cockpit
IT Service Manager

Supervision - Configuration du contrôle "Web - Scénario"

Document FAQ

§Table des matières

Introduction

I. Objectif

- Présenter le fonctionnement du contrôle de supervision « Web - Scénario ».

II. Prérequis

Le contrôle « Web - Scénario » utilise ChromeDriver et le navigateur Google Chrome.
ChromeDriver est un exécutable utilisé pour contrôler Chrome.
Les prérequis suivants concernent les moteurs de supervision et les différents cas (moteur de supervision sous Windows ou Linux).

A. Windows

- Google Chrome doit être installé dans le répertoire par défaut proposé à l'installation.
- L'exécutable « chromedriver.exe » doit être dans le répertoire « C:\koaly\exp\lib\windows\ ».

B. Linux

1. Debian / Ubuntu

Installer le package « chromium-chromedriver » via la commande :

sudo apt-get install chromium-chromedriver

2. RedHat / CentOS

Créer un fichier repository dans le répertoire suivant :

vi /etc/yum.repos.d/google-chrome.repo

Renseigner le fichier avec les données suivantes :

```
name=google-chrome
baseurl=http://dl.google.com/linux/chrome/rpm/stable/x86_64
enabled=1
gpgcheck=1
gpgkey=https://dl-ssl.google.com/linux/linux_signing_key.pub
```

Installer Google Chrome via la commande :

yum install -y google-chrome-stable

III. Fonctionnement

- Les scénarios web se présentent sous forme de scripts de type JavaScript.
- Les scripts utilisent des commandes telles que « browser.click(objet) » ou « assert.titleEquals(NomFenêtre) » pour exécuter les actions à effectuer au cours du scénario.
- Les scripts sont exécutés par ChromeDriver.

Paramétrage du contrôle

I. Paramètres génériques

- Disponibilité équipement : Non
- Statut inversé : Non
- Graphique : Oui
- Double seuil : Non
- Disponible dans les rapports : Oui (Disponibilité / Supervision)

II. Paramètres spécifiques

Paramètre	Information	Valeur	Obligat.
Script	Mettre le script du scénario à exécuter. Voir les commandes et exemples de script plus loin dans le document.		Oui
Chiffrement	Dans le champ « Script » sélectionner à la souris le texte à chiffrer (utilisateur ou mot de passe d'authentification) puis cliquer sur le bouton « Chiffrement », le texte sélectionné est alors crypté. Dans le script il faudra utiliser la fonction « crypto.decrypt » (voir la partie « Crypto » plus bas). L'objectif est de masquer des informations sensibles comme l'utilisateur ou le mot de passe qui pourraient être vues par des opérateurs accédant à la gestion du contrôle.		Non
Seuil d'alerte	Alerte si le scénario n'aboutit pas.		
	Alerte si le temps de réponse est supérieur à Xms.	Nombre entier entre 500 et 60 000	Non
	Conserver les résultats (métriques)		Non

III. Alertes de configuration

- L'exécutable « chromedriver.exe » n'est pas dans le répertoire des librairies.
- Le script contient un élément qui ne peut être interprété.

Conception du scénario

I. Commandes

Le scénario est rédigé sous la forme d'un script JavaScript.

Le moteur met à disposition des objets et fonctions permettant d'effectuer des requêtes et d'interagir avec les pages dans le navigateur Web.

Ci-dessous la liste des objets et fonctions pouvant être utilisés.

A. Locator

Les « locator » permettent de rechercher et d'identifier un élément d'une page Web :

Locator	Description	Exemple
id	Élément identifié par l'attribut « id »	id=userInput
name	Élément identifié par l'attribut « name »	name=username
tag	Éléments identifés par leur nom	tag=div
linkText	Liens (<a>) identifiés par le texte affiché	linkText=Click here
xpath	Expression XPath	xpath=//button[@class='login-button z-button']
className	Élément identifié par l'attribut « class » Si l'attribut contient plusieurs classes, la recherche se fait sur chacune de ces classes	className=loginButton
css	Sélecteur CSS	Css=.class=toto

> Note : Un locator peut identifier plusieurs éléments. Dans ce cas, le premier élément trouvé sur la page Web est pris en compte.

B. Browser

Permet d'interagir avec le navigateur et avec la page affichée :

Fonction	Paramètres	Description
browser. setDefaultWait	Timeout : integer	Le délai d'attente par défaut (en millisecondes) qui sera utilisé dans les commandes « browser ». Le timeout par défaut est de **5 secondes**.
browser.open	url : string	Ouvre une Url Exemple : browser.open(« http://www.site.com »)
browser.waitForElement	locator : string errorMessage : string	Attend la présence d'un élément avec le timeout par défaut et affiche le message d'erreur si l'élément n'est pas présent après le timeout. Exemple :

		browser.waitForElement(id=« login-button », « Connexion »)
browser.waitForElement	locator : string timeout : integer errorMessage : string	Attend la présence d'un élément avec le timeout spécifié (en millisecondes) et affiche le message d'erreur si l'élément n'est pas présent après le timeout.
browser.click	locator : string	Effectue un clic sur un élément.
browser.type	locator : string value : string	Renseigne une valeur dans un champs.
browser.submit	locator : string	Soumet un formulaire (possible pour des éléments <form> uniquement).
browser.resizeWindow	width : integer height : integer	Changer la taille de la fenêtre du navigateur.

C. Assert

Permet d'effectuer des vérifications sur des éléments de la page actuelle sans interagir avec elle.

Fonction	Paramètres	Description
assert.titleEquals	Expected : string	S'assurer que le titre de la page correspond à la valeur attendue.
assert.textEquals	locator : string expected : string	S'assurer que le texte d'un élément correspond à la valeur attendue.
Assert.isDisplayed	failMessage : string locator : string	S'assurer qu'un élément est visible
Assert.isNotDisplayed	failMessage : string locator : string	S'assurer qu'un élément n'est pas visible
Assert.isEnabled	failMessage : string locator : string	S'assurer qu'un élément est actif
Assert.isDisabled	failMessage : string locator : string	S'assurer qu'un élément n'est pas actif

D. Crypto

Permet de déchiffrer des données chiffrées à l'aide de l'outil de chiffrement disponible au niveau de la configuration du contrôle.

Fonction	Paramètres	Description
crypto.decrypt	data : string	Retourne les données déchiffrées

Exemple insertion d'un texte non encrypté :

```
browser.type(name=username, "user1");
```

Exemple insertion d'un texte encrypté via l'outil de cryptage du contrôle :

```
browser.type(name=username, crypto.decrypt("password"));
```

E. Environnement

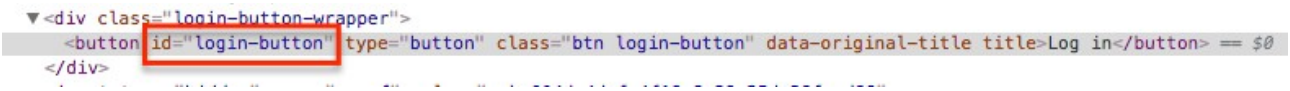
Permet d'accéder à des variables d'environnement. Par exemple le nom DNS de l'équipement sur lequel le contrôle est déployé. L'objectif est de ré-utiliser des scénarios sur des URLs ayant peu de différence.

Nom	Type	Description
env.dnsName	string	Nom DNS de l'équipement associé au contrôle

II. Méthode

A. Recherche des éléments

La recherche d'éléments sur la page Web à contrôler est nécessaire, pour cela utiliser un navigateur type Firefox ou Google Chrome.
Afficher la page Web à contrôler et effectuer un clic droit sur le champ ou bouton sur lequel vous souhaitez interagir puis sélectionner « Inspecter » ou « Inspecter l'élément » (l'appellation change selon le navigateur). Ses propriétés sont alors affichées.
Exemple :



B. Élaboration du scénario

Avant de commencer à rédiger le code d'un scénario en JavaScript, il est conseiller d'élaborer le scénario afin de visualiser les étapes.
Exemple :

- Page de connexion
- Connexion
- Attendre l'apparition du menu « xxx »
- Déconnexion
- Attendre le bouton de connexion

> Note : Remarquez que pour un scénario incluant une connexion / déconnexion, il est intéressant de contrôler le processus de déconnexion jusqu'au bout.

C. Rédaction du code

1. Variables

Objectif : Déclarer les variables, cela permet d'identifier tous les éléments du site qui seront manipulés dans le scénario (bouton, menu, etc.).
Dans les exemples suivants nous recherchons des éléments à déclarer dans des variables, nous utilisons souvent « xpath » qui se construit de la manière suivante :

```
xpath=//type[@Element='Value']
```

Où le type dépend de ce que l'on cherche sur la page (button, div, img, etc.).

Exemples :

- Des champs où vous renseignez des valeurs (utilisateur, terme recherché, mot de passe, etc.) :

```
var usernameInput = "name=username";
```

```
<input id="username-input" type="text" value class="username-input form-control" name="username" required="required" autofocus="autofocus" aria-required="true">
```

- Des boutons qu'il faut actionner :

```
var loginButton = "xpath=//button[@class='btn login-button']";
```

```
<div class="login-button-wrapper">
  <button id="login-button" type="button" class="btn login-button" data-original-title title>Conn
</div>
```

```
var logoutConfirmationButton = "xpath=//button[text()='Oui']";
```

```
<td id="mKCVn1-chdex" style="height:100%">
  <button type="button" id="mKCVn1" class="z-messagebox-button z-button">Oui</button> == $0
</td>
```

```
var userMenuButton = "xpath=//img[@src='/images/specific/blueClear/list-action-execute.png']";
```

```
<td id="mKCVl-chdex" style="height:100%">
  <img id="mKCVl" class="z-image" src="/images/specific/blueClear/list-action-execute.png">
</td>
```

- Des menus auxquels on voudra accéder, dans l'exemple suivant on retrouve le menu via la class et le texte du menu :

```
var ticketMenu = "xpath=//div[@class='koalyexpMainMenuItem-text' and text()='Tickets']";
```

```
<div id="mKCVo" class="koalyexpMainMenuItem">
  <div id="mKCVo-label" class="koalyexpMainMenuItem-text">Tickets</div>
</div>
```

> Note importante : On peut voir que les éléments peuvent être recherchés de différentes manières (par la classe, le texte, etc.).
> La méthode « xpath » permet différentes sortes d'éléments (button, div, img, etc.).

2. Paramètres généraux

Indiquer le timeout en millisecondes au-delà duquel le script considère que la page ne répond plus.

```
// set default wait (in ms)
browser.setDefaultWait(10000);
```

3. Actions

Objectifs : Présenter les principales actions utilisées dans les scénarios.

Dans les exemples suivants nous utilisons les variables déclarées précédemment.

Les quelques commandes présentées permettent de naviguer dans un site Web.

- Afficher un site, on attend l'élément « loginButton » dans la page, qui est une variable que l'on a déclaré précédemment ainsi que le nom de la page « Cockpit ITSM » :

```
browser.open("http://support.cockpit-itsm.com");
browser.waitForElement(loginButton, "Login button not found on home page");
assert.titleEquals("ITSM Cockpit");
```

- Renseigner un champ par la valeur « name » :

```
browser.type(usernameInput, "name");
```

- Cliquer sur un bouton :

```
browser.click(loginButton);
```

- Attendre qu'un menu s'affiche et cliquer dessus :

```
browser.waitForElement(ticketMenu, "Ticket menu not found");
browser.click(ticketMenu);
```

> Note : Ajouter un message comme « Ticket menu not found » est important, il est repris dans le message d'alerte, cela aide à la résolution du problème.

Exemples

I. Site Web Cockpit ITSM

```
//
// Cockpit ITSM - Web scenario //
//
// On déclare en variable tout ce qui sera manipulé (boutons, champs, menus, etc.)
var loginButton = "xpath=//button[@class='login-button z-button']";
var usernameInput = "name=username";
var passwordInput = "name=password";

var ticketMenu = "xpath=//div[@class='koalyexpMainMenuItem-text' and text()='Tickets']";
var ticketListTitle = "xpath=//div[text()='Tickets affectés à mes équipes']";

var taskMenu = "xpath=//div[@class='koalyexpMainMenuItem-text' and text()='Tâches']";
var taskListTitle = "xpath=//span[text()=\"File d'attente des tâches à acquitter\"]";

var userMenuButton = "xpath=//img[@src='/images/specific/blueClear/list-action-execute.png']";
var logoutButton = "xpath=//img[@src='/images/specific/blueClear/menu-top-button-exit.png']";
var logoutConfirmationMessage = "xpath=//div[text()='Quitter le portail']";
var logoutConfirmationButton = "xpath=//button[text()='Oui']";


// Paramétrage du timeout par défaut (in ms)
browser.setDefaultWait(10000);


// Affichage du site à contrôler
browser.open("http://YourPortalCockpit.com");
browser.waitForElement(loginButton, "Login button not found on home page");
assert.titleEquals("Cockpit ITSM");

// Authentification
browser.type(usernameInput, "name");
browser.type(passwordInput, "password");
browser.click(loginButton);


// Dans cette boucle nous changeons de menu 10 fois
// Le but est de générer un peu de charge
for (i = 0; i < 10; i++) {
       // Show tickets
       browser.waitForElement(ticketMenu, "Ticket menu not found");
       browser.click(ticketMenu);
       browser.waitForElement(ticketListTitle, "Ticket list not found");

       // Show tasks
       browser.waitForElement(taskMenu, "Task menu not found");
       browser.click(taskMenu);
       browser.waitForElement(taskListTitle, "Task list not found"); }
}


// Ouvrir le menu contenant l'action de déconnexion
browser.waitForElement(userMenuButton, "User menu not found");
browser.click(userMenuButton);

// Cliquer sur le bouton de déconnexion
browser.waitForElement(logoutButton, "Logout button not found");
browser.click(logoutButton);

// Confirmation de la déconnexion
browser.waitForElement(logoutConfirmationMessage, "Logout confirmation message not found");
browser.waitForElement(logoutConfirmationButton, "Logout confirmation button not found");
browser.click(logoutConfirmationButton);

// Retour à la page d'accueil
browser.waitForElement(loginButton, "Login button not found after logout");
assert.titleEquals("Cockpit ITSM");
```

> Note : Lors d'un scénario, ne pas oublier de vérifier le contenu de chaque page, notamment la page de déconnexion comme dans l'exemple ci-dessus.

Fin du document