

# Génie Logiciel

Travaux pratiques de conception,  
implémentation et automatisation de tests  
fonctionnels

Mise en œuvre d'une chaîne outillée Yest – Selenium Java avec Robot  
Framework

Abbas AHMAD – Bruno LEGEARD

[abbas.ahmad@univ-fcomte.fr](mailto:abbas.ahmad@univ-fcomte.fr)

423 C

# Introduction

## Combinaison de TP/TD sur 2 semaines

- Semaine 1 : 22-26/11/21
- Semaine 2 : 29/11 – 03/12 2021

## Semaine 1

- 1 TD: 1h30
- 2 TP: 3h
- Total: 4h30

## Semaine 2

- 2 TD: 3h
- 2 TP: 3h
- Total: 6h

# Introduction

## Travail en binôme

- 3 groupes TP (A/B/C) et 2 TD (1/2)
- Grouper vous avec camarades de TP et TD quand c'est possible

## 2 Exercices, un pour chaque semaine

- Semaine 1: Assurance Auto
- Semaine 2: eShop

## Travail à rendre

- Uniquement sur exercice 2 (eShop)
- Date limite rendu: vendredi 10/12/21 à 22h
- Dépôt et modalité sur Moodle

# TD: Tests piloté par mots clés (Keyword-Driven Testing « KDT »)

```
//Fields
/// <summary>
/// Go to web page 'http://TestSite/login.aspx' using new
browser instance
/// </summary>
public string UIWelcomeToTestSiteWinWindowUrl
= "http://TestSite/login.aspx";

/// <summary>
/// Type 'test-user' in 'txtUserName' text box
/// </summary>
public string UITxtUserNameEditText = "test-user";

/// <summary>
/// Type '{Tab}' in 'txtUserName' text box
/// </summary>
public string UITxtUserNameEditSendKeys = "{Tab}";

/// <summary>
/// Type '*****' in 'txtPassword' text box
/// </summary>
public string UITxtPasswordEditPassword =
"toVpCSU2LKdInH9vdzPA0ZmKuc60K";

/// <summary>
/// Type '{Enter}' in 'txtPassword' text box
/// </summary>
public string UITxtPasswordEditSendKeys = "{Enter}";

//Actions
/// Go to web page the webpage using new browser instance
this.UIWelcomeToTestSiteWinWindow.LaunchUrl(new
System.Uri(this.LoginParams.UIWelcomeToTestSiteWinWindowU
rl));

/// Type 'test-user' in 'txtUserName' text box
UITxtUserNameEditText =
this.LoginParams.UITxtUserNameEditText;

/// Type '{Tab}' in 'txtUserName' text box
Keyboard.SendKeys(UITxtUserNameEditText,
this.LoginParams.UITxtUserNameEditSendKeys,
ModifierKeys.None);

/// Type '*****' in 'txtPassword' text box
UITxtPasswordEdit.Password =
this.LoginParams.UITxtPasswordEditPassword;

/// Type '{Enter}' in 'txtPassword' text box
Keyboard.SendKeys(UITxtPasswordEdit,
this.LoginParams.UITxtPasswordEditSendKeys,
ModifierKeys.None);
```

```
[TestMethod]
public void Login_TestMethod()
{
    WaitIn.Core.Settings.WaitForCompleteTimeout = 120;
    IE ie = new IE("http://TestSite/login.aspx", true);
    ie.TextField(FindById("txtUserName")).Value = "test-
user";
    ie.TextField(FindById("txtPassword")).Value =
"12345678";
    ie.Button(Find.ById("btnLogin")).Click();
    ie.WaitForComplete();
    // If "Welcome" message is displayed, then the test
is passed
    if (ie.Text.Contains("Welcome"))
    {
        Console.WriteLine("Testing Passed");
    }
    else
    {
        //If not, then the test is failed
        Console.WriteLine("Testing Failed");
    }
}
```

```
1 using Xunit;
2 using Xunit.Extensions;
3
4 namespace ExcelDataDrivenTests
5 {
6     public class ExcelDataDrivenTests
7     {
8         [Theory]
9         [ExcelData("ExcelDataSource.xls", "Select * from TestData")]
10        public void SampleTest1(int number, bool expectedResult)
11        {
12            var sut = new CheckThisNumber(1);
13            var result = sut.CheckIfEqual(number);
14            Assert.Equal(result, expectedResult);
15        }
16    }
17 }
```

```
*** Settings ***
Suite Setup      Open Browser To Login Page
Suite Teardown   Close Browser
Test Setup       Go To Login Page
Test Template     Login With Invalid Credentials Should Fail
Resource         resource.txt

*** Test Cases ***
Invalid Username    invalid    ${VALID USER}    ${VALID PASSWORD}
Invalid Password    invalid    ${VALID USER}    invalid
Invalid Username And Password    invalid    whatever
Empty Username      ${EMPTY}    ${VALID PASSWORD}
Empty Password      ${VALID USER}    ${EMPTY}
Empty Username And Password    ${EMPTY}    ${EMPTY}

*** Keywords ***
Login With Invalid Credentials Should Fail
[Arguments]    ${username}    ${password}
Input Username    ${username}
Input Password    ${password}
Submit Credentials
Login Should Have Failed

Login Should Have Failed
Location Should Be    ${ERROR URL}
Title Should Be      Error Page
```

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

*** Variables ***
${SERVER}         localhost:7272
${BROWSER}        Firefox
${DELAY}          0
${VALID USER}     demo
${VALID PASSWORD} mode
${LOGIN URL}       http://${SERVER}/
${WELCOME URL}     http://${SERVER}/welcome.html
${ERROR URL}       http://${SERVER}/error.html

*** Keywords ***
Open Browser To Login Page
Open Browser       ${LOGIN URL}    ${BROWSER}
Maximize Browser Window
Set Selenium Speed    ${DELAY}
Login Page Should Be Open

Login Page Should Be Open
Title Should Be      Login Page

Go To Login Page
Go to              ${LOGIN URL}
Login Page Should Be Open

Input Username
[Arguments]         ${username}
Input Text          username_field    ${username}

Input Password
[Arguments]         ${password}
Input Text          password_field    ${password}

Submit Credentials
Click Button        login_button

Welcome Page Should Be Open
Location Should Be    ${WELCOME URL}
Title Should Be      Welcome Page
```

Linéaire

Structuré

Piloté par les  
données

Piloté par les  
Mots Clés

Dirigé par  
processus

Maturité du Scripting

# Tests piloté par les mots clés

## En anglais : Keyword-Driven Testing (KDT)

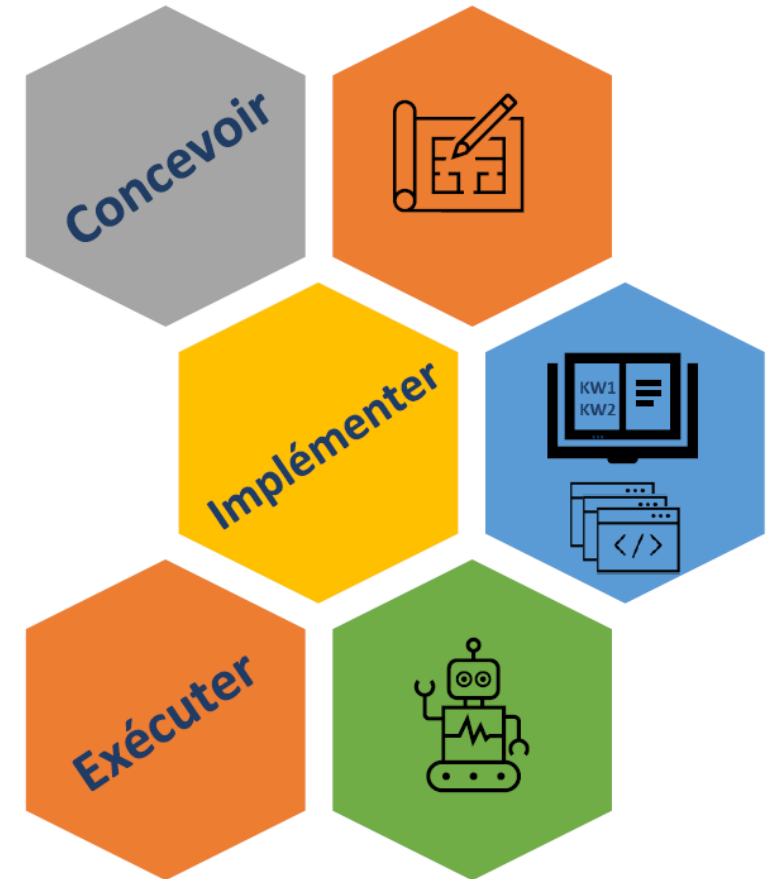
- C'est l'une des approches les plus répandues pour l'automatisation des tests. Le Keyword Driven Testing sépare la conception des tests de la partie implémentation et exécution des tests.

## Concept

- « Keyword-DrivenTesting » se repose sur l'utilisation des mots-clefs qui sont définis de façon conjointe entre le concepteur de test et l'automaticien, habituellement liés à l'application sous test ou son domaine, afin de construire un dictionnaire de procédures automatisées.

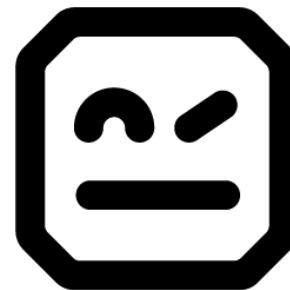
## Motivation

- Scripts de tests plus lisibles
- Scripts de tests compréhensibles
- Effort de maintenance diminué




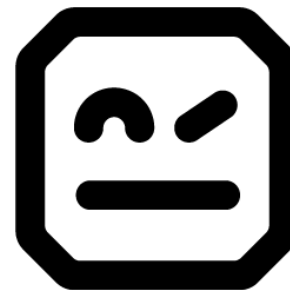
# Tests piloté par les mots clés

Avantages	Inconvénients
Cout d'ajout de nouveaux tests réduit (Dès lors que l'implémentation des mots-clés est faite)	L'implémentation des mots clés demande du travail au automaticiens de tests
Autonomie relative des Analyste de tests pour écrire de nouveaux tests	Peu adapté à des « petites » applications (rentabilité)
Définition d'actions de haut niveau (ex: « Check_order_status »)	Risque d'implémenter des mots clés qui ne seront peu ou pas utilisés (cout)
Cas de tests plus facile à maintenir	
Complexité cachée dans les mots clés	



# Robot Framework, informations utiles

- **Python (PREREQUIS)**, toutes les informations ici sont données sur la base d'une utilisation sous WINDOWS.
- Robot Framework instructions d'installation :  
<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#installation-instructions>
- Documentation librairie Selenium  
<https://robotframework.org/Selenium2Library/Selenium2Library.html>
- Avant d'exécuter les tests, télécharger le webdriver Firefox [geckodriver]:  
<https://github.com/mozilla/geckodriver/releases> et positionner le a la racine de votre projet ou ajouter le au PATH.



# Robot Framework, informations utiles

Robot Framework installation  
avec pip

- <https://pypi.org/project/robotframework/>

Selenium2Library installations  
avec pip

- <https://pypi.org/project/robotframework-selenium2library/>

Pour exécuter des tests Robot  
en ligne de commande, étant  
à la racine du projet

- ***python -m robot -d results .***





# Robot Framework KDT

- Robot Framework est un framework de test automatique en **python** pour concevoir et exécuter des tests de validation.
- Avec la bibliothèque **Selenium2Library** (bibliothèque de Robot Framework), écrire des tests pour des applications Web est aisé :

```
test.robot
1 *** Settings ***
2 Library Selenium2Library
3
4 *** Test Cases ***
5 TC_Example
6 Open Browser      https://www.google.com firefox
7 Maximize Browser Window
8 Wait Until Element Is Visible  xpath=//*[@id="L2AGLb"]
9 Click Button      xpath=//*[@id="L2AGLb"]
10 Input Text       xpath=/html/body/div[1]/div[3]/form/div[1]/div[1]/div[1]/div/div[2]/input  Génie Logiciel
11 Click Button      xpath=//*[@id="gbqfbb"]
12 Close Browser
```



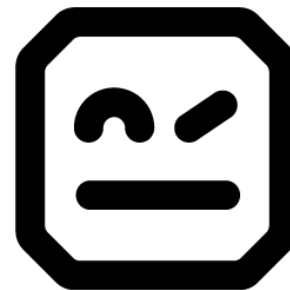
- Avec ce cas, le test (fichier test.robot) ouvre la page Google dans Chrome et recherche le texte « Génie Logiciel » et appuie sur le bouton Chercher.



# Robot Framework KDT

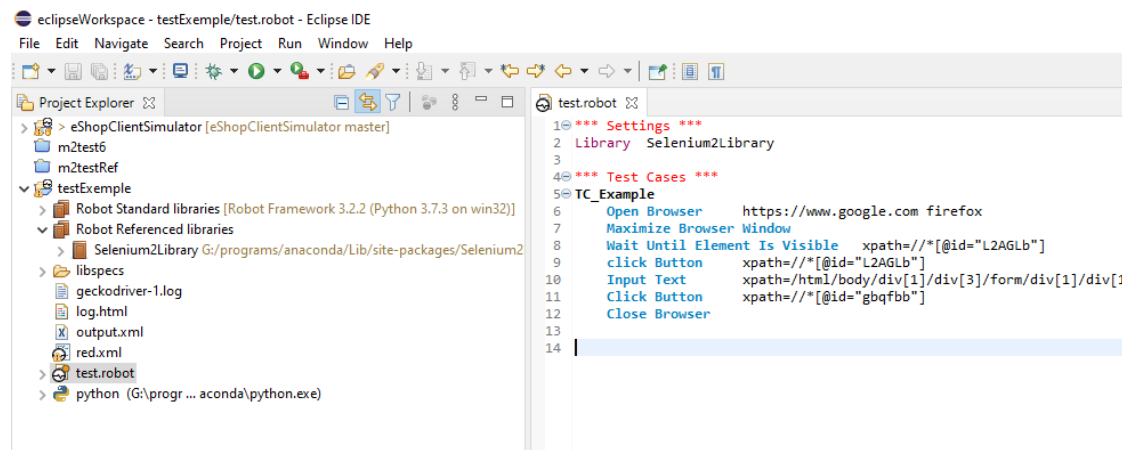
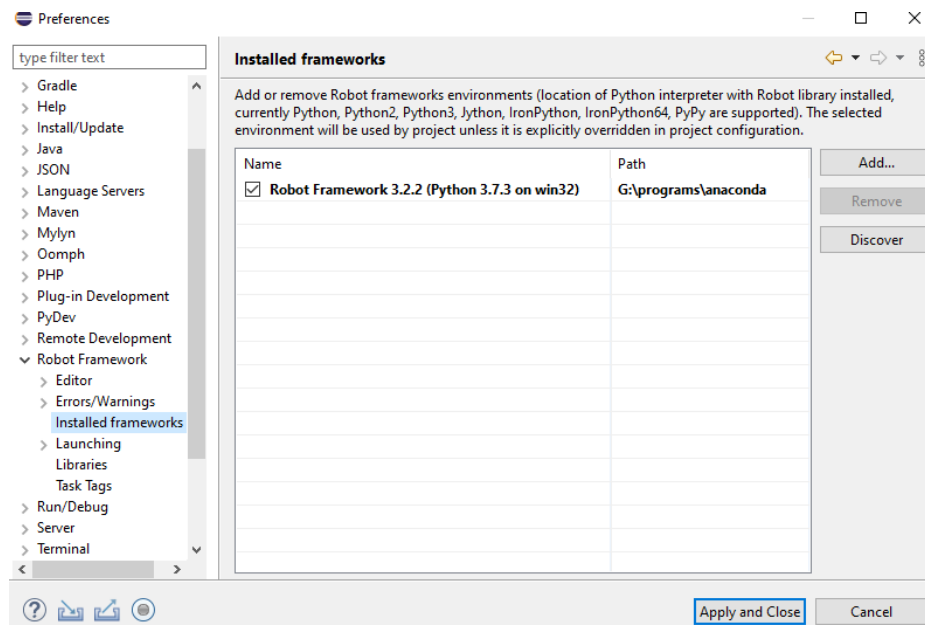
- Résultats en ligne de commande:

```
G:\Dropbox\ENSEIGNEMENT\GL_COMP\Automatisation de tests M1\TP-TD\robot>python -m robot -d results .
=====
Robot
=====
Robot.Test
=====
TC_Example                                     | PASS |
-----
Robot.Test                                     | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
Robot                                     | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
Output:  G:\Dropbox\ENSEIGNEMENT\GL_COMP\Automatisation de tests M1\TP-TD\robot\results\output.xml
Log:     G:\Dropbox\ENSEIGNEMENT\GL_COMP\Automatisation de tests M1\TP-TD\robot\results\log.html
Report:  G:\Dropbox\ENSEIGNEMENT\GL_COMP\Automatisation de tests M1\TP-TD\robot\results\report.html
```



# Robot Framework KDT

- Vous pouvez aussi utiliser l'IDE de votre choix.
- Avec Eclipse: <https://marketplace.eclipse.org/content/red-robot-editor>



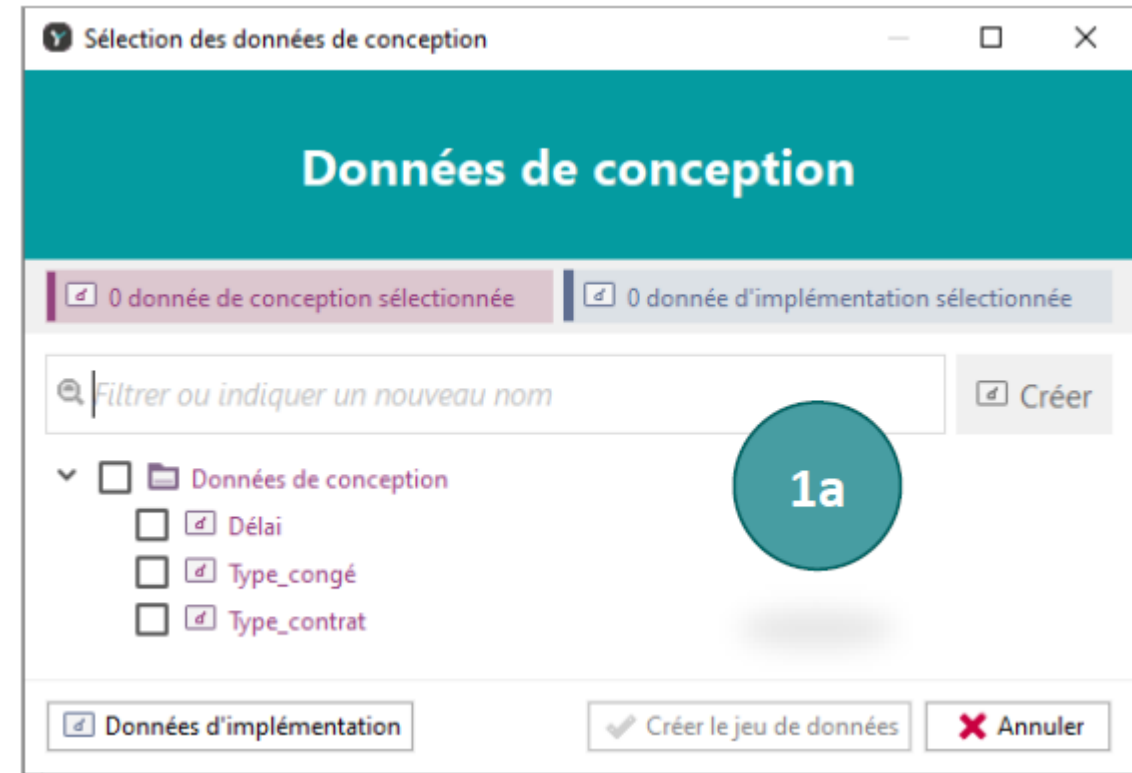
# TD: Automatisation de tests avec Yest

Indépendamment de l'exécution des tests (manuelle ou automatisée), la **conception et l'implémentation visuelle des tests** dans cette étape reste une approche itérative :

- **Construire** les ensembles de jeux de données (JDD) pour l'exécution des tests
- **Gérer** les combinaisons de jeux de données pour l'exécution des tests
- **Utiliser** les données d'implémentation et les jeux de données dans les étapes de test (pour une exécution manuelle ou automatisée)
- **Publier** les résultats

# TD: Construire les ensembles de JDD

- Bien faire la différence entre données de conception et d'implémentation pour **faciliter la maintenance**.
- Les **données de conception (1a)** représentent des **valeurs logiques** qui sont utilisées pour concevoir les tests. Elles représentent généralement des partitions d'équivalence.
- Les données de conception ne sont pas susceptibles de changer en raison des contextes d'exécution des tests et en gardent l'indépendance. Elles représentent souvent des valeurs fonctionnelles.



# TD: Construire les ensembles de JDD

- Les **données d'implémentation (1b)** sont proches des **données physiques** utilisées lors de l'exécution des tests, bien qu'elles puissent encore conserver un certain niveau intermédiaire.
- Par exemple une date de demande dans les 30 jours, au lieu d'une date précise, pour conserver l'indépendance des dates de calendrier obsolètes.

Sélection des données d'implémentation

## Données d'implémentation

0 donnée de conception sélectionnée | 1 donnée d'implémentation sélectionnée

1b

Filtrer ou indiquer un nouveau nom [Créer]

Données d'implémentation

- ☐ Age
- ☐ Date
- ☐ Navigateur
- ☐ Nom
- ☐ Prénom
- ☐ Sexe

[Données de conception] [Créer le jeu de données] [Annuler]

# Bonnes pratiques – gestion des combinaisons de JDD

La production des combinaisons des jeux de données pour un scénario de test peut se faire en choisissant l'une des options de génération suivantes:

- Générer une combinaison aléatoire
- Générer toutes les combinaisons possibles (notre choix ici)

Cela produira toutes les combinaisons pour chaque test

- Il est possible d'ajouter des filtres lors de la génération (ex.Browser=Chrome)

# Bonnes pratiques –gestion des combinaisons de JDD

Les ensembles de jeux de données peuvent être **incohérents** par rapport aux conditions que vous avez dans le parcours applicatif.

- Pour celles-ci, Yest ne produira aucune combinaison

Si une combinaison existante devient incohérente au cours d'une évolution (modification du parcours applicatif ou des règles métier)

- Une notification avertira l'utilisateur en pointant vers la combinaison devenue incohérente (vous pouvez essayer de le faire en supprimant la ligne)

Repère: utilisez le panneau «Détails des scripts» pour savoir si vous avez bien défini les descriptions détaillées pour toutes les étapes de test de votre campagne.



# Exercice 1: Assurance Auto

Accès : <https://demo-simulator.herokuapp.com/>

## Tâches

- Télécharger le projet Yest Assurance Auto depuis Moodle, créer une campagne de test pour les champ d'application donné ci-après
  - Le projet est doté de données de conception et d'implémentation ainsi que d'un jeux de données.
- Publier la campagne de test pour une exécution de tests automatisés dans votre environnement Robot Framework
- Automatiser les mots clefs pour les exécuter sur le simulateur en ligne

# Exercice 1: Assurance Auto.

## Description du périmètre de test

- Une compagnie d'assurance française souhaite développer un nouveau portail de simulation pour son offre d'assurance automobile.
- Exigences
  - Un écran doit permettre de saisir les informations relatives au véhicule à assurer.



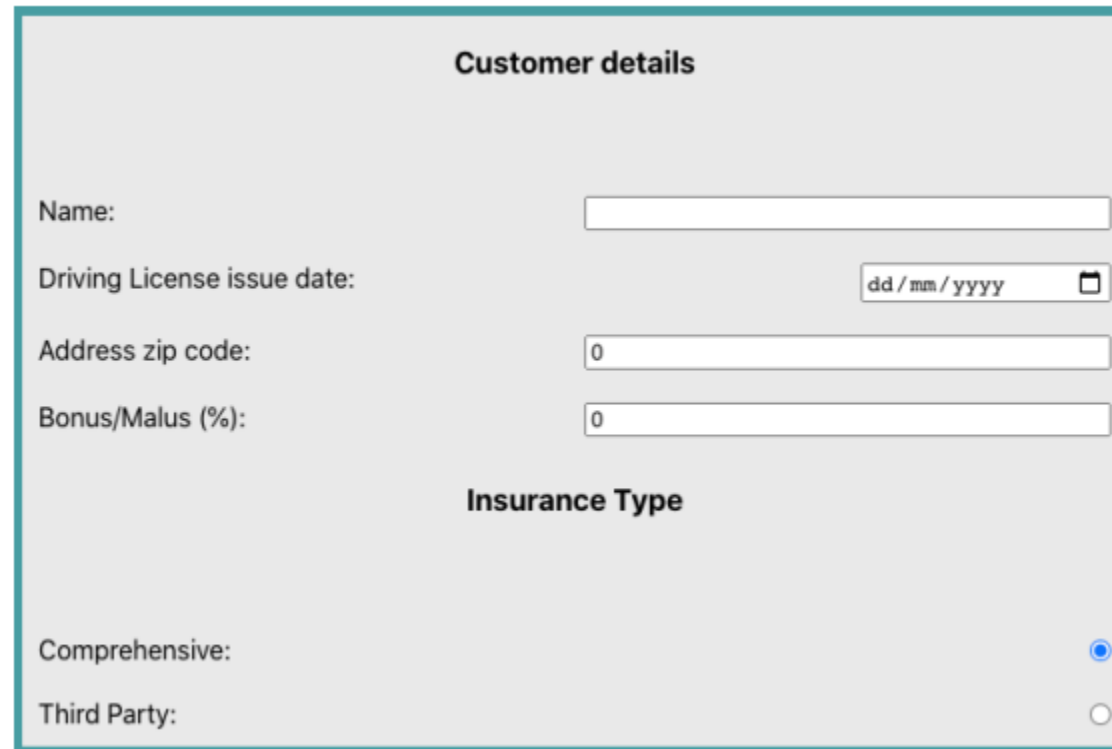
The image shows a web form titled "Car details" with a light gray background and a teal border. It contains five input fields arranged vertically on the left, with their corresponding values or options on the right:

- Car Brand:** A dropdown menu showing "Peugeot".
- Car Model:** A dropdown menu showing "308".
- Fuel Type:** A dropdown menu showing "Petrol".
- New car:** A checkbox that is currently unchecked.
- Year of first use:** A text input field containing the value "2020".

# Exercice 1: Assurance Auto.

## Description du périmètre de test

- Un écran permet de saisir les informations relatives au client (conducteur du véhicule).



The screenshot shows a web form titled "Customer details". It contains the following fields:

- Name:** A text input field.
- Driving License issue date:** A date input field with a placeholder "dd/mm/yyyy" and a calendar icon.
- Address zip code:** A text input field with the value "0".
- Bonus/Malus (%):** A text input field with the value "0".

Below these fields is a section titled "Insurance Type" with two radio button options:

- Comprehensive:** A radio button that is currently selected.
- Third Party:** An unselected radio button.

# Exercice 1: Assurance Auto.

## Description du périmètre de test

- Après la validation du véhicule, s'il est retrouvé dans la base de données, la cotisation annuelle d'assurance sera affichée. Dans le cas contraire, le simulateur affichera un message.

La cotisation annuelle d'assurance dépend de	
Type d'assurance	Cotisation de base pour "Assurance tous risques" = 120% x Cotisation de base pour "Assurance aux tiers"
Date du permis de conduire	Si le permis de conduire date de 3 ans ou plus, le montant total de la cotisation reste inchangé ("100% BASE")
	Si le du permis de conduit date de moins de 3 ans, alors le montant total de la cotisation est augmenté de 10% ("110% BASE")
Lieu de résidence du conducteur	Si le lieu de résidence est "Paris" alors le montant total de la cotisation est incrémenté de 20% ("120% BASE")
	Si le lieu de résidence est "Province" alors le montant total de la cotisation reste inchangé ("100% BASE")
Bonus / Malus	Le prix total est multiplié par le pourcentage de bonus/malus. Plus de 0% for malus, alors le prix total est augmenté (e.g. 10% malus = 110% COTISATION). Plus de 0% for bonus, alors le prix total est diminué (e.g. 15% bonus = 85% COTISATION)

# Exercice 1: Assurance Auto

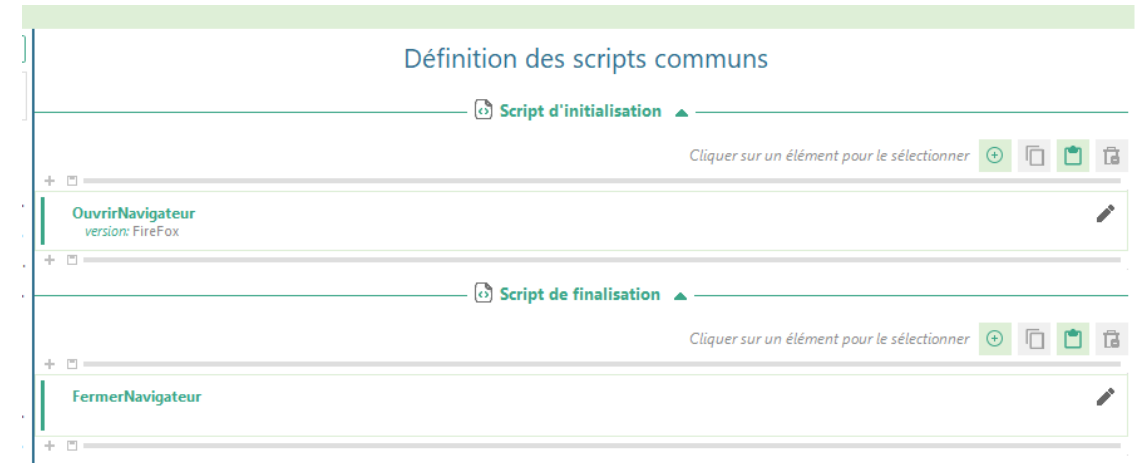
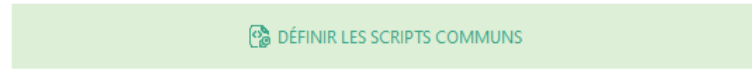
## Conseil de progression itérative et incrémentale

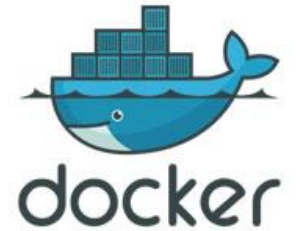
1. Faire un projet basique dans Yest qui représente un scénario de bout en bout pour l'assurance auto
  - Bien faire une distinction **Données de conception** vs **Données d'implémentation**
2. Définir un ensemble de mot clefs en Robot Framework
  - Ajouter une implémentation basique par exemple « Not yet implemented »
3. Créer une campagne de Yest et sa publication associée en RobotFramework
4. Mapper vos mots clefs aux étapes de tests et produire le scénario en RobotFramework
5. Exécuter le scénario → SUCCESS
6. Maintenant vous pouvez faire d'autres itérations
  - Au niveau de l'implémentation de vos mots clefs
    - Implémenter les mots clefs
    - Raffiner vos mots clefs (mettez en pratique vos capacités de développement, si nécessaire ajouter des paramètres, refactoriser des mots clefs pour réutiliser du code etc.)
  - Au niveau de la conception des scénarios dans Yest
    - Pour compléter les règles métier
    - Incrémenter les jeux de données

# Exercice 1: Assurance Auto

## Conseils utiles

- Définir des scripts commun

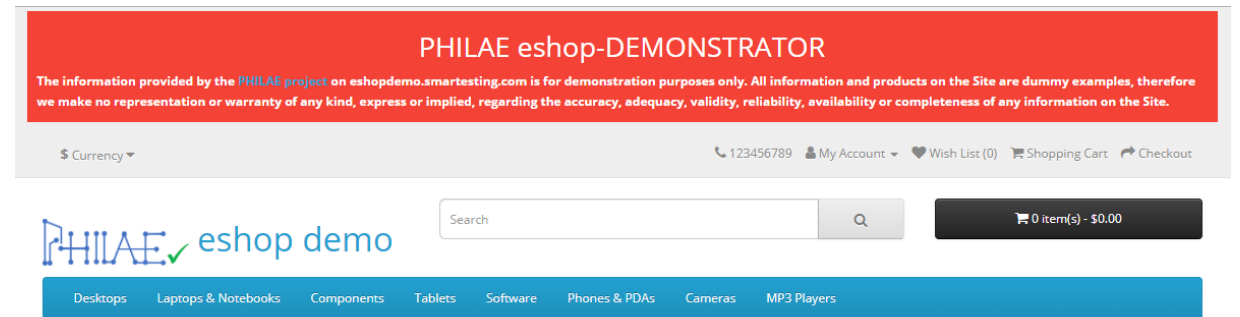




# TD: Introduction à Docker

- Installation sur vos machines: <https://www.docker.com/get-started>
- Docker compose est préinstallé sur Windows.
  - Installation pour linux : <https://docs.docker.com/compose/install/>
- **Définition:** Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels.
- Il ne s'agit pas de virtualisation, mais de conteneurisation, une forme plus légère qui s'appuie sur certaines parties de la machine hôte pour son fonctionnement.
- => Démo

# Exercice 2: eShop



- L'application eShop se base sur un projet open source « OpenCart »
- OpenCart est un logiciel de gestion de Boutique en ligne.
- Il est basé sur PHP, utilisant une base de données MySQL et des composants HTML.
- eShop est une instantiation de OpenCart avec des données exemples préconfiguré
- Fruit d'un travail du projet « PHILAE »: <https://projects.femto-st.fr/philae/en>



# Exercice 2: eShop, A FAIRE et a RENDRE

- Code source de l'application « dockerisé »:  
<https://github.com/PHILAE-PROJECT/eShopDemoDocker>
- 1. Lancement en local a l'aide d'un script Docker-compose
  - Voir slides suivantes.
- 2. Créer le projet Yest, les Jeux de données et des campagne de tests
- 3. Publication Robot Framework et exécution des tests automatisés

# Exercice 2 : eShop, informations Utiles

- Si vous n'avez pas d'ordinateur personnel venez me voir pour la procédure a suivre sur machine de l'université
- Procédures qui suivent sont conçue pour machine personnelle sous **Windows**.
- Pour tout autre OS appeler moi si vous avez des difficultés à adapter.

# Exercice 2 : eShop, informations Utiles

- Etapes de lancement du SUT eShop:
  1. Prérequis: Docker installé sur votre machine.
  2. Cloner le repo eShop GitHub : <https://github.com/PHILAE-PROJECT/eShopDemoDocker>
  3. Accéder a votre projet (au niveau du fichier docker-compose.yml )
  4. **IMPORTANT** modifier les ports par default dans le docker-compose si ils sont utilisé sur votre machine

```
web:
  image: nginx
  ports:
    - "8080:80"
```

```
aahmad@cr700-docker:~$ git clone https://github.com/PHILAE-PROJECT/eShopDemoDocker.git
Clonage dans 'eShopDemoDocker'...
remote: Enumerating objects: 6125, done.
remote: Counting objects: 100% (6125/6125), done.
remote: Compressing objects: 100% (4380/4380), done.
remote: Total 6125 (delta 1684), reused 5946 (delta 1515), pack-reused 0
Réception d'objets: 100% (6125/6125), 16.57 MiB | 13.33 MiB/s, fait.
Résolution des deltas: 100% (1684/1684), fait.
Extraction des fichiers: 100% (5073/5073), fait.
aahmad@cr700-docker:~$ ls
dfsdds.txt  eShopDemoDocker
aahmad@cr700-docker:~$ cd eShopDemoDocker/
aahmad@cr700-docker:~/eShopDemoDocker$ ls
docker-compose.yml  LICENSE  Makefile  phpcli-img  php-img  README.md  sites  sql  www
aahmad@cr700-docker:~/eShopDemoDocker$
```

# Exercice 2 : eShop, informations Utiles

- Lancer l'application a l'aide de la commande : *docker-compose up*

```
G:\Docker\eShopDemoDocker>docker-compose up
Creating network "eshopdemodocker_default" with driver "bridge"
Creating eshopdemodocker_mariadb_1 ... done
Creating eshopdemodocker_php_1 ... done
Creating eshopdemodocker_composer_1 ... done
Creating eshopdemodocker_web_1 ... done
Attaching to eshopdemodocker_mariadb_1, eshopdemodocker_composer_1, eshopdemodocker_php_1, eshopdemodocker_web_1
composer_1 | Installing dependencies from lock file (including require-dev)
composer_1 | Verifying lock file contents can be installed on current platform.
composer_1 | Nothing to install, update or remove
mariadb_1 | 2021-11-19 10:57:17+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.6.4+maria~f
composer_1 | Generating autoload files
mariadb_1 | 2021-11-19 10:57:17+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
```

- Accéder a l'application depuis un navigateur web: localhost:8080

# Exercice 2 : eShop, informations Utiles

- Pour fermer l'application proprement: *ctrl+c* puis entrer la commande: *docker-compose down*

```
mariadb_1 | 2021-11-19 10:57:26 0 [Note] mysqld: ready for connections.  
mariadb_1 | Version: '10.6.4-MariaDB-1:10.6.4+maria~focal' socket: '/run/mysqld/mysqld.s  
mariadb_1 | 2021-11-19 10:57:26 0 [Note] InnoDB: Buffer pool(s) load completed at 211119  
php_1      | 172.19.0.5 - 19/Nov/2021:10:57:57 +0000 "GET /index.php" 200  
Gracefully stopping... (press Ctrl+C again to force)  
Stopping eshopdemodocker_web_1      ... done  
Stopping eshopdemodocker_php_1      ... done  
Stopping eshopdemodocker_mariadb_1  ... done  
  
G:\Docker\eShopDemoDocker>docker-compose down  
Removing eshopdemodocker_web_1      ... done  
Removing eshopdemodocker_composer_1 ... done  
Removing eshopdemodocker_php_1      ... done  
Removing eshopdemodocker_mariadb_1  ... done  
Removing network eshopdemodocker_default  
  
G:\Docker\eShopDemoDocker>
```