
  Then the todo #3 should be private
  When I submit a professional todo "Automatiser un cas de test de plus"
  Then the todo #4 should be professional

*** Keywords ***
# STEP DEFINITIONS
I submit a todo "${description:[^\n]+}"
  Submit a todo  ${description}

I submit a ${category:(private|professional)} todo "${description:[^\n]+}"
  Submit a todo  ${description}  ${CATEGORIES.${category}}

a new todo #${number:\d+} should be created being "${description:[^\n]+}"
  Should have a new todo  ${number}  ${description}
```

```
an existing todo #${number:\d+} being "${description:[^\n]+}"
  Submit a todo  ${description}
  Should have a new todo  ${number}  ${description}

I complete the todo #${number:\d+}
  Select Checkbox  data-id:input.checkbox.done-${number}

the todo #${number:\d+} should be completed
  Checkbox Should Be Selected  data-id:input.checkbox.done-${number}

the todo #${number:\d+} should be uncompleted
  Checkbox Should Not Be Selected  data-id:input.checkbox.done-${number}

I remove a todo #${number:\d+}
  Page Should Contain Button  data-id:button.remove_todo-${number}
  Click Button  data-id:button.remove_todo-${number}

the todo #${number:\d+} should be deleted
  Page Should Not Contain Element  todo:${number}

the todo #${number:\d+} should not be categorized
  Page Should Not Contain Element  data-id:category-${number}

the todo #${number:\d+} should be ${category:(private|professional)}
  Element Text Should Be  data-id:category-${number}  ${CATEGORIES.${category}}

# KEYWORDS EXCEPT STEPS
Submit a todo
  [Arguments]  ${description}  ${category}=${None}
  Run Keyword Unless  '${category}' == '${None}'
  ...  Select From List By Value  data-id:select.category  ${category}
  Input Text  data-id:input.text.description  ${description}
  Submit Form

Should have a new todo
  [Arguments]  ${number}  ${description}
  Element Should Contain  todo:${number}  ${description}
  Checkbox Should Not Be Selected  data-id:input.checkbox.done-${number}
```

setup_teardown.resource

```
*** Settings ***
Library  SeleniumLibrary

*** Keywords ***
Prepare the test suite
  Open Browser  url=http://localhost:9090  browser=chrome
  Set Window Size  width=600  height=300  inner=true
  Set Selenium Timeout  10 seconds
  ${selenium_speed} =  Get Variable Value  ${SPEED}  0
  Set Selenium Speed  ${selenium speed} seconds
  Add Location Strategy  strategy_name=data-id  strategy_keyword=test id locator
strategy
  Add Location Strategy  strategy_name=todo  strategy_keyword=todo locator strategy

Test id locator strategy
  [Arguments]  ${browser}  ${locator}  ${tag}  ${constraints}
  ${element}=  Execute Javascript  return
window.document.querySelector('[data-id="${locator}"]');
  [Return]  ${element}

Todo locator strategy
  [Arguments]  ${browser}  ${locator}  ${tag}  ${constraints}
  ${element}=  Execute Javascript  return window.document.querySelector('ul >
li[data-id="todo-${locator}"]');
  [Return]  ${element}

Prepare the test case
  Wait Until Page Contains Element  id:heading
  Title Should Be  Tout Doux

Clean up the test case
  Reload Page

Clean up the test suite
  Close All Browsers
```

Le dépôt du code source est disponible ici.

Après avoir exécuté les tests, Robot génère le rapport ci-après.

LOG

Todos Report

Generated
20201101 20:53:11 UTC+01:00
6 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20201101 20:52:46.421
End Time:	20201101 20:53:11.028
Elapsed Time:	00:00:24.607
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:12	
All Tests	4	4	0	00:00:12	

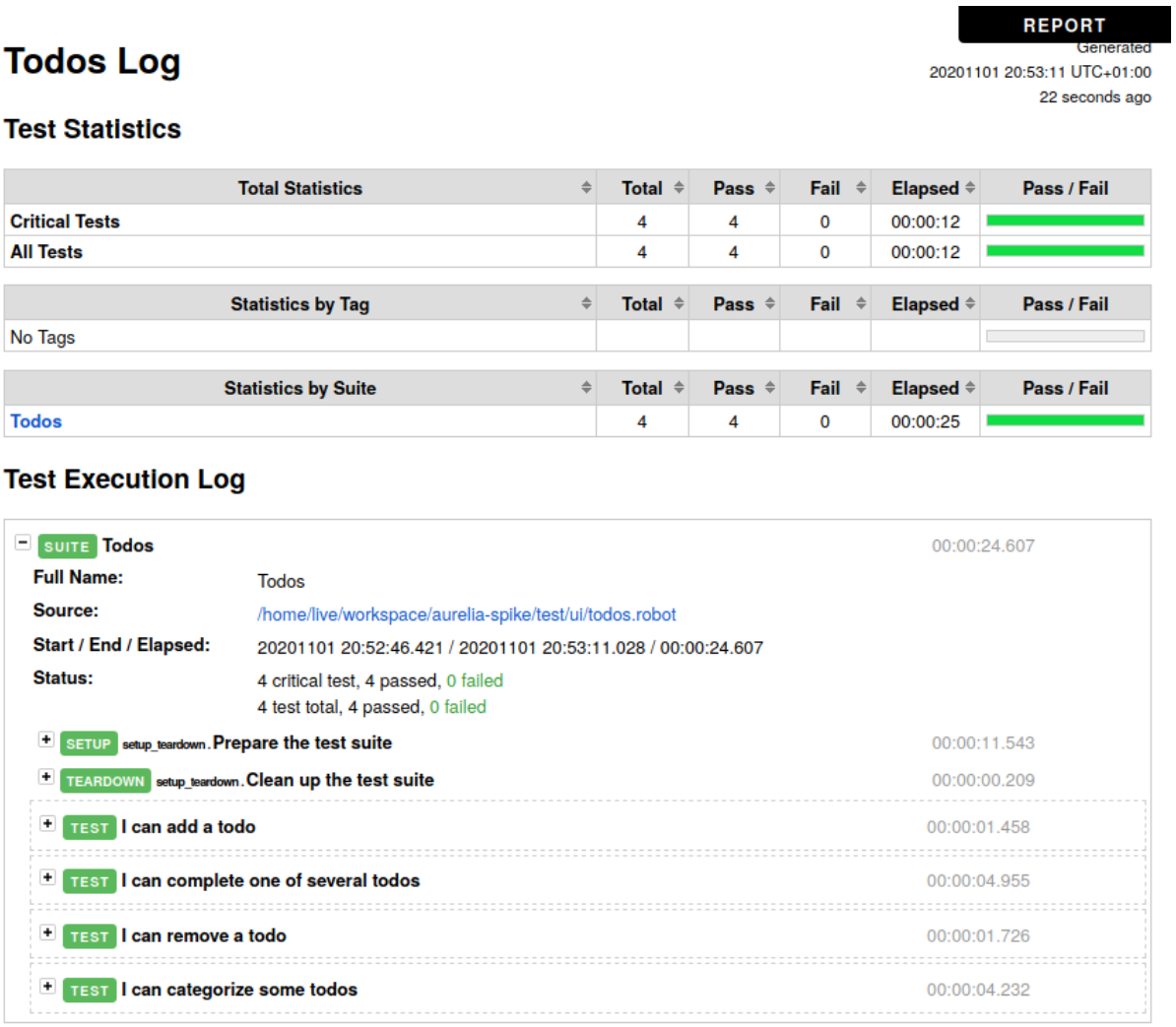
Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:25	

Test Details

Totals | Tags | Suites | Search

Type:
- ○ Critical Tests
- ○ All Tests



Todos Log

REPORT
Generated
20201101 20:53:11 UTC+01:00
22 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:12	
All Tests	4	4	0	00:00:12	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:25	

Test Execution Log

SUITE Todos 00:00:24.607

Full Name: Todos
Source: /home/live/workspace/aurelia-spike/test/ui/todos.robot
Start / End / Elapsed: 20201101 20:52:46.421 / 20201101 20:53:11.028 / 00:00:24.607
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

SETUP setup_teardown . Prepare the test suite 00:00:11.543
TEARDOWN setup_teardown . Clean up the test suite 00:00:00.209
TEST I can add a todo 00:00:01.458
TEST I can complete one of several todos 00:00:04.955
TEST I can remove a todo 00:00:01.726
TEST I can categorize some todos 00:00:04.232

Et il est facile de consulter le niveau de détail souhaité simplement en dépliant les mots d'action dans le rapport.

REPORT
Generated
20201101 21:02:48 UTC+01:00
4 seconds ago

Todos Log

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:11	
All Tests	4	4	0	00:00:11	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Todos	4	4	0	00:00:16	

Test Execution Log

SUITE **Todos** 00:00:15.676

Full Name: Todos
Source: /home/live/workspace/aurelia-spike/test/ui/todos.robot
Start / End / Elapsed: 20201101 21:02:33.013 / 20201101 21:02:48.689 / 00:00:15.676
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

SETUP setup_teardown.**Prepare the test suite** 00:00:03.590

TEARDOWN setup_teardown.**Clean up the test suite** 00:00:00.167

TEST **I can add a todo** 00:00:01.594

TEST **I can complete one of several todos** 00:00:03.789

TEST **I can remove a todo** 00:00:01.813

Full Name: Todos.I can remove a todo
Start / End / Elapsed: 20201101 21:02:42.343 / 20201101 21:02:44.156 / 00:00:01.813
Status: PASS (critical)

SETUP setup_teardown.**Prepare the test case** 00:00:00.343

KEYWORD **Given an existing todo #1 being "Choisir le bon type de framework de test"** 00:00:01.000

KEYWORD **When I remove a todo #1** 00:00:00.093

Start / End / Elapsed: 20201101 21:02:43.693 / 20201101 21:02:43.786 / 00:00:00.093

KEYWORD SeleniumLibrary.**Page Should Contain Button** data-id:button.remove_todo-${number} 00:00:00.012

KEYWORD SeleniumLibrary.**Click Button** data-id:button.remove_todo-${number} 00:00:00.079

KEYWORD **Then the todo #1 should be deleted** 00:00:00.036

TEARDOWN setup_teardown.**Clean up the test case** 00:00:00.328

TEST **I can categorize some todos** 00:00:04.301

Implémentation avec Cucumber.js et Protractor en TypeScript

Cucumber.js est l'implémentation JavaScript de Cucumber, c'est un framework de test pour *Behavior-Driven Development*, permettant de réaliser des tests automatisés en Gherkin.

Protractor est un framework de test d'application web, c'est une surcouche de Selenium WebDriver qui couvre toutes ses fonctionnalités. De plus, il fournit des fonctions et des stratégies de sélection qui sont conçues pour les applications Angular et AngularJS. Protractor supporte différents frameworks BDD tels que Cucumber.js.

Ci-après, nous avons implémenté un framework BDD avec Protractor et Cucumber.js pour automatiser les cas de test de l'application web "Tout Doux".

Todos.feature

```
Feature: Todos

   Scenario: I can add a todo
      When I submit a todo "Adopter de bonnes pratiques de test"
      Then a new todo 1 should be created being "Adopter de bonnes pratiques de
test"

   Scenario: I can complete one of several todos
      Given an existing todo 1 being "Adopter de bonnes pratiques de test"
      And an existing todo 2 being "Comprendre le Keyword-Driven Testing"
      And an existing todo 3 being "Automatiser des cas de test avec Robot
Framework"
      When I complete the todo 2
      Then the todo 2 should be completed
      And the todo 1 should be uncompleted
      And the todo 3 should be uncompleted

   Scenario: I can remove a todo
      Given an existing todo 1 being "Choisir le bon type de framework de test"
      When I remove a todo 1
      Then the todo 1 should be deleted

   Scenario: I can categorize some todos
      When I submit a todo "Choisir un livre intéressant"
      Then the todo 1 should not be categorized
      When I submit a "private" todo "Marcher et faire du vélo avec mon chien"
      Then the todo 2 should be "private"
      When I submit a todo "Faire un câlin avec mon chat"
      Then the todo 3 should be "private"
```

```
    When I submit a "professional" todo "Automatiser un cas de test de plus"
    Then the todo 4 should be "professional"
```

TodoStepDefinitions.ts

```
import { Given, Then, When } from "cucumber";
import { Todos } from "../tests/Todos";
import { expect } from 'chai';

const todos = new Todos;

When('I submit a todo {string}', async (todo: string) => {
  await todos.submit(todo);
});

When('I remove a todo {int}', async (todoNumber: number) => {
  let isPresent = await todos.hasRemoveButton(todoNumber);
  expect(isPresent).equal(true);
  await todos.remove(todoNumber);
});

Then('a new todo {int} should be created being {string}', async (todoNumber:
number, expectedDescription: string) => {
  let actualDescription = await todos.getDescription(todoNumber);
  expect(actualDescription).contain(expectedDescription);
  let isSelected = await todos.isCompleted(todoNumber);
  expect(isSelected).equal(false);
});

Given('an existing todo {int} being {string}', async (todoNumber: number,
description: string) => {
  await todos.submit(description);
  let actual Description = await todos.getDescription(todoNumber);
  expect(actualDescription).contain(description);
});

When('I complete the todo {int}', async (todoNumber: number) => {
  await todos.complete(todoNumber);
});

Then('the todo {int} should be completed', async (todoNumber: number) => {
  let isSelected = await todos.isCompleted(todoNumber);
  expect(isSelected).equal(true);
});
```

```
Then('the todo {int} should be uncompleted', async (todoNumber: number) => {
  let isSelected = await todos.isCompleted(todoNumber);
  expect(isSelected).equal(false);
});

Then('the todo {int} should be deleted', async (todoNumber: number) => {
  const isPresent = await todos.hasTodo(todoNumber)
  expect(isPresent).equal(false);
});

Then('the todo {int} should not be categorized', async (todoNumber: number) => {
  const isPresent = await todos.hasCategory(todoNumber)
  expect(isPresent).equal(false);
});

When('I submit a {string} todo {string}', async (category: string, description:
string) => {
  await todos.selectCatagory(category);
  await todos.submit(description);
});

Then('the todo {int} should be {string}', async (todoNumber: number,
expectedCategory: string) => {
  let actualCategory = await todos.getCategory(todoNumber);
  expect(actualCategory).contain(todos.categories[expectedCategory]);
});
```

Todo.ts

```
import { browser, by, element } from 'protractor';
export class Todos {

    /* Variables  */
    categories = {
        private: "Personnel",
        professional: "Professionnel"
    };

    /* Methods */
    async remove(todoNumber: number) {
        let removeButtonElement = await element(by.xpath("//*[@data-id='todo-" +
todoNumber + "']")).element(by.xpath("//button[@data-id='button.removetodo']"));
        return removeButtonElement.click();
```

```
    }

    async submit(todo: string) {
        browser.manage().timeouts().implicitlyWait(300);
        await element(by.xpath("//input[@data-id='input.text.description']"
")).sendKeys(todo);
        await element(by.xpath("//button[@type='submit']")).click();
    }

    async getDescription(todoNumber: number) {
        return await element(by.xpath("//*[@data-id='todo-" + todoNumber +
"']")).getText();
    }

    async getCategory(todoNumber: number) {
        return await element(by.xpath("//*[@data-id='category-" + todoNumber +
"']")).getText();
    }

    async hasCategory(todoNumber: number) {
        let todo = await element(by.xpath("//*[@data-id='category-" + todoNumber
+ "']"));
        return await todo.isPresent();
    }

    async hasRemoveButton(todoNumber: number) {
        let removeButtonElement = await element(by.xpath("//li[@data-id='todo-" +
todoNumber + "']")).element(by.xpath("//button[@data-id='button.removetodo']"));
        return await removeButtonElement.isPresent();
    }

    async hasTodo(todoNumber: number) {
        let todo = element(by.xpath("//li[@data-id='todo-" + todoNumber + "']"));
        return await todo.isPresent();
    }

    async isCompleted(todoNumber: number) {
        return await element(by.xpath("//input[@data-id='input.checkbox.done-" +
todoNumber + "']")).isSelected();
    }

    async complete(todoNumber: number) {
        await element(by.xpath("//input[@data-id='input.checkbox.done-" +
todoNumber + "']")).click();
    }
```

```
    async selectCatagory(catagory: string) {
        await element(by.xpath("//select[@data-id='select.category']")).click();
        await element(by.css('option[value=' + this.categories[catagory] +
']')).click();
    }
}
```

hooks.ts

```
const { Before, AfterAll } = require('cucumber');
import { browser } from 'protractor';
import { config } from '../../conf.js';

Before({ timeout: 100 * 1000 }, async () => {
   browser.get(config.baseUrl);
});

AfterAll({ timeout: 600 * 1000 }, async () => {
   browser.driver.quit();
});
```

protractor.conf.js

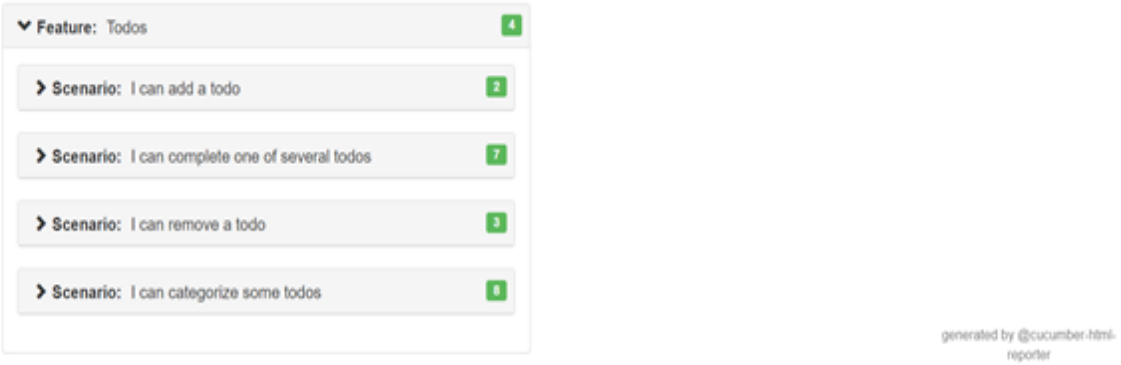
```
const Reporter = require("./src/support/reporter");
exports.config = {
  directConnect: true,
  baseUrl: 'http://localhost:9090/',
  seleniumAddress: 'http://127.0.0.1:4444/wd/hub',
  getPageTimeout: 60000,
  allScriptsTimeout: 500000,
  framework: 'custom',
  frameworkPath: require.resolve('protractor-cucumber-framework'),
  capabilities: {
    acceptInsecureCerts: true,
    'browserName': 'chrome',
    chromeOptions: {
      args: ["--disable-gpu", "--window-size=600,300" ]
    }
  },

  specs: ['src/features/Todos.feature'],
  onPrepare: async function () {
```

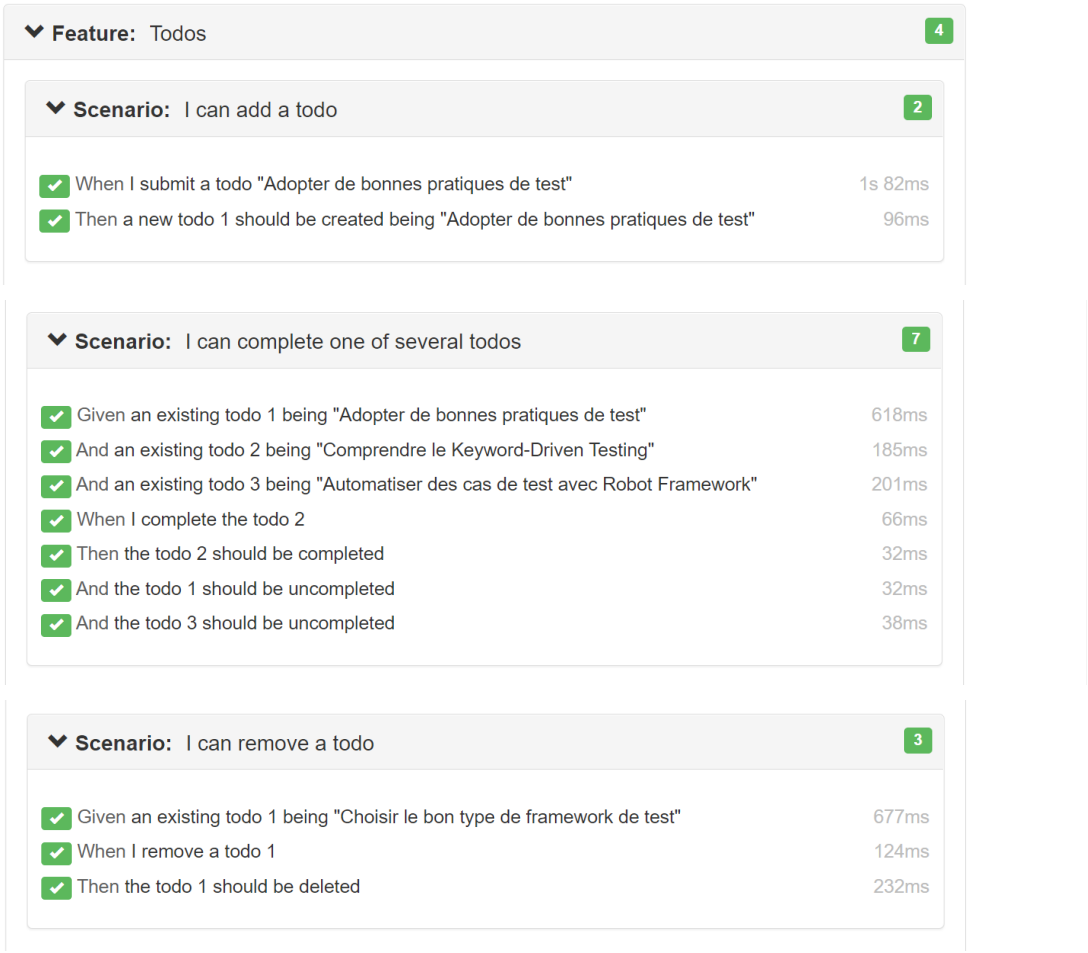
```
    browser.ignoreSynchronization = true;
    await browser.manage().window().maximize();
    browser.allScriptsTimeout = 10000;
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.e2e.json')
    });
  },
  cucumberOpts: {
    compiler: "ts:ts-node/register",
    format:
["json:./e2e/reports/json/cucumber_report.json","node_modules/cucumber-pretty"],
    tags: ["~@ignore"],
    require:
      [
        "src/support/*.ts",
        "src/steps-definitions/*.ts"
      ],
    strict: true
  },
  onComplete: () => {
    Reporter.createHTMLReport()
  }
}
```

Le dépôt du code source est disponible ici.

Après avoir exécuté les tests, CucumberJs génère le rapport ci-après.



Et il est facile de consulter le niveau de détail souhaité simplement en dépliant les étapes des tests dans le rapport.



Scenario: I can categorize some todos 8

When I submit a todo "Choisir un livre intéressant"	332ms
Then the todo 1 should not be categorized	246ms
When I submit a "Personnel" todo "Marcher et faire du vélo avec mon chien"	301ms
Then the todo 2 should be "Personnel"	41ms
When I submit a todo "Faire un câlin avec mon chat"	159ms
Then the todo 3 should be "Personnel"	50ms
When I submit a "Professionnel" todo "Automatiser un cas de test de plus"	366ms
Then the todo 4 should be "Professionnel"	64ms

Besoin d'aide pour mettre en place ces frameworks et former vos collaborateurs ?

Il est important de comprendre les principes de fonctionnement de ces frameworks, ainsi que leur pertinence et impact pour votre organisation. Or leur mise en place est souvent complexe. Nous sommes donc à votre disposition pour vous aider dans le choix et l'implémentation d'un cadre méthodologique et des outils appropriés, en vous accompagnant dans vos projets. Outre l'automatisation des tests, nous pouvons vous aider à adopter une approche globale de la qualité pour vos futurs produits. Grâce à notre accompagnement, vous gagnerez un temps précieux comme en compétitivité.

All4Test

All4Test est un *pure player* du test et de la qualité logiciel en France depuis 2006, présent à Paris, Sophia-Antipolis, Bordeaux et Tunis. Nous intervenons en audit, formation, AT ou projet d'externalisation des tests (TRA).

CONTACT@ALL4TEST.COM

CHRYSO CODE

Chrysocode est un cabinet de conseil spécialisé en stratégie IT (augmentation des capacités opérationnelles des organisations) et dans la conduite du changement en ingénierie logicielle. Son ambition est de rendre à l'Ingénierie du Code ses lettres de noblesse, en libérant les potentiels humains et les savoir-faire. Découvrez la proposition de valeur de Chrysocode à travers son pitch deck.

CONTACT@CHRYSOCODE.IO

Nous collaborons ensemble pour construire une offre complète centrée sur la *Quality By Design*, une approche globale de la qualité, dont l'objectif est d'accompagner les clients sur les thèmes du "Mieux Spécifier", "Mieux Développer", "Mieux Tester".

MIEUX_SPÉCIFIER MIEUX_DÉVELOPPER MIEUX_TESTER