Tests fonctionnels avec Selenium WebDriver

Livrables à déposer sur Ametice (voir la dernière page de ce document)

Outils et technologies utilisées

IDE : IntelliJ IDEA

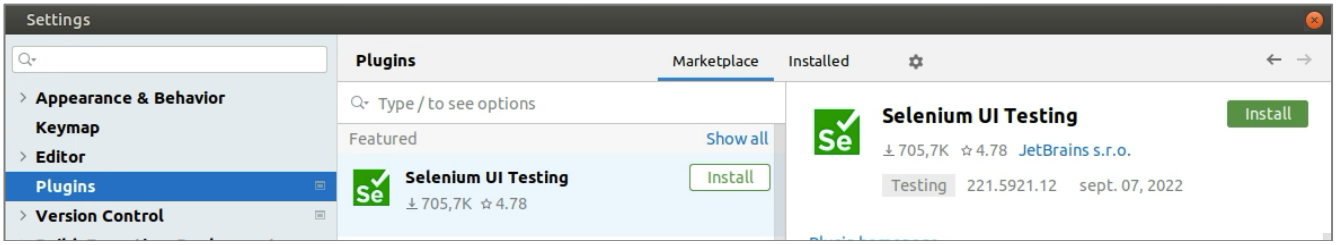
Tests fonctionnels : Selenium WebDriver API

WebDriver est une Recommandation du W3C de juin 2018, qui permet de contrôler le comportement d'un navigateur local ou distant.

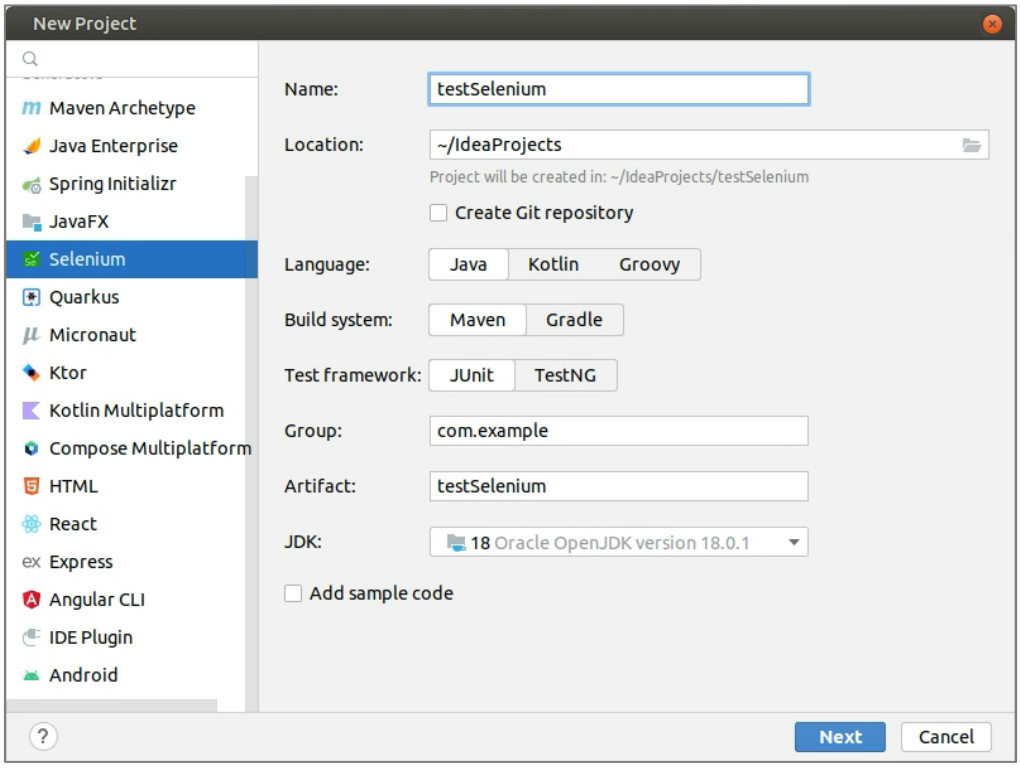
Selenium Web Driver API permet de piloter les actions dans un navigateur à partir d'un script WebDriver. Il supporte les navigateurs Firefox, Google Chrome/Chromium, Opera, Safari, Internet Explorer ≥ 7, Edge. Des pilotes sont utilisés pour communiquer avec les navigateurs. Le script peut être écrit dans les langages C#, Java, JavaScript, Perl, PHP, Python, R, Ruby.

1. Installation

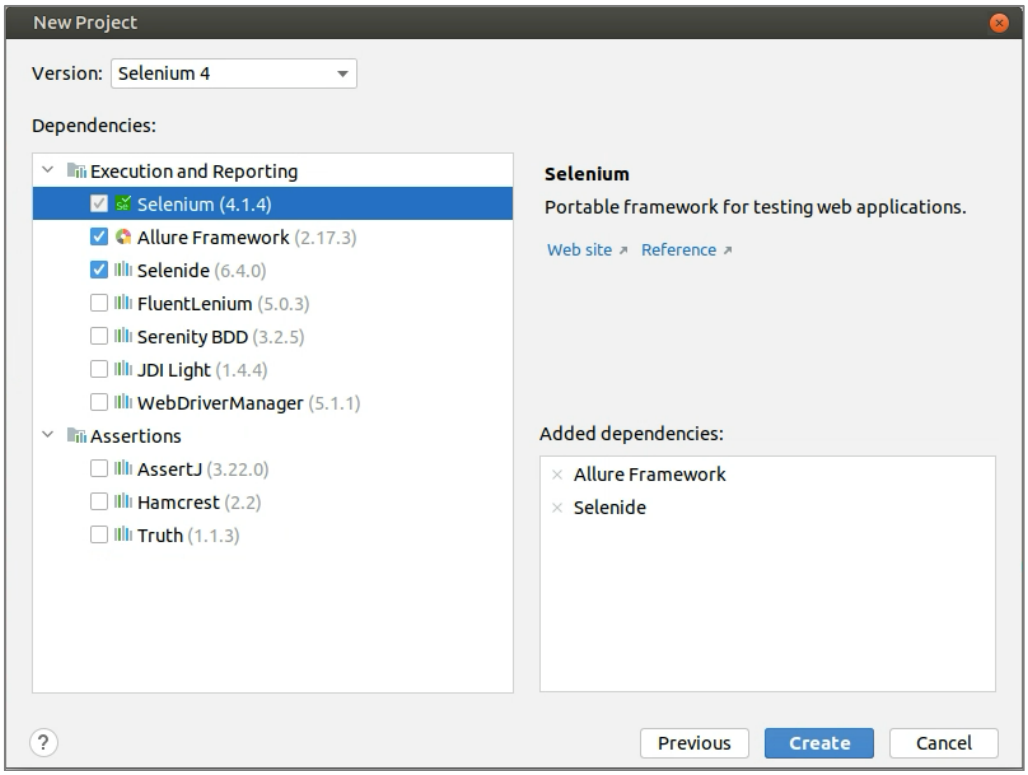
- Télécharger le plugin Selenium pour IntelliJ
  Aller dans les préférences IntelliJ, cliquer sur « Plugins », ajouter Selenium UI Testing. Cliquer sur « Apply » puis sur « OK ».



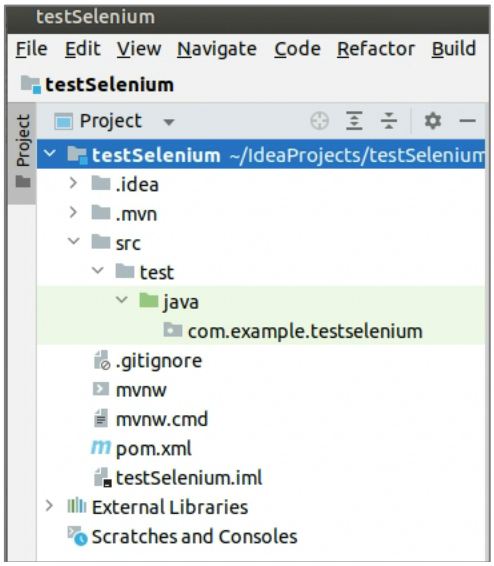
- Créer un projet Selenium nommé *testselenium*, choisir le langage Java, le framework de test JUnit, décocher « Add sample code », puis cliquer sur « Next ».



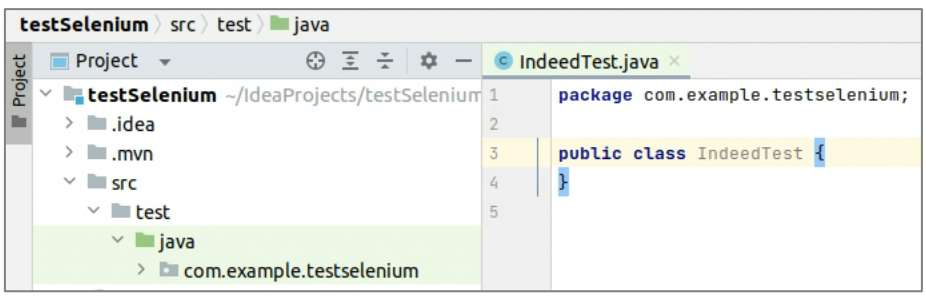
Sélectionner les dépendances Selenium, Allure Framework et Selenide, puis cliquer sur « Create »



Le projet créé contient un répertoire java dans le répertoire test, les classes de test seront placées dans ce répertoire.



Créer une classe IndeedTest :



Nous allons utiliser le pilote Chrome :

- Ouvrir un terminal dans le répertoire *test-selenium-ide*
- Installer le pilote chrome qui correspond à la version de Chromium. La correspondance entre le pilote et la version est indiquée sur la page de téléchargement de ChromeDriver :
  https://chromedriver.chromium.org/downloads

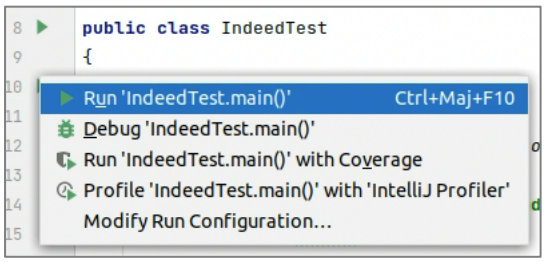
Par exemple, si Chromium version 101.x est installé sur VDI, entrer la commande suivante pour installer le pilote correspondant :

```
npm install chromedriver@101
```

2. Créer un premier test fonctionnel WebDriver

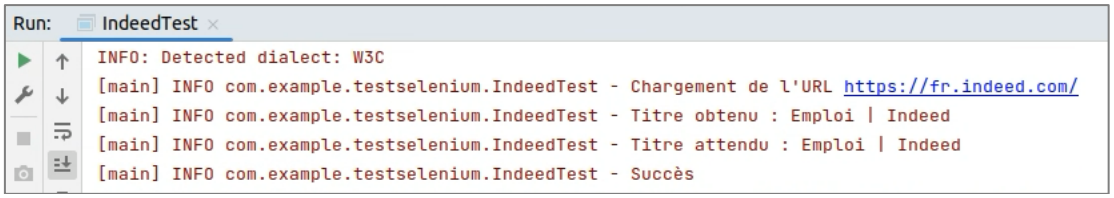
Nous allons créer un test fonctionnel dans la classe IndeedTest, qui vérifie dans Chrome que la page de Indeed contient bien le texte « Emploi | Indeed » dans l'élément HTML *title*.

```
package com.example.testselenium;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class IndeedTest
{
  public static void main(String[] args) {

    Logger logger = LoggerFactory.getLogger(IndeedTest.class);
    WebDriver driver;
    String titreAttendu = "Emploi | Indeed";
    String titreObtenu = "";

    // Créer le pilote pour Chrome
    // remplacer CHEMIN/ par le chemin d'accès à geckodriver sur votre disque
    System.setProperty( "webdriver.chrome.driver", "CHEMIN/chromedriver");
    driver = new ChromeDriver();

    // Lancer le navigateur, charger l'URL
    logger.info("Chargement de l'URL https://fr.indeed.com/");
    driver.get("https://fr.indeed.com/");

    // Récupérer le titre de la page web
    titreObtenu = driver.getTitle();
    logger.info("Titre obtenu : " + titreObtenu);
    logger.info("Titre attendu : " + titreAttendu);

    // Vérifier que le titre de la page est celui obtenu
    if (titreObtenu.contentEquals(titreAttendu)){
      logger.info("Succès");
    } else {
      logger.info("Echec");
    }
    // Quitter le navigateur
    driver.close();
  }
}
```



- Exécuter le test

- Le navigateur est ouvert, l'URL est chargée, le résultat du test est affiché dans la console.



- Plutôt que d'utiliser `System.setProperty`, vous pouvez ajouter le chemin du répertoire contenant chromedriver dans la variable d'environnement PATH du système.

3. Localiser des objets dans une page

Utiliser la documentation de Selenium WebDriver Java pour trouver les méthodes permettant de sélectionner les éléments dans la page, afin de réaliser les vérifications ci-après.

https://www.selenium.dev/documentation/webdriver/elements/finders/

- Modifier la classe pour qu'elle vérifie que la page web contient un formulaire dont l'**id** est *jobsearch*.
- Vérifier que l'**élément** h2 contient la chaîne « Recherches populaires ».
- Vérifier que la page contient un lien hypertexte dont le texte est « À propos ».
- Vérifier que la page contient un **lien hypertexte** dont une partie du texte est « Guide ».
- Vérifier que la page contient un champ de formulaire dont le **nom** est q.

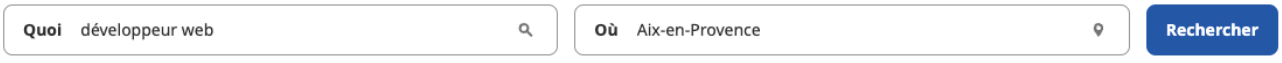
4. Exécuter des actions dans la page

- Entrer la chaîne de recherche « Développeur web » dans l'input dont le **nom** est q avec la méthode *sendKeys*.

- Faire une **tabulation** sur l'élément dont le nom est q. Le contenu du champ de formulaire « où » devrait apparaître sélectionné. Utiliser *Keys*.

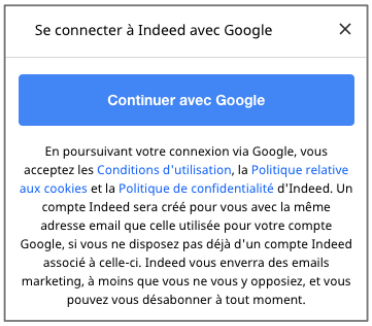
https://www.selenium.dev/selenium/docs/api/py/webdriver/selenium.webdriver.common.keys.html

- Entrer la chaîne de recherche « Aix-en-Provence » dans l'input dont le **nom** est l, avec la méthode *sendKeys*.



- Utiliser un sélecteur CSS (By.cssSelector) pour **sélectionner** et **envoyer le formulaire** avec la méthode *submit*. Vous devriez obtenir la page de résultat contenant les offres pour les développeurs web à Aix-en-Provence.
- Faire une **pause** de 5 secondes

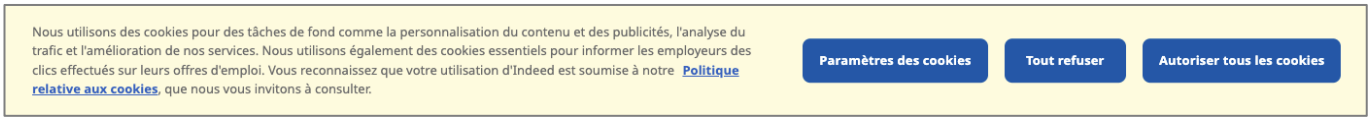
- Si la page de résultat affiche une fenêtre popup de connexion, fermer la fenêtre en cliquant sur « X ».



  NB : si le popup est présent dans la page, mais qu'il n'apparaît pas rapidement, une exception est obtenue car l'élément à fermer n'est pas trouvé dans la page. Dans ce cas, vous pouvez définir un temps d'attente « Implicit wait » :

  https://www.selenium.dev/documentation/webdriver/waits/#implicit-wait

- Cliquer sur le bouton pour accepter tous les cookies



- Vérifier que la page de résultat contient un élément dont l'**id** est « filter-radius ».
- Cliquer sur le lien hypertexte *date* pour classer les résultats par date.

- La page de résultats contient des offres dans des blocs dont la classe est *job_seen_beacon* :

  Importer *java.util.List*, puis utiliser *findElement**s*** et un sélecteur CSS pour récupérer la liste de tous les éléments div de la classe *job_seen_beacon*. Parcourir les offres afin qu'elles défilent dans la page, en utilisant *Actions* et *moveToElement*. Faire une pause de 1 seconde entre chaque déplacement.

Livrables à déposer sur Ametice en indiquant les noms du binôme

- Copies d'écran montrant l'exécution des tests.
- Fichier IndeedTest.java

Références

Selenium WebDriver : https://www.selenium.dev/documentation/webdriver/

API Java WebDriver : https://seleniumhq.github.io/selenium/docs/api/java/