K-lity

Version 0.2.0

Mindiell

févr. 03, 2022

Contents:

K-lity est un outil permettant d'effectuer des tests web de manière automatisée. Pour cela, il repose sur les composants selenium, qui permet de contrôler le navigateur, et behave, qui permet de lancer des tests par fonctionnalité.

L'idée principale de K-lity est de **faciliter l'écriture et la lecture des tests**, permettant ainsi à des personnes non techniques de participer à ceux-ci.

Les tests sont regroupés par *Fonctionnalité*. Une *Fonctionnalité* est un ensemble de Scénarios qui permettent de valider une fonctionnalité d'un logiciel. Chaque *Scénario* contient un ensemble d'étapes permettant d'exécuter celui-ci afin de valider son bon déroulement. Le langage Gherkin est utiliséet le but est de pouvoir écrire ou lire un *Scénario* en le rendant compréhensible à toute personne, même si celle-ci ne connait pas l'application.

CHAPITRE 1

Démarrage rapide

Démarrage rapide

CHAPITRE 2

Installation

Installation

CHAPITRE 3

Options

Klity se lance en ligne de commande et accepte des options :

3.1 Choix des fonctionnalités

Afin de limiter les fonctionnalités à tester, il peut être intéressant de filtrer celles-ci. Pour cela, l'option –filename (ou -f) est utilisée. Les mots situés après cette option doivent tous être présents dans le nom global du fichier de la fonctionnalité pour que celle-ci soit prise en compte. Si on utilise plusieurs fois cette option, plusieurs fonctionnalités peuvent donc être trouvés indépendamment les uns des autres.

3.2 Choix des scénarios

Afin de limiter les scénarios à exécuter, l'option –name (ou -n) peut être utilisée. Les noms des Scénarios qui correspondent à cette option seront alors les seuls à être exécutés.

3.3 Nettoyage des anciens résultats

Afin d'être sûr que les résultats des tests soient les plus récents, il peut être nécessaire de les supprimer avant de lancer les tests. L'option –clean permet cela.

3.4 Ne pas exécuter les tests

Afin de s'assurer par exemple que les filtres des fonctionnalités sont corrects, il est possible de ne pas exécuter les tests. Il suffit alors d'utiliser l'option –dry-run.

3.5 Ne pas produire de rapport

Il est également possible de ne pas produire le rapport en fin de test. Il suffit alors d'utiliser l'option –no-report.

3.6 Spécifier le titre du rapport

Il est possible de spécifier le titre du rapport en utilisant l'option –title (ou -t).

3.7 Spécifier le template du rapport

Il est possible de spécifier le template du rapport en utilisant l'option –template. Les templates actuellement disponibles dans k-lity sont : « default », « dotmap » et « default_with_logs ».

Il est possible d'utiliser *vos propres templates*.

3.7.1 Démarrage rapide

Afin de lancer un premier test, vous devez créer un répertoire projet. K-lity vous aide via la commande *klity-newproject* :

```
klity-newproject mytest
```

Vous obtenez alors un répertoire appelé mytest avec cette structure :

```
mytest
├─ configuration.yml
├─ environment.py
└─ steps
   └─ project_steps.py
```

- ***configuration.yml* est un fichier de configuration par défaut. Modifiez-le pour** spécifier votre propre configuration.
- *environment.py* est un fichier python pour usage interne, n'y touchez pas.
- *steps* est le répertoire dans lequel vous allez écrire tous vos tests.
- ***project_steps.py* est un fichier python qui contient les étapes spécifiques à votre** projet.

Un premier test

Pour écrire votre test, vous n'avez besoin que d'un éditeur de texte. Créer un fichier first_test.feature dans le répertoire steps et écrivez le test suivant :

```
Fonctionnalité: Premier test avec K-lity

Scénario: Un exemple avec example.org
    Etant donné que je visite le site "https://example.org"
    Alor la page contient "Example Domain"
```

Vous pouvez alors lancer votre test depuis le répertoire parent *my_test* et voir le test se dérouler à votre écran. Logiquement, vous devriez voir Firefox se lancer et rapidement se refermer une fois que le site *example.org* est chargé :

```
> klity
first_test
  Executing tests
```

Maintenant, vous pouvez découvrir qu'un fichier (geckodriver.log) et trois répertoires ont été créés :

- *geckodriver.log* contient les logs fournis par geckodriver.
- *report* contient un rapport sur vos tests.
- ***results* contient les résultats de toutes les commandes behave exécutées pour** chacune de vos fonctionnalités.Lors de la création des tests, ces fichiers sont très importants car ils peuvent vous aider à comprendre ce qui se passe.
- *screenshots* contient les captures d'écran de vos tests.

3.7.2 Installation

Pour installer K-lity, il suffit d'utiliser pip :

```
pip install k-lity
```

Si vous souhaitez utiliser postgresql lors de vos tests, il est nécessaire de le signaler lors de l'installation :

```
pip install k-lity[postgresql]
```

Pour le moment, seul postgresql est supporté.

Mise à jour

Pour mettre K-lity à jour, là encore pip est suffisant :

```
pip install -U k-lity
```

Geckodriver

Actuellement, seul Firefox est géré par K-lity. Il est donc nécessaire d'installer un outil pour piloter celui-ci. C'est l'outil geckodriver qui est utilisé ici. Il faut bien entendu ajouter le chemin de l'exécutable au PATH de l'utilisateur pour le rendre accessible.

Remarques

— Seul Firefox est utilisé pour le moment
— Seules les bases postgresql sont utilisables pour le moment

Installation pas à pas sous Windows

Installation et configuration de Python

Il est d'abord nécessaire d'installer une version récente de Python. Commencez par télécharger la dernière version de Python disponible.

Attention : K-lity est compatible avec les version de Python 3.7 et supérieur.

Vous devriez pouvoir maintenant ouvrir une fenêtre de commande et exécuter les commandes suivantes :

```
python --version
```

qui devrait vous renvoyer quelque chose comme :

```
Python 3.9.6
```

et

```
pip --version
```

qui devrait vous renvoyer quelque chose comme :

```
pip 18.1.6 from [un chemin] (python3.9)
```

Quelle que soit la version de *pip*, il est toujours intéressant d'utiliser sa dernière version. Pour cela, mettez-le à jour :

```
python -m pip install -U pip
```

Vous devriez maintenant pouvoir installer et utilier K-lity en reprenant *les étapes d'installation*.

3.7.3 Contribuer

K-lity est un projet libre (licence APGL). Tout le monde peut contribuer de bien des manières :
— en l'utilisant tout d'abord, et en remontant les éventuels problèmes rencontrés
— en le testant
— en aidant à sa documentation
— en développant également
En cas de besoin, vous pouvez joindre l'équipe via IRC sur le canal #k-lity.

3.7.4 Utiliser Klity

Les étapes

Chaque étape est composée d'une phrase pouvant comporter une ou plusieurs variables qui doivent systématiquement être entourée de guillemets (pour des raisons techniques et de lisibilité).

Les étapes sont rendues sous trois types principaux :
- *Les étapes de type "Etant donné"* : Ce type d'étape permet de définir un état de départ pour chaque test
- *Les étapes de type "Quand"* : Ce type d'étape permet de définir un enchainement d'actions à réaliser
- *Les étapes de type "Alors"* : Ce type d'étape permet de définir un état de sortie pour chaque test

Et un sous type spécifique :
- *Etapes utilitaires* : Ces étapes sont utilisables à n'importe quel niveau

Rédiger un test

Lorsque plusieurs étapes s'enchainent, et pour la simplicité de lecture, il est possible de remplacer le mot clef par « **Et** ». Par exemple :

```
Etant donné que je visite le site "https//example.org/"
Quand je clique sur le bouton "Aide"
Et que je tape "toto" dans le champ "titi"
Et que je coche la case à cocher "Monsieur"
Et que je sélectionne la valeur "Oui" du champ "Acceptez-vous"
Et que je clique sur le bouton "Appeler"
Alors la page contient "Bonjour"
Et la page contient "Comment allez-vous ?"
```

Enfin, chaque étape peut avoir quelques subtilités à connaitre pour bien comprendre comment tel ou tel élément de la page est sélectionné. Dans tous les cas, la sélection recherche au plus large, et une fois la liste des éléments potentiels faite, elle prend en compte uniquement le premier élement visible.

La sélection des éléments est détaillée pour chaque étape au niveau de la section « Subtilités de sélection ».

Les étapes de type "Etant donné"

Se rendre à une url précise

Vocabulaire

```
que je visite le site "{url}"
que je visite l'url "{url}"
```

Exemples

```
Etant donné que je visite le site "https://toto.fr/"
Etant donné que je visite l'url "https://toto.fr/"
```

Quand l'utiliser

On utilise cette étape lorsqu'on souhaite afficher une page spécifique.

Les étapes de type "Quand"

Cliquer sur un bouton

Vocabulaire

```
je clique sur le bouton "{valeur}"
que je clique sur le bouton "{valeur}"
```

Exemples

```
Quand je clique sur le bouton "Enregistrer"
Et que je cliquer sur le bouton "Sauver"
```

Quand l'utiliser

Le cas le plus simple reste le clic sur un bouton, mais parfois les images ou les liens sont représentés sous la forme de bouton. Dans ce cas, il est aussi simple d'utiliser cette étape.

Subtilités de sélection

1. En premier lieu, tous les éléments dont l'**id** vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** des éléments.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le **title** des éléments de type *button*, *input*, *img* ou *a* uniquement.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur la **value** des éléments de type *button*, *input*, *img* ou *a* uniquement.
5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des éléments de type *button*, *input*, *img* ou *a* uniquement.

Cliquer sur un lien

Vocabulaire

```
je clique sur le lien "{valeur}"
que je clique sur le lien "{valeur}"
```

Exemples

```
Quand je clique sur le lien "Enregistrer"
Et que je cliquer sur le lien "Sauver"
```

Quand l'utiliser

Contrairement à l'étape précédente, celle-ci ne vise que les liens.

Subtilités de sélection

1. En premier lieu, tous les **a** dont l'**id** vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** des **a**.
3. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **title** des **a**.
4. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le texte contenu des **a**.

Cliquer sur un élément

Vocabulaire

```
je clique sur l'élément contenant "{valeur}"
que je clique sur l'élément contenant "{valeur}"
```

Exemples

```
Quand je clique sur l'élément contenant "49000 - Angers"
Et que je cliquer sur l'élément contenant "35000 - Rennes"
```

Quand l'utiliser

Cette étape permet de déclencher des actions qui ne sont pas déclenchées naturellement en HTML. C'es ttypiquement le cas pour des actions javascript.

Subtilités de sélection

1. En premier lieu, tous les éléments sauf **p** et **div** dont le texte contenu vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur l'"**id** de n'importe quel élément.
3. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** de n'importe quel élément.
4. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **value** de n'importe quel élément.
5. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le texte contenu de n'importe quel élément.

Sélection d'une option dans une liste de choix

Vocabulaire

```
je sélectionne la valeur "{valeur}" du champ "{champ}"
que je sélectionne la valeur "{valeur}" du champ "{champ}"
```

Exemples

```
Quand je sélectionne la valeur "Oui" du champ "Voulez-vous"
Et que je sélectionne la valeur "Bleu" du champ "Couleur"
```

Quand l'utiliser

Cette étape permet de sélectionner une valeur précise dans un champ de type sélection (select ou bouton radio) dans un formulaire.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un **select** dont l'"**id** vaut celui trouvé, puis une option dont le texte vaut *valeur*.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite en se basant sur l'"**id** des tags **select**, puis sur une option dont le texte vaut *valeur*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur l'"**id** des tags **select**, puis sur une option dont le **title** vaut *valeur*.
4. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite en se basant sur le **name** des tags **select**, puis sur une option dont le texte vaut *valeur*.

5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le **name** des tags **select**, puis sur une option dont le **title** vaut *valeur*.
6. Dans le cas où aucun **select** correspondant n'est trouvé, la recherche s'oriente alors vers les cases à cocher
7. Une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio*, qui doit être égal à *valeur*.
8. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio* dont l'"**id** vaut *champ*, qui doit être égal à *valeur*.
9. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio* dont le **name** vaut *champ*, qui doit être égal à *valeur*.
10. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *radio* dont l'"**id** vaut *champ* et dont le **value** doit être égal à *valeur*.
11. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *radio* dont le **name** vaut *champ* et dont le **value** doit être égal à *valeur*.

Saisie d'une valeur dans un champ de formulaire

Vocabulaire

```
je tape "{valeur}" dans le champ "{champ}"
que je tape "{valeur}" dans le champ "{champ}"
je tape "" dans le champ "{champ}"
que je tape "" dans le champ "{champ}"
je vide le champ "{champ}"
que je vide le champ "{champ}"
je tape ""
que je tape ""
je tape "{valeur}"
que je tape "{valeur}"
```

Exemples

```
Quand je tape "1, rue Jean Jaurès" dans le champ "Adresse"
Et que je tape "Paris" dans le champ "Ville"
Et que je tape "" dans le champ "Pays"
Et que je vide le champ "Téléphone"
Et que je tape "01" dans le champ "Début de validité"
Et que je tape "08"
Et que je tape "2010"
```

Quand l'utiliser

Ces étapes permettent de définir des valeurs pour des champs de formulaire. Chacune peut être utilisée suivant les besoins du test. Il faut noter que les étapes "je tape « {valeur} »" tape dans le navigateur sans pointer un champ spécifique.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un élément dont l'**id** vaut celui trouvé.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont l'**id** vaut *champ*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **name** vaut *champ*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **title** vaut *champ*.

Sélectionner un bouton radio

Vocabulaire

```
je clique sur le bouton radio "{valeur}"
que je clique sur le bouton radio "{valeur}"
```

Exemples

```
Quand je clique sur le bouton radio "Oui"
```

Quand l'utiliser

Ces étapes permettent de sélectionner la valeur d'un bouton radio (**input** de type *radio*).

Cocher ou décocher une case à cocher

Vocabulaire

```
je coche la case à cocher "{valeur}"
que je coche la case à cocher "{valeur}"
je décoche la case à cocher "{valeur}"
que je décoche la case à cocher "{valeur}"
```

Exemples

```
Quand je coche la case à cocher "Oui"
Et que je décoche la case à cocher "Recevoir des notifications"
```

Quand l'utiliser

Ces étapes permettent de cocher ou décocher des cases à cocher (**input** de type *checkbox*).

Subtilités de sélection

1. La recherche est faite en se basant sur le texte contenu des **input** de type *checkbox*, qui doit être égal à *valeur*.
2. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *checkbox* dont l'**id** vaut *champ*, qui doit être égal à *valeur*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *checkbox* dont le **name** vaut *champ*, qui doit être égal à *valeur*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *checkbox* dont l'**id** vaut *champ* et dont le **value** doit être égal à *valeur*.
5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *checkbox* dont le **name** vaut *champ* et dont le **value** doit être égal à *valeur*.

Sélectionner un fichier à transmettre

Vocabulaire

```
que je sélectionne le fichier "{fichier}" dans le champ "{champ}"
```

Exemples

```
Quand je sélectionne le fichier "pièce jointe.pdf" dans le champ "Ajouter une pièce␣
↪jointe"
```

Quand l'utiliser

Ces étapes permettent de fournir un fichier au formulaire afin de transmettre le fichier au serveur. Le fichier doit être situé dans le répertoire où se déroulent les tests. Il est généralement intéressant de laisser les fichiers liés à un test à côté du fichier *feature* qui lui correspond.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un élément dont l'**id** vaut celui trouvé.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont l'**id** vaut *champ*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **name** vaut *champ*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **title** vaut *champ*.

Attendre l'apparition d'un élément

Vocabulaire

```
j'attends un élément contenant "{valeur}"
que j'attends un élément contenant "{valeur}"
```

Exemples

```
Quand j'attends un élément contenant "Nouveaux échanges"
```

Quand l'utiliser

Cette étape est spécifique et concerne généralement des actions Ajax. En effet, il est nécessaire d'attendre que le navigateur ait reçu et pris en compte la réponse. Le test attend alors l'affichage d'un contenu spécifique. Cette étape est généralement utilisée avant de cliquer sur l'élement.

Subtilités de sélection

Le test va tenter de trouver l'élement toutes les 0,1 seconde.

1. En premier lieu, tous les éléments sauf **p** et **div** dont le texte contenu vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur l'**id** de n'importe quel élément.
3. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** de n'importe quel élément.
4. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **value** de n'importe quel élément.
5. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le texte contenu de n'importe quel élément.

Cliquer sur le ReCaptcha

Vocabulaire

```
je clique sur le ReCaptcha
que je clique sur le ReCaptcha
```

Exemples

```
Quand je clique sur le ReCaptcha
```

Quand l'utiliser

Cette étape est spécifique et concerne uniquement le ReCaptcha de test. Elle ne pourra absolument pas servir à passer un ReCaptcha configuré pour protéger un site mais bien un ReCaptcha configuré en mode TEST.

Les étapes de type "Alors"

Les étapes du type "Alors" sélectionnent des éléments invisibles par défaut.

Vérification du titre de la page

Vocabulaire

```
le titre de la page contient "{valeur}"
le titre de la page est "{valeur}"
```

Exemples

```
Alors le titre de la page contient "Titre"
Et le titre de la page est "Accueil"
```

Quand l'utiliser

Cette étape permet de vérifier un titre de page, et donc généralement que la page affichée est celle attendue.

Vérification du contenu de la page

Vocabulaire

```
la page contient "{valeur}"
la page contient "{valeur}" ou "{autre*valeur}"
la page ne contient pas "{valeur}"
```

Exemples

```
Alors la page contient "Accueil"
la page contient "résultats trouvés" ou "Aucun résultat"
la page ne contient pas "Ré-essayer"
```

Quand l'utiliser

Cette étape permet de vérifier la présence (ou l'absence) de texte dans la page.

Subtilités de sélection

Le texte testé est celui présent entre les tags (et non « dans » les tags), il est donc impossible de « vérifier » le contenu d'un bouton, même si celui-ci semble visible sur la page.

Afin de maximiser la reconnaissances, le texte de la page est complètement dépuillé de tous les tags, puis tous les sauts de ligne et autres multiples espaces sont supprimés.

Vérification du titre ou infobulle d'un élément

Vocabulaire

```
le titre de l'élément "{element}" est "{text}"
le titre de l'élément "{element}" contient "{text}"
le titre de l'élément qui contient "{element}" est "{text}"
le titre du champ "{field}" est "{text}"
```

Exemples

```
Alors le titre de l'élément "Entete" est "Accueil"
Et le titre de l'élément "Entete" contient "Acc"
Et le titre de l'élément qui contient "Ceci est un message" est "Message à caractère␣
↪informatif"
Et le titre du champ "Titre" est "Titre de la page"
```

Quand l'utiliser

Cette étape permet de vérifier la valeur *title* d'un infobulle.

Subtilités de sélection

S'agissant de l'élément :

1. En premier lieu, tous les éléments sauf **p** et **div** dont le texte contenu vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur l'"**id** de n'importe quel élément.
3. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** de n'importe quel élément.
4. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **value** de n'importe quel élément.
5. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le texte contenu de n'importe quel élément.

S'agissant du champ :

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un élément dont l'"**id** vaut celui trouvé.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont l'"**id** vaut *champ*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **name** vaut *champ*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **title** vaut *champ*.

Vérification de la présence d'un bouton

Vocabulaire

```
la page contient un bouton "{valeur}"
```

Exemples

```
Alors la page contient un bouton "Retour"
```

Quand l'utiliser

Cette étape permet de vérifier la présence d'un bouton dans la page.

Vérification de la présence d'un lien

Vocabulaire

```
la page contient un lien "{valeur}"
la page contient un lien "{valeur}" qui pointe vers "{target}"
```

Exemples

```
Alors la page contient un lien "Retour"
Et la page contient un lien "Retour" qui pointe vers "https://back.org"
```

Quand l'utiliser

Cette étape permet de vérifier la présence d'un lien dans la page.

Vérification de la présence d'une image

Vocabulaire

```
la page contient une image "{valeur}"
```

Exemples

```
Alors la page contient une image "logo"
Et la page contient une image "https://example.org/logo.png"
```

Quand l'utiliser

Cette étape permet de vérifier la présence d'une image dans la page.

Vérification de la valeur d'une variable

Vocabulaire

```
la variable "{variable}" vaut "{valeur}"
la variable "{variable_a}" est égale à la variable "{variable_b}"
la variable "{variable_a}" est différente de la variable "{variable_b}"
la variable "{variable_a}" est inférieure à la variable "{variable_b}"
la variable "{variable_a}" est supérieure à la variable "{variable_b}"
```

Exemples

```
Alors la variable "foo" vaut "bar"
```

Quand l'utiliser

Cette étape permet de vérifier la valeur d'une variable.

Vérification du titre d'un champ

Vocabulaire

```
le titre du champ "{champ}" est "{texte}"
```

Exemples

```
Alors le titr edu champ "foo" est "bar"
```

Quand l'utiliser

Cette étape permet de vérifier la valeur du titre d'un champ.

Vérification du contenu d'un élément

Vocabulaire

```
le champ "{champ}" est vide
le champ "{champ}" contient "{texte}"
le champ "{champ}" ne contient pas "{texte}"
le champ "{champ}" n'est pas vide
le champ "{champ}" contient "{nombre}" caractères
le champ "{champ}" contient moins de "{nombre}" caractères
le champ "{champ}" contient plus de "{nombre}" caractères
le champ "{champ}" contient entre "{nombre*min}" et "{nombre*max}" caractères
```

Exemples

```
Alors le champ "Adresse" est vide
Et le champ "Code postal" contient "49000"
Et le champ "Code postal" ne contient pas "51000"
Et le champ "Ville" n'est pas vide
Et le champ "SIRET" contient "12" caractères
Et le champ "Informations complémentaires" contient moins de "120" caractères
Et le champ "Password" contient plus de "12" caractères
Et le champ "Code postal" contient entre "5" et "10" caractères
```

Quand l'utiliser

Cette étape permet de vérifier le contenu de l'attribut *value* d'un champ de formulaire avec plusieurs possibilités.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un élément dont l'**id** vaut celui trouvé.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont l'**id** vaut *champ*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **name** vaut *champ*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **title** vaut *champ*.

Vérification des options d'une liste de choix

Vocabulaire

```
le champ "{champ}" contient l'option "{valeur}"
le champ "{champ}" contient l'option ""
le champ "{champ}" ne contient pas l'option "{valeur}"
le champ "{champ}" ne contient pas l'option ""
le champ "{champ}" contient "{valeur}" options
l'option "{valeur}" du champ "{champ}" est sélectionnée
l'option "" du champ "{champ}" est sélectionnée
```

Exemples

```
Alors le champ "Logement" contient l'option "T1"
Et l'option "F4" du champ "Logement" est sélectionnée
```

Quand l'utiliser

Cette étape permet de vérifier la liste des options d'une liste de choix. Elle permet également de vérifier quelle option est sélectionnée.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un **select** dont l'"**id** vaut celui trouvé, puis une option dont le texte vaut *valeur*.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite en se basant sur l'"**id** des tags **select**, puis sur une option dont le texte vaut *valeur*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur l'"**id** des tags **select**, puis sur une option dont le **title** vaut *valeur*.
4. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite en se basant sur le **name** des tags **select**, puis sur une option dont le texte vaut *valeur*.
5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le **name** des tags **select**, puis sur une option dont le **title** vaut *valeur*.
6. Dans le cas où aucun **select** correspondant n'est trouvé, la recherche s'oriente alors vers les cases à cocher
7. Une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio*, qui doit être égal à *valeur*.
8. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio* dont l'"**id** vaut *champ*, qui doit être égal à *valeur*.
9. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *radio* dont le **name** vaut *champ*, qui doit être égal à *valeur*.
10. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *radio* dont l'"**id** vaut *champ* et dont le **value** doit être égal à *valeur*.
11. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *radio* dont le **name** vaut *champ* et dont le **value** doit être égal à *valeur*.

Vérification de la sélection d'une case à cocher

Vocabulaire

```
la case à cocher "{champ}" est cochée
la case à cocher "{champ}" n'est pas cochée
la case à cocher "{champ}" est décochée
```

Exemples

```
Alors la case à cocher "Conservez mes informations" est cochée
Et la case à cocher "Se souvenir de moi" n'est pas cochée
```

Quand l'utiliser

Cette étape permet de vérifier si une case à cocher est cochée ou décochée.

Subtilités de sélection

1. La recherche est faite en se basant sur le texte contenu des **input** de type *checkbox*, qui doit être égal à *valeur*.
2. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *checkbox* dont l'**id** vaut *champ*, qui doit être égal à *valeur*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des **input** de type *checkbox* dont le **name** vaut *champ*, qui doit être égal à *valeur*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *checkbox* dont l'**id** vaut *champ* et dont le **value** doit être égal à *valeur*.
5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur les **input** de type *checkbox* dont le **name** vaut *champ* et dont le **value** doit être égal à *valeur*.

Cliquer sur un lien ou un bouton

Vocabulaire

```
je clique sur le bouton "{valeur}"
je clique sur le lien "{valeur}"
```

Exemples

```
Alors je clique sur le bouton "Valider"
Et je clique sur le lien "Retour"
```

Quand l'utiliser

Cette étape permet de vérifier l'existence d'un élément spécifique.

Vérification de l'existence d'un élément

Vocabulaire

```
un élément contenant "{valeur}" existe
```

Exemples

```
Alors un élément contenant "Demande validée" existe
```

Quand l'utiliser

Cette étape permet de vérifier l'existence d'un élément spécifique.

Vérification de l'existence d'un champ de formulaire

Vocabulaire

```
le champ "{champ}" existe
le champ "{champ}" n'existe pas
```

Exemples

```
Alors le champ "Deuxième enfant" existe
Et le champ "Quinzième enfant" n'existe pas
```

Quand l'utiliser

Cette étape permet de vérifier si un champ de formulaire est présent ou pas.

Subtilités de sélection

1. En premier lieu, la recherche se fait pour les tags **label** dont le contenu vaut *champ*.

   1.1. Si un label est trouvé, son attribut **for** est alors utilisé pour rechercher un élément dont l'**id** vaut celui trouvé.

2. Si aucun **label** correspondant n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont l'**id** vaut *champ*.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **name** vaut *champ*.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite sur tous les éléments dont le **title** vaut *champ*.

Vérification sur les tableaux

Vocabulaire

```
le tableau contient "{nombre}" colonnes
le tableau contient "{nombre}" lignes
le tableau contient moins de "{nombre}" lignes
la ligne "{nombre}" de la colonne "{colonne}" du tableau contient "{valeur}"
la ligne "{nombre}" de la colonne "{colonne}" du tableau contient ""
la ligne "{nombre}" de la colonne "{colonne}" du tableau est vide
la ligne "{nombre}" de la colonne "{colonne}" du tableau n'est pas vide
la colonne "{colonne}" du tableau est triée dans l'ordre croissant
la colonne "{colonne}" du tableau est triée dans l'ordre décroissant
la colonne "{colonne}" du tableau est triée dans l'ordre alphabétique
la colonne "{colonne}" du tableau est triée dans l'ordre alphabétique inversé
```

Exemples

```
Alors le tableau contient "6" colonnes
Et le tableau contient "10" lignes
Et le tableau contient moins de "21" lignes
Et la ligne "3" de la colonne "Code postal" du tableau contient "75000"
Et la ligne "3" de la colonne "Code postal" du tableau contient ""
Et la ligne "3" de la colonne "Code postal" du tableau n'est pas vide
Et la ligne "3" de la colonne "Code postal" du tableau est vide
Et la colonne "Commune" du tableau est triée dans l'ordre croissant
Et la colonne "Commune" du tableau est triée dans l'ordre alphabétique
Et la colonne "Commune" du tableau est triée dans l'ordre décroissant
Et la colonne "Commune" du tableau est triée dans l'ordre alphabétique inversé
```

Quand l'utiliser

Cette étape permet de vérifier plusieurs choses sur un tableau de données.

Subtilités de sélection

1. La recherche est faite en se basant sur le premier élément **table** de la page.
2. La recherche des colonnes se base d'abord sur les éléments **th** de l'élément **thead**.

   2.1. Si aucun élément **thead** n'est trouvé, alos les **th**, puis les premiers **td** sont utilisés.

3. La recherche des lignes se base sur les éléments **td** de chaque élément **tr**.

Vérification de présence d'un bouton

Vocabulaire

```
le bouton "{bouton}" existe
le bouton "{bouton}" n'existe pas
```

Exemples

```
Alors le bouton "Créer une demande" existe
Et le bouton "Supprimer une demande" n'existe pas
```

Quand l'utiliser

Cette étape permet de vérifier la présence ou l'absence d'un bouton sur la page.

Subtilités de sélection

1. En premier lieu, tous les éléments dont l'"**id** vaut *valeur* sont sélectionnés.
2. Si aucun élément n'est trouvé, une nouvelle recherche se fait en se basant sur le **name** des éléments.
3. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le **title** des éléments de type *button*, *input*, *img* ou *a* uniquement.
4. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur la **value** des éléments de type *button*, *input*, *img* ou *a* uniquement.
5. Si aucun élément n'est trouvé, une nouvelle recherche est faite en se basant sur le texte contenu des éléments de type *button*, *input*, *img* ou *a* uniquement.

Etapes utilitaires

Ces étapes sont utilisables à tout moment des tests.

Attente

Vocabulaire

```
j'attends {secondes} secondes
que j'attends {secondes} secondes
j'attends {seconde} seconde
que j'attends {seconde} seconde
```

Exemples

```
Etant donné que j'attends 5 secondes
Quand j'attends 3 secondes
Alors j'attends 1 secondes
```

Quand l'utiliser

Cette étape permet d'attendre un certain temps, par exemple lorsqu'on fait les tests manuellement et qu'on a besoin de vérifier que les étapes précédentes se sont bien déroulées. Il n'est, en général, pas nécessaire de conserver ces étapes pour les tests finaux.

Faire une capture d'écran

Vocabulaire

```
je fais un screenshot
que je fais un screenshot
je fais un screenshot sous le nom "{nom_fichier}"
que je fais un screenshot sous le nom "{nom_fichier}"
je fais une capture d'écran
que je fais une capture d'écran
je fais une capture d'écran sous le nom "{nom_fichier}"
que je fais une capture d'écran sous le nom "{nom_fichier}"
```

Exemples

```
Etant donné que je fais une capture d'écran
Quand je fais un screenshot sous le nom "page d'accueil"
Alors je fais une capture d'écran sous le nom "résultat_$identifiant$"
```

Quand l'utiliser

Cette étape permet de faire une capture de l'écran actuel et de sauver celle-ci dans le répertoire **screenshots** prévu à cet effet. Par défaut, chaque capture d'écran est sauvée avec la date et l'heure de la capture en tant que nom de fichier.

Comme indiqué dans le dernier exemple, il est tout à fait possible d'utiliser une *variable* dans le nom du fichier.

Assigner une valeur à une variable

Vocabulaire

```
j'assigne la valeur "{valeur}" à la variable "{variable}"
que j'assigne la valeur "{valeur}" à la variable "{variable}"
j'assigne la valeur du champ "{champ}" à la variable "{variable}"
que j'assigne la valeur du champ "{champ}" à la variable "{variable}"
```

Exemples

```
Etant donné que j'assigne la valeur "toto" à la variable "prénom"
Quand j'assigne la valeur "$mail$" à la variable "identifiant"
Alors j'assigne la valeur du champ "Nom" à la variable "nom"
```

Quand l'utiliser

Cette étape permet d'assigner une valeur à une variable. On notera qu'il ne faut pas entourer le nom de la variable ciblée de caractères « $ » sous peine de la voir remplacer par sa valeur actuelle, ce qui n'est peut-être pas le but recherché, mais cela reste possible. Voir *Les variables* pour plus d'infomations.

Exécuter une requête

Vocabulaire

```

```

j'exécute la requête « {requête} » que j'exécute la requête « {requête} » j'exécute la requête « {requête} » avec le paramètre « {paramètre} » que j'exécute la requête « {requête} » avec le paramètre « {paramètre} » j'exécute la requête « {requête} » avec les paramètres « {paramètres} » que j'exécute la requête « {requête} » avec les paramètres « {paramètres} »

Exemples

```
Etant donné que j'exécute la requête "première_requête"
Quand j'exécute la requête "seconde_requête" avec le paramètre "$param$"
Alors j'exécute la requête "dernière_requête" avec les paramètres "toto,titi"
```

Quand l'utiliser

Cette étape permet d'exécuter une requête SQL à travers un connecteur spécifique. Si plusieurs paramètre sont fournis, ils sont séparés par des virgules.

Comme indiqué dans le deuxième exemple, il est tout à fait possible d'utiliser une *variable* dans le nom de la requête ou des paramètres.

Assigner le résultat d'une requête à une variable

Vocabulaire

```
j'assigne le résultat de la requête "{requête}" à la variable "{variable}"
que j'assigne le résultat de la requête "{requête}" à la variable "{variable}"
j'assigne le résultat de la requête "{requête}" avec le paramètre "{paramètre}" à la␣
↪variable "{variable}"
que j'assigne le résultat de la requête "{requête}" avec le paramètre "{paramètre}" à la␣
↪variable "{variable}"
j'assigne le résultat de la requête "{requête}" avec les paramètres "{paramètres}" à la␣
↪variable "{variable}"
que j'assigne le résultat de la requête "{requête}" avec les paramètres "{paramètres}" à␣
↪la variable "{variable}"
```

Exemples

```
Etant donné que j'assigne le résultat de la requête "première_requête" à la variable "un"
Quand j'assigne le résultat de la requête "seconde_requête" avec le paramètre "$param$"␣
↪à la variable "deux"
Alors j'assigne le résultat de la requête "dernière_requête" avec les paramètres "toto,
↪titi" à la variable "trois"
```

Quand l'utiliser

Cette étape est identique à la précédente. Elle ajoute en plus la possibilité de récupérer la première valeur de la première colonne pour l'assigner à une variable.

Vérifier qu'une requête renvoie une valeur spécifique

Vocabulaire

```
le résultat de la requête "{requête}" est vide
que le résultat de la requête "{requête}" est vide
le résultat de la requête "{requête}" avec le paramètre "{paramètre}" est vide
que le résultat de la requête "{requête}" avec le paramètre "{paramètre}" est vide
le résultat de la requête "{requête}" avec les paramètres "{paramètres}" est vide
que le résultat de la requête "{requête}" avec les paramètres "{paramètres}" est vide
le résultat de la requête "{requête}" contient "{texte}"
```

(suite sur la page suivante)

(suite de la page précédente)

```
que le résultat de la requête "{requête}" contient "{texte}"
le résultat de la requête "{requête}" avec le paramètre "{paramètre}" contient "{texte}"
que le résultat de la requête "{requête}" avec le paramètre "{paramètre}" contient "
↪{texte}"
le résultat de la requête "{requête}" avec les paramètres "{paramètres}" contient "
↪{texte}"
que le résultat de la requête "{requête}" avec les paramètres "{paramètres}" contient "
↪{texte}"
```

Exemples

```
Alors le résultat de la requête "cherche_utilisateur" est vide
Alors le résultat de la requête "cherche_utilisateur" contient "$identifiant$"
```

Quand l'utiliser

Cette étape permet de valider la valeur de la première colonne du premier enregistrement retourné par la requête.

Vérifier qu'une requête renvoie quelque chose

Vocabulaire

```
le résultat de la requête "{requête}" n'est pas vide
que le résultat de la requête "{requête}" n'est pas vide
le résultat de la requête "{requête}" avec le paramètre "{paramètre}" n'est pas vide
que le résultat de la requête "{requête}" avec le paramètre "{paramètre}" n'est pas vide
le résultat de la requête "{requête}" avec les paramètres "{paramètres}" n'est pas vide
que le résultat de la requête "{requête}" avec les paramètres "{paramètres}" n'est pas␣
↪vide
```

Exemples

```
Alors le résultat de la requête "dernière_requête" n'est pas vide
```

Quand l'utiliser

Cette étape permet de valider que la première colonne du premier enregistrement retourné par la requête n'est pas vide.

Les variables

Afin de permettre la dynamisation des tests, il est possible d'utiliser des variables dans les valeurs données aux étapes.

Une variable est définie par une chaine de caractères au format "$…..$", où la partie située entre les double dollars correspond au nom de la variable.

Les fonctions

Afin de permettre la dynamisation des tests, il est possible d'utiliser des fonctions dans les valeurs données aux étapes.

Une fonction est définie par une chaine de caractères au format "##……##", où la partie située entre les double dièses correspond au comportement attendu.

Le comportement correspond à un champ *quoi* et un champ *comment*. Ces deux champs sont séparés par un caractère « _ ». Alors que le champ *quoi* désigne ce qui est attendu, le champ *comment* désigne le format attendu.

Les *quoi* possibles sont :
- JOUR

JOUR

Cette fonction correspond à une date. Par défaut, la date du jour est sélectionnée. Il est cependant possible d'augmenter ou diminuer le jour en ajoutant une opération.

La fonction supporte les formats suivants :
- JOUR
- MOIS
- ANNEE

Exemples :

```
##JOUR_JOUR##
```

Cette fonction renvoit le jour du jour actuel.

```
##JOUR-1_MOIS##
```

Cette fonction renvoit le mois de la veille du jour actuel.

JOUR

Renvoit le jour sur 2 chiffres.

MOIS

Renvoit le mois sur 2 chiffres.

ANNEE

Renvoit l'année sur 4 chiffres.

3.7.5 Configurer K-lity

Le fichier *configuration.yml* permet d'ajuster le comportement de K-lity au plus juste.

behave

Cette option permet de configurer *behave* lui-même.

```
behave:
    lang: fr
```

Valeurs par défaut

La langue est positionnée à « fr ».

databases

Cette option permet de configurer une à plusieurs connexions à des bases de données. Il s'agit d'un dictionnaire d'informations de connexions dont la clef représente le nom de la connexion à utiliser dans les requêtes SQL.

Actuellement, seules les connexions *postgresql* sont gérées.

```
databases:
    db_1:
        type: postgresql
        host: localhost
        database: my_db
        user: user
        password: password
    db_2:
        type: postgresql
        host: some_server
        database: other_db
        user: user
        password: complex_password
```

Valeurs par défaut

La liste est vide.

variables

Cette option permet de spécifier des variables globales à tous les tests. Les variables sont ré-initialisées à chaque test.

```
variables:
    foo: bar
```

Valeurs par défaut

La liste est vide.

reporting

Cette option permet de configurer les générations de rapport.

```
reporting:
    title: Some title
    template: some_template
```

Valeurs par défaut

title : Default title

template : default

options

Cette option permet de configurer le comportement de K-lity.

```
options:
    timeout: 15
    headless: True
```

Valeurs par défaut

timeout : 10

headless : False

3.7.6 Utiliser ses propres templates

Si vous souhaitez utiliser vos propres templates pour la génération du rapport de test, il vous faut créer la structure adéquate :

```
your_project
├─ steps
└─ templates
   └─ template_name
      ├─ index.html
      └─ static
```

Le répertoire *templates* contient tous vos templates personnels. Chacun de ces templates est représenté par un répertoire distinct. Chaque template contient au moins un fichier *index.html* qui sera utilisé pour générer le rapport.

Le répertoire *static* est optionnel et permet de stocker des médias qui seront tous copiés sous la même structure lors de la génération du rapport.

Modèle de template

Les rapports générés par K-lity s'appuient sur le système de templating Jinja2.

Données fournies au template

Les données fournies au template le sont sous la forme d'un dictionnaire aisément utilisable :

```
data = {
    "title": string,                    -- Titre du rapport
    "total": integer,                   -- Nombre total de fonctionnalités
    "danger": integer,                  -- Nombre de fonctionnalités ayant plus de 40%␣
↪de scénarios échoués
    "warning": integer,                 -- Nombre de fonctionnalités ayant au moins un␣
↪scénario échoué
    "success": integer,                 -- Nombre de fonctionnalités sans scénario échoué
    "passed": integer,                  -- Indique le nombre de fonctionnalités passées
    "failed": integer,                  -- Indique le nombre de fonctionnalités échouées
    "time": integer,                    -- Indique le temps écoulé en secondes
    "time_elapsed": string,             -- Indique le temps écoulé au format HH:MM:SS
    "features": [                       -- Liste des fonctionnalités
        "name": string,                 -- Nom de la fonctionnalité
        "scenarios": [                  -- Liste des scénarios de la fonctionnalité
            {
                "name": string,         -- Nom du scénario
                "passed": boolean,      -- Flag indiquant si le scénario est passé ou␣
↪échoué
                "logs": [],             -- Liste des lignes de logs du scénario
            },
        ],
        "passed": integer,              -- Indique le nombre de scénarios passés
        "failed": integer,              -- Indique le nombre de scénarios échoues
        "state": string,                -- Indique l'état de la fonctionnalité (danger,␣
↪warning, success)
        "total": integer,               -- Nombre total de scénarios
        "time": integer,                -- Indique le temps écoulé en secondes
        "time_elapsed": string,         -- Indique le temps écoulé au format HH:MM:SS
    ],
    "generation": time,                 -- Date/heure de génération du rapport
}
```

3.7.7 À propos

K-lity est une application de test automatisés créée au sein de Klee Group dans le but d'améliorer la qualité des applications produites.

Licence

K-lity est publié sous la licence AGPL V3+.