Зображення, що містить текст

Автоматично згенерований опис

**Лабораторна робота №2**

**Тема:** «Створення проекту з автоматичного тестування та вивчення базових локаторів елементів».

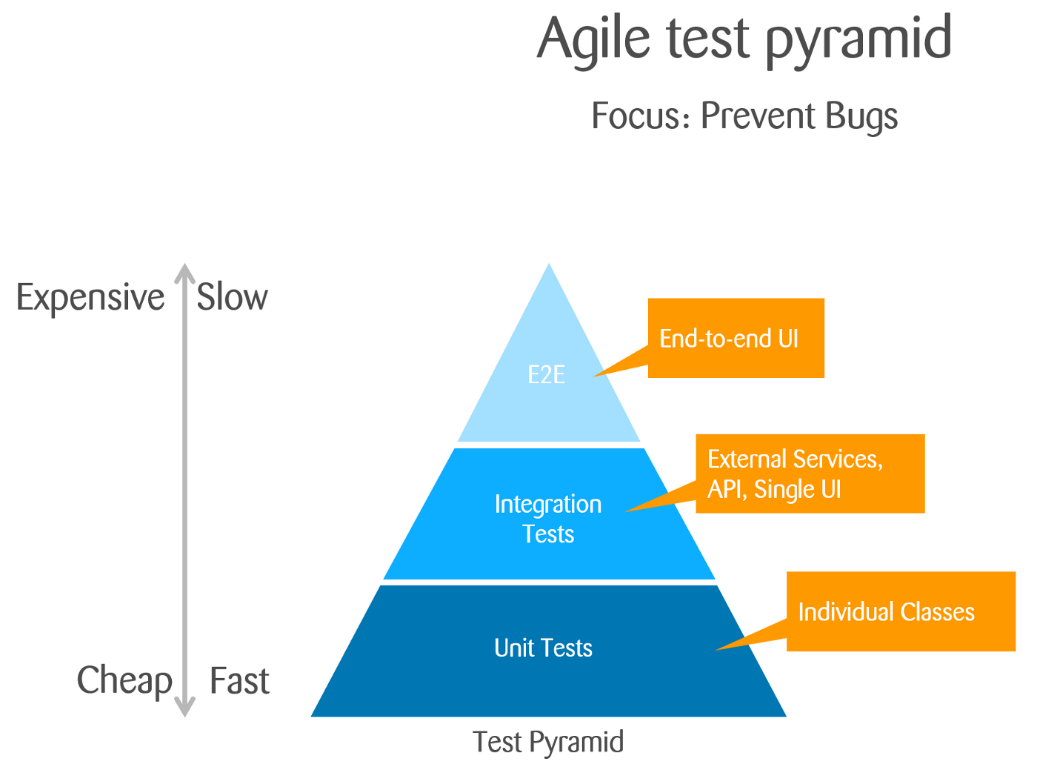
**Мета:** Створити власний проєкт для тестування; як приклад спробувати протестувати сайт університету; вивчити різницю між локаторами та ознайомитись з опціями драйверів.

**Теоретична частина**

*Тестування імітуючи дії користувача.* Такий вид тестування має свої позитивні та негативні сторони, але є дуже важливим для веб-застосунків для підтримки верифікації та валидації продукту. Будь-який продукт потребує мануального тестування, але коли доставка версій застосунка іде швидше аніж мануальний тестувальник перевірить, що новий функціонал не вплинув на старий та коректно працює, то приходить час подумати про автоматизацію тестування.

Наприклад, якщо один тестувальник без автоматизації у день в середньому може пройти близько 20 тестів, то якщо автоматизувати таку роботу у декілька потоків на декількох окремих віртуальних агентах, то можна перевищити 8 тисяч тестів за 6 годин роботи ( враховуючи, що час відповіді застосунку може бути довшим на складних операціях та навантажувати сервер для відповіді).

Але потрібно пам’ятати, що не будь-який тест можна автоматизувати та є сенс проводити тестування діями користувача. Існує тестова піраміда за Agile, яка демонструє, яку кількість тестів від загальної кількості повинно бути реалізовано саме діями користувача.



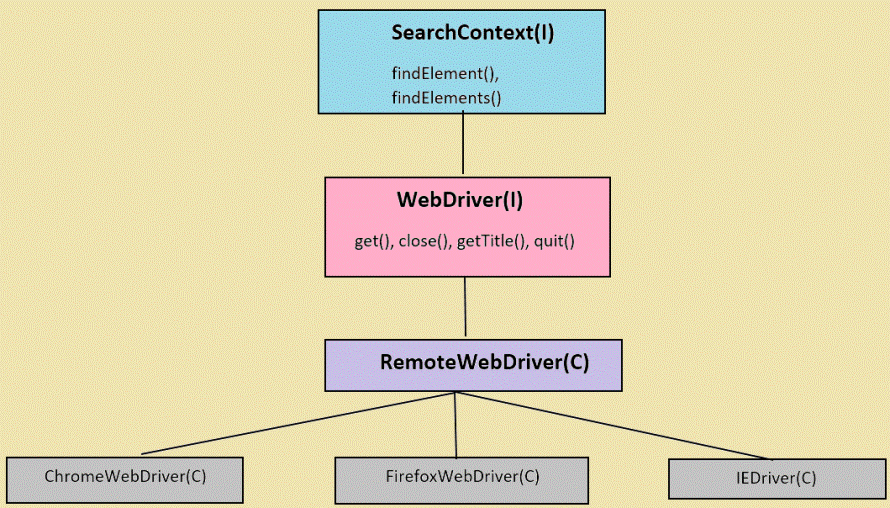
Саме тестування користувачем або його імітацією повинно займати лише 10 відсотків від загальної кількості тестів. У свою чергу, кожен з розробників несе відповідальність за написання модульних тестів для кожного рядка коду. Покриття кожного рядка коду за стандартом повинно бути не менше 85 відсотків, а загальний обсяг тестів відносно інших рівнів тестування повинен бути приблизно 60 відсотків або більше.

*Що таке Selenium?* Отже, саме для того, щоб зімітувати користувача було розроблено інструмент, який дозволяє керувати діями комп’ютера, а саме надсилати події до застосунків та знаходити елементи.

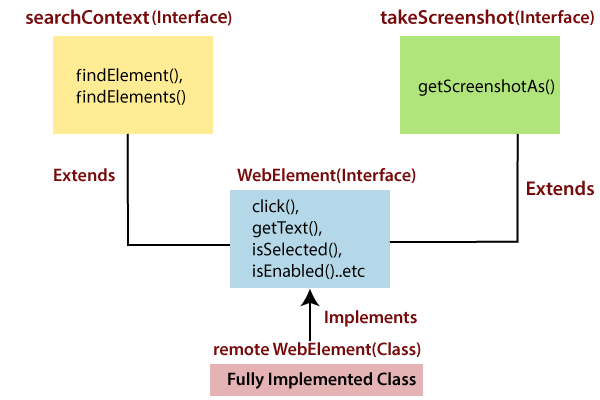
Цей інструмент надає лише можливість працювати з елементами – тому перед тестувальником виникає багато питань щодо реалізації певних дій користувача та створення архітектури проєкту. Існує такі продукти Selenium:

* + **Selenium IDE** — інтегроване середовище розробки у вигляді Firefox-додатка, який дозволяє записувати та відтворювати тести в Firefox 2+.
  + **Selenium Client API** — набір API, що дозволяє писати тести на Java, C#, Ruby, JavaScript та Python.
  + **Selenium Remote Control** — це клієнт / серверна система, яка дозволяє керувати веб-браузерами локально або на іншому комп'ютері, використовуючи практично будь-яку мову програмування та тестування системи.
  + **Selenium WebDriver** — драйвер що дозволяє керувати веб-браузером за допомогою Selenese або API.
  + **Selenium Grid** — дозволяє одночасно запускати тести на кількох серверах та типах веб-браузерів зменшуючи час на тестування.

У цій лабораторній роботі ми будемо використовувати саме драйвер та його функційність. Нижче наведена діаграма ієрархії класу WebDriver:

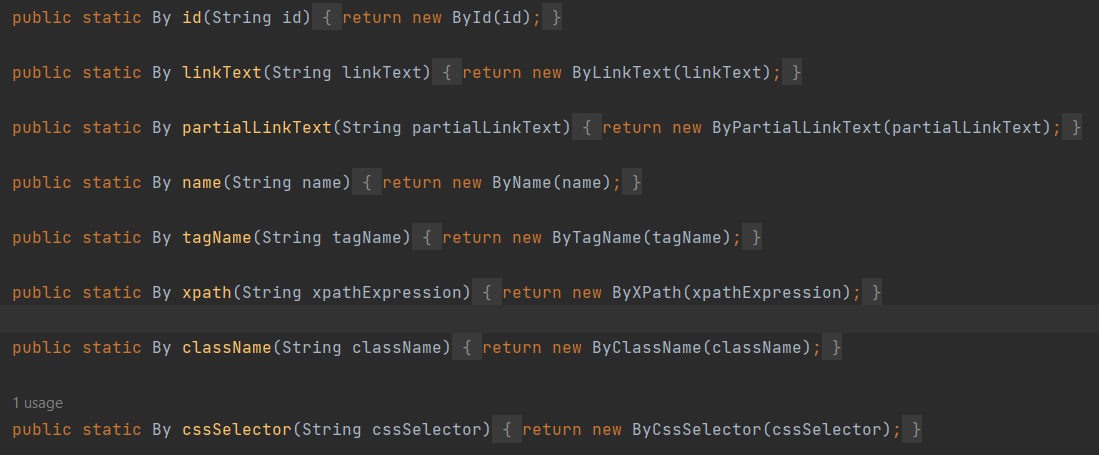


Саме він буде запускати браузер та знаходити елементи на сторінках. В свою чергу SearchContext є інтерфейсом, який має таку структуру:



Також Selenium має такі класи як Actions, KeyInput, Select, Urls, Credential, By, Point, Rectangle, Pdf, ChromeOptions, FluentWait, WebDriverWait та багато інших. Але розглянемо ми лише By та ChromeOptions.

Клас By потрібен для пошуку елемента певним способом, нижче наведені методи цього класу:



Клас ChromeOptions використовується для запуску GoogleChrome з параметрами. Наприклад, в повному екрані, інкогніто моді, без відображення вікна браузера, з використанням додаткових інструментів, профілів та тощо.

*Що таке TestNG?* TestNG – це тестовий фреймворк, створений Cédric Beust, допомагає задовольнити значну кількість потреб AQA & Dev у тестуванні. А саме, TestNG дає розробнику можливість створювати більш гнучкі та потужні тести. TestNG широко використовується разом з Selenium. TestNG схожий на JUnit, але він більш продуктивний, коли справа стосується управління потоком виконання Вашої програми. Архітектура фреймворку допомагає нам зробити тести більш структурованими та забезпечити кращі точки валідації. Серед властивостей, які надають переваги для тестування цей фреймворк надає наступні:

* Потужні і різноманітні анотації для підтримки тест-кейсів.
* TestNG використовує більше можливостей Java і OOП
* Розділяє тестовий код часу компіляції і інформацію про конфігурацію даних часу виконання.
* Паралельне виконання тестів, використання залежностей між тестами. Скажімо тестування навантаження і часткової відмови.
* Гнучкість виконання тестів з різними наборами даних, через файл TestNG.xml або через концепцію постачальників даних (data-provider).
* Угруповання і пріоритизація тест-кейсів.
* Генерація HTML-звітів, встановлення додаткового ПЗ з різних плагінів.
* Особливо важливо гнучкий плагін API.
* Генерація логів виконання тестів.
* Легка інтеграція з Eclipse, Maven, Jenkins і ін.

Саме анотації реалізують функційність цього фреймворку. Ось список анотацій, які підтримує TestNG:

* + @BeforeSuite. Анотований метод буде запущений лише один раз до запуску всіх тестів цього набору.
  + @AfterSuite. Анотований метод буде запущений лише один раз після виконання всіх тестів цього набору.
  + @BeforeClass. Анотований метод буде запущений лише один раз, перш ніж буде викликано перший тестовий метод у поточному класі.
  + @ AfterClass. Анотований метод буде запущено лише один раз після запуску всіх тестових методів у поточному класі.
  + @BeforeTest. Анотований метод буде запущений до запуску будь-якого тестового методу, що належить класам усередині тега <test>.
  + @AfterTest. Анотований метод буде запущено після запуску всіх тестових методів, що належать класам усередині тега <test>.
  + @BeforeGroups. Список груп, які цей спосіб конфігурації буде запускати раніше. Цей метод гарантовано буде запущений незадовго до того, як буде викликано перший метод тестування, який належить будь-якій із цих груп.
  + @AfterGroups. Список груп, після яких запускатиметься цей метод конфігурації. Цей метод гарантовано буде запущено незабаром після виклику останнього методу тестування, що належить будь-якій із цих груп.
  + @BeforeMethod. Анотований метод запускатиметься перед кожним тестовим методом.
  + @AfterMethod. Анотований метод запускатиметься після кожного методу тестування.
  + @DataProvider. Зазначає метод як надання даних методу тестування. Анотований метод повинен повертати об'єкт [] [], де кожному об'єкту [] можна призначити список параметрів тестування. Метод @Test, який хоче отримувати дані від цього DataProvider, повинен використовувати ім'я dataProvider, що дорівнює імені цієї анотації.
  + @Factory. Позначає метод як фабрику, яка повертає об'єкти, які будуть використовуватися TestNG як тестові класи. Метод має повертати Object [].
  + @Listeners. Визначає слухачів на тестовому класі.
  + @Parameters. Описує, як передавати параметри метод @Test.
  + @ Test. Позначає клас чи метод як частину тесту.

*Бібліотека WebDriverManager.* WebDriverManager — це бібліотека Java з відкритим вихідним кодом, яка повністю автоматизовано керує (тобто завантажує, налаштовує та підтримує) драйвери, необхідні для Selenium WebDriver. Крім того, WebDriverManager надає інші відповідні функції, такі як можливість виявлення браузерів, установлених у локальній системі, створення об’єктів WebDriver і безперебійний запуск браузерів у контейнерах Docker. Тобто ця бібліотека полегшує процес розробки проєкту для тестування веб-застосунку тим, що розробнику не потрібно буде піклуватися про встановку драйвера, складання його в ресурси проєкту та про вказання шляху до кожного драйвера для різних браузерів.

*XPath.* XPath є основним елементом стандарту XSLT та його можна використовувати для навігації між елементами та атрибутами в документі XML. Його синтаксис дуже схожий на систему вкладеності папок на комп’ютері.



Більш детальну інформацію про синтаксис можна знайти за посиланням <https://www.w3schools.com/xml/xpath_syntax.asp>. Також ця мова має осі, вузли, предикати та містить понад 200 вбудованих функцій.

Осі – це напрямок навігації по дереві XML документа. Є такі осі:

|  |  |  |
| --- | --- | --- |
| Розширений синтаксис | Скорочений синтаксис | Пояснення |
| ancestor |  | Вибирає всіх предків (батька, прабатька і т. д.) поточного вузла |
| ancestor-or-self |  | Вибирає всіх предків (батька, прабатька і т. д.) поточного вузла і його самого |
| attribute | @ | Вибирає всі атрибути поточного вузла |
| child |  | Вибирає всіх дітей поточного вузла |
| descendant |  | Вибирає всіх нащадків (дітей, внуків і т. д.) поточного вузла |
| descendant-or-self | // | Вибирає всіх нащадків (дітей, внуків і т. д.) поточного вузла і його самого |
| following |  | Вибирає все в документі після закриваючого тегу поточного вузла |
| following-sibling |  | Вибирає всіх «братів» після поточного вузла |
| namespace |  | Вибирає всі вузли простору імен поточного вузла |
| parent | .. | Вибирає всіх батьків поточного вузла |
| preceding |  | Вибирає всі вузли, що з'являлись перед поточним вузлом у документі, крім предків, вузлів з атрибутами і вузлів простору імен |
| preceding-sibling |  | Вибирає всіх «братів» перед поточним вузлом |
| self | . | Вибирає поточний вузол |

Визначення вузла може складатися з конкретних імен вузлів або більш загальних виразів. Іншими словами вузли – це тегі, які вказані в коді.

Функціями вузла є:

* **comment()**  
  знаходить XML вузол з коментарем (наприклад, <!-- Comment -->)
* **text()** з  
  знаходить вузол з відповідним текстом (наприклад, hello world в <k>hello<m> world</m></k>)
* **processing-instruction()**  
  знаходить інструкції з обробки XML документа, такі як <?php echo $a; ?>. В цьому випадку processing-instruction('php') збігатимуться.
* **node()**  
  знаходить будь-який вузол загалом

Предикати, записані у вигляді виразів у квадратних дужках, можуть бути використані для обмеження набору вузлів, для вибору тільки тих вузлів, для яких виконується деяка умова. Наприклад, a[@href='help.php'] вибере ті a елементи (з-поміж дітей контекстного вузла), що містять атрибут href із значенням help.php.

Нема обмеження на кількість предикатів, застосованих на одному кроці, і вони не повинні бути обмежені до останнього кроку в XPath виразі. Також вони можуть бути вкладені на будь-яку глибину. Шлях, визначений предикатами, починається в контексті поточного кроку і не змінює цей контекст. Всі умови повинні виконатися, щоб відбулась дія.

Коли значенням предиката є цифра, вона інтерпретується як тест на позицію (номер) вузла. Таким чином p[1] вибирає перший p елемент-дитину, в той час, як p[last()] вибирає останню.

В іншим випадках значення предиката є автоматично конвертовані до булевих. Коли предикат оцінює множину вузлів, результат є true коли множина вузлів не є порожньою. Отже, p[@x] вибирає ті p елементи, котрі обов’язково мають атрибут x, не важливо з яким значенням.

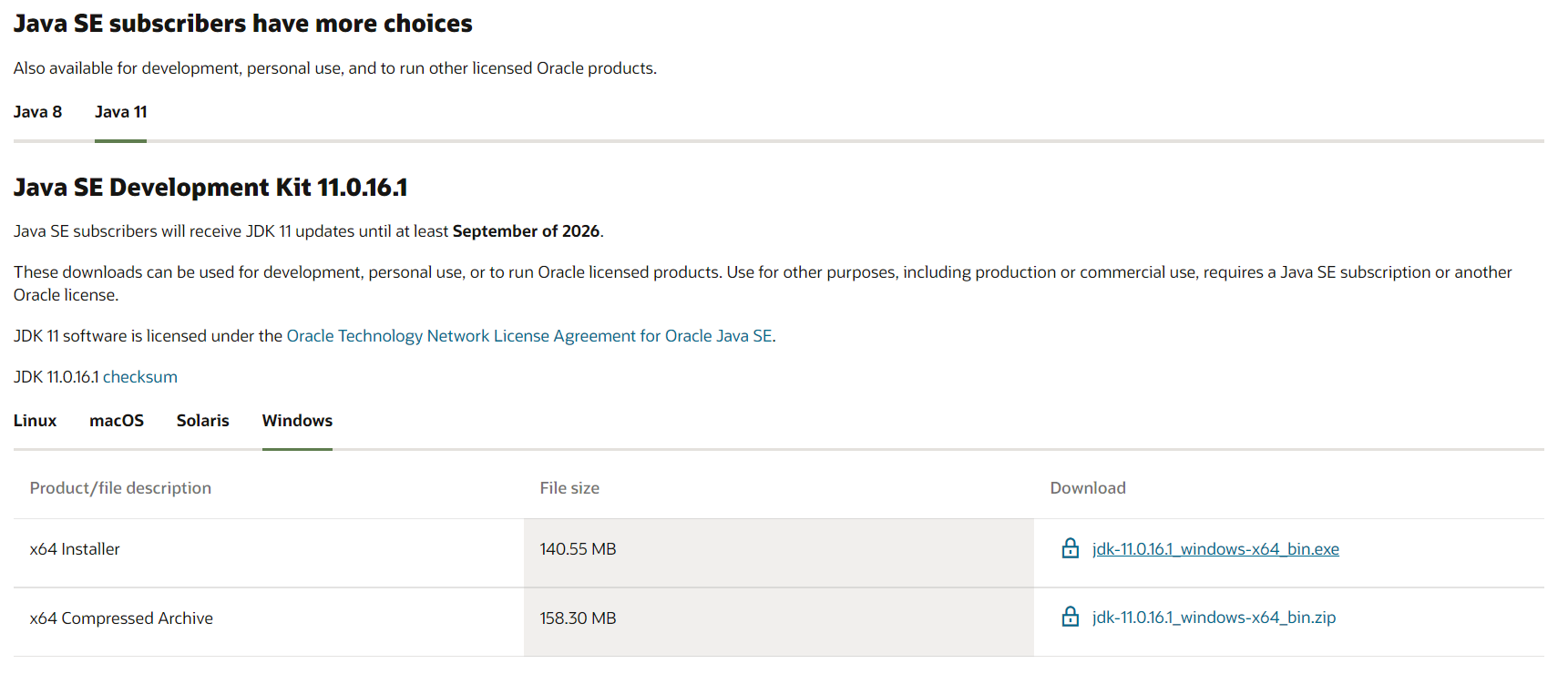
*CSS Selectors.* Нижче наведена таблиця усіх селекторів за стилем:

| Селектор | Приклад | Опис |
| --- | --- | --- |
| [.class](https://www.w3schools.com/cssref/sel_class.asp) | .intro | Обирає елемент з *class="intro"* |
| .class1.class2 | .name1.name2 | Обирає елемент з обома *name1* та *name2*  назвами класу у атрибуті елемента |
| .class1 .class2 | .name1 .name2 | Обирає всі елементи з *name2*, що єнаслідником елемента з *name1* |
| [#id](https://www.w3schools.com/cssref/sel_id.asp) | #firstname | Обирає елемент з *id="firstname"* |
| [\*](https://www.w3schools.com/cssref/sel_all.asp) | \* | Обирає всі елементи |
| [element](https://www.w3schools.com/cssref/sel_element.asp) | p | Обирає всі <p> елементи |
| [[element](https://www.w3schools.com/cssref/sel_element.asp).class](https://www.w3schools.com/cssref/sel_element_class.asp) | p.intro | Обирає всі <p> елементи з *class="intro"* |
| [[element](https://www.w3schools.com/cssref/sel_element.asp), [element](https://www.w3schools.com/cssref/sel_element.asp)](https://www.w3schools.com/cssref/sel_element_comma.asp) | div, p | Обирає всі <div> та <p> елементи |
| [element](https://www.w3schools.com/cssref/sel_element.asp)1 [element](https://www.w3schools.com/cssref/sel_element.asp)2 | div p | Обирає всі <p> елементи з <div> всередині |
| [[element](https://www.w3schools.com/cssref/sel_element.asp)>[element](https://www.w3schools.com/cssref/sel_element.asp)](https://www.w3schools.com/cssref/sel_element_gt.asp) | div > p | Обирає всі <p> елементи де батько <div> елемент |
| [[element](https://www.w3schools.com/cssref/sel_element.asp)+[element](https://www.w3schools.com/cssref/sel_element.asp)](https://www.w3schools.com/cssref/sel_element_pluss.asp) | div + p | Обирає перший <p> елемент, який розташован одразу після <div> елементу |
| [[element](https://www.w3schools.com/cssref/sel_element.asp)1~[element](https://www.w3schools.com/cssref/sel_element.asp)2](https://www.w3schools.com/cssref/sel_gen_sibling.asp) | p ~ ul | Обирає кожен <ul> елемент, який йде перед <p> елементом |
| [[attribute]](https://www.w3schools.com/cssref/sel_attribute.asp) | [target] | Обирає всі елементи з target атрибутом |
| [[attribute=value]](https://www.w3schools.com/cssref/sel_attribute_value.asp) | [target=\_blank] | Обирає всі елементи з target="\_blank" |
| [[attribute~=value]](https://www.w3schools.com/cssref/sel_attribute_value_contains.asp) | [title~=flower] | Обирає всі елементи з title атрибутом, який має в собі слово"flower" |
| [[attribute|=value]](https://www.w3schools.com/cssref/sel_attribute_value_lang.asp) | [lang|=en] | Обирає всі елементи з мовним атрібутом, який дорівнює "en" або поччинається з "en-" |
| [[attribute^=value]](https://www.w3schools.com/cssref/sel_attr_begin.asp) | a[href^="https"] | Обирає кожен <a> елемент, чиє значення href атрибута починається з "https" |
| [[attribute$=value]](https://www.w3schools.com/cssref/sel_attr_end.asp) | a[href$=".pdf"] | Обирає кожен <a> елемент, чиє значення href атрибута закінчується ".pdf" |
| [[attribute\*=value]](https://www.w3schools.com/cssref/sel_attr_contain.asp) | a[href\*="w3schools"] | Обирає кожен <a> елемент, чиє значення href атрибута має в собі "w3schools" |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Обирає активне посилання |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Вставляє щось після кожного <p> елемента |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Вставляє щось перед кожним <p> елементом |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Обирає кожен відмічений <input> елемент |
| [:default](https://www.w3schools.com/cssref/sel_default.asp) | input:default | Обирає дефолтний <input> елемент |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Обирає кожен заблокований <input> елемент |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Обирає кожен <p> , який не має дітей |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Обирає кожен доступний <input> елемент |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Обирає кожен <p> елемент, який є першою дитиною свого батька |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Обирає першу літеру кожного <p> елементу |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Обирає першу лінію кожного <p> елементу |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Обирає кожен <p> елемент, який є першим <p> елементом свого батька |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Обирає input елемент, на якому є фокус |
| [:fullscreen](https://www.w3schools.com/cssref/sel_fullscreen.asp) | :fullscreen | Обирає елемент який має full-screen мод |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Обирає лінки, на які наведена миша |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Обирає input елементи зі значенням в певному інтервалі |
| [:indeterminate](https://www.w3schools.com/cssref/sel_indeterminate.asp) | input:indeterminate | Обирає input елементи, які у indeterminate стані |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Обирає всі input елементи з невалідним значенням |
| [:lang(language)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Обирає кожен <p> елемент, мова якого дорівнює "it" (італійська) |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Обирає кожен <p> елемент, який є останньою дитиною батька |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Обирає кожен <p> елемент, який є останнім<p> елементом свого батька |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Обирає всі не відвідані посилання |
| [::marker](https://www.w3schools.com/cssref/sel_marker.asp) | ::marker | Обирає маркери елементів списку |
| [:not(selector)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Обирає кожен елемент, який не є<p> елементом |
| [:nth-child(n)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Обирає кожен <p> елемент, який є другою дитиною батька |
| [:nth-last-child(n)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Обирає кожен <p> елемент, який є другою дитиною батька з кінця |
| [:nth-last-of-type(n)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Обирає кожен <p> елемент, який є другим <p> елементом батька з кінця |
| [:nth-of-type(n)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Обирає кожен <p> елемент, який є другим <p> елементом батька |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Обирає лише <p> елементи, які є дитиною батька |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Обирає кожен <p> елемент, який є дитиною якогось батька |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Обирає input елементи без " required" атрибуту |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Обирає input елементи зі значенням в поза певного інтервалу |
| [::placeholder](https://www.w3schools.com/cssref/sel_placeholder.asp) | input::placeholder | Обирає input елементи з "placeholder" атрибутом |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Обирає input елементи з "readonly" атрибутом |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Обирає input елементи без "readonly" атрибуту |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Обирає input елементи з "required" атрибутом |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | :root | Обирає коріння документу |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | ::selection | Обирає елемент, який був обраний користувачем |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | Обирає активний #news елемент (ім’я якоря) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Обирає всі input елементи з правильним значенням |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Обирає всі відвідані лінки |

**Практична частина**

Спершу треба встановити Java 11 або 17:

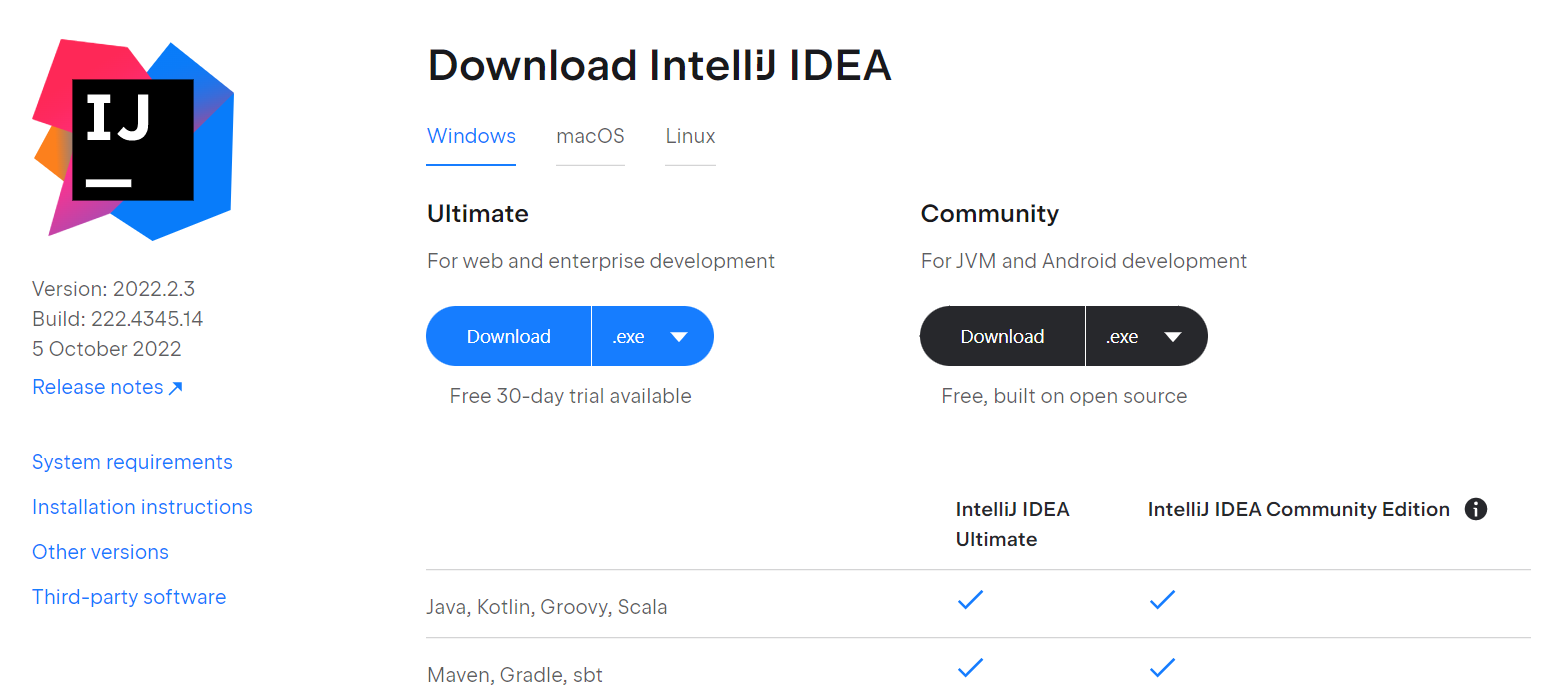
* Java 11  
  (<https://www.oracle.com/java/technologies/downloads/#java11-windows>)



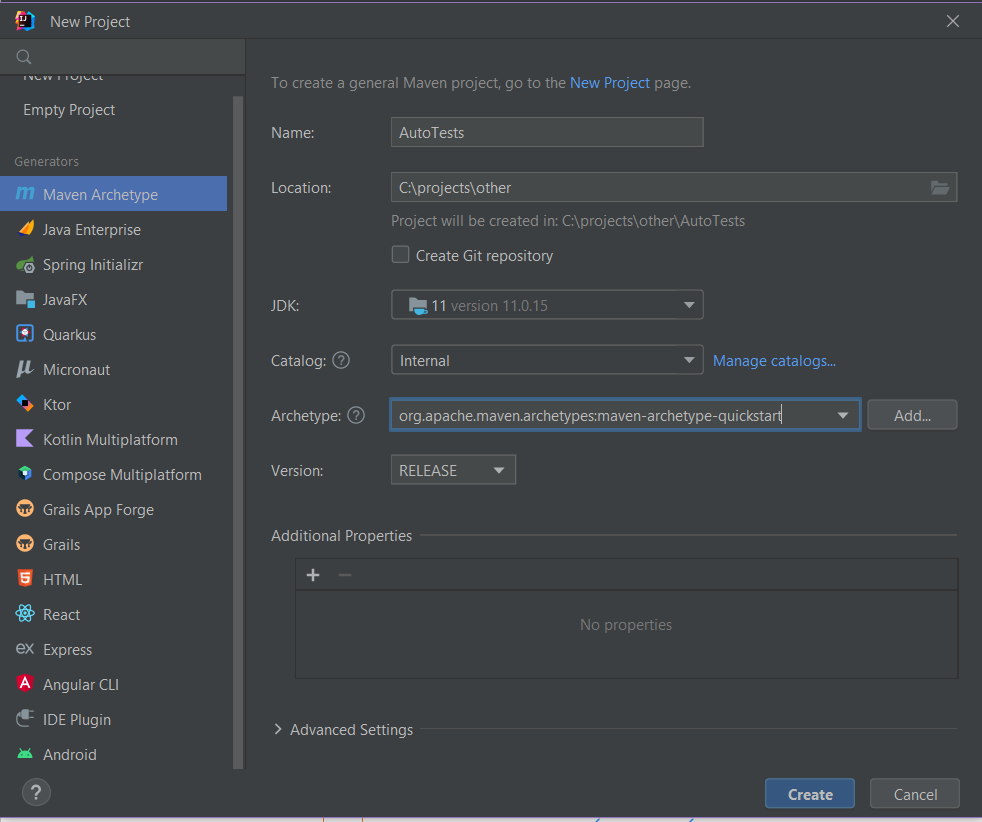
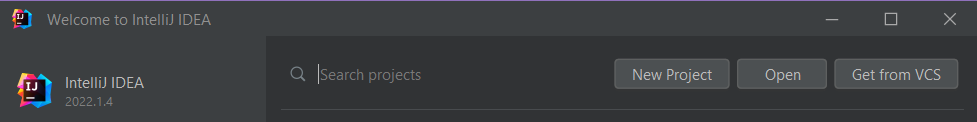
* Java 17  
  (<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>)



Далі необхідно встановити відповідний редактор, який має вже за замовчуванням Maven для збірки проєкту – це IntelliJ IDEA (<https://www.jetbrains.com/idea/download/#section=windows>).



Після встановлення оточення можна створювати проєкт для тестування веб-застосунку або будь-якого сайту. Для цього необхідно відкрити IntelliJ IDEA та натиснути «New Project». Далі обрати Maven як інструмент для збірки і ввести усі параметри як вказано нижче (назву проєкту можна встановити за бажанням будь-яку):

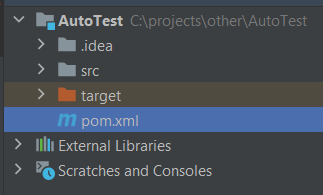


У дереві проєкту треба знайти файл pom.xml (у разі, якщо він не відкрився автоматично після відкриття проєкту) та додати залежності, щоб імпортувати усі необхідні для роботи бібліотеки та фреймворки. Усі залежності можна знайти на офіційному сайті репозиторія Maven або скористатися посиланнями нижче:

* Selenium <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.5.0>
* TestNG <https://mvnrepository.com/artifact/org.testng/testng/7.1.0>
* WebDriverManager <https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager>

Слідуйте скріншотам нижче, щоб додати залежності:

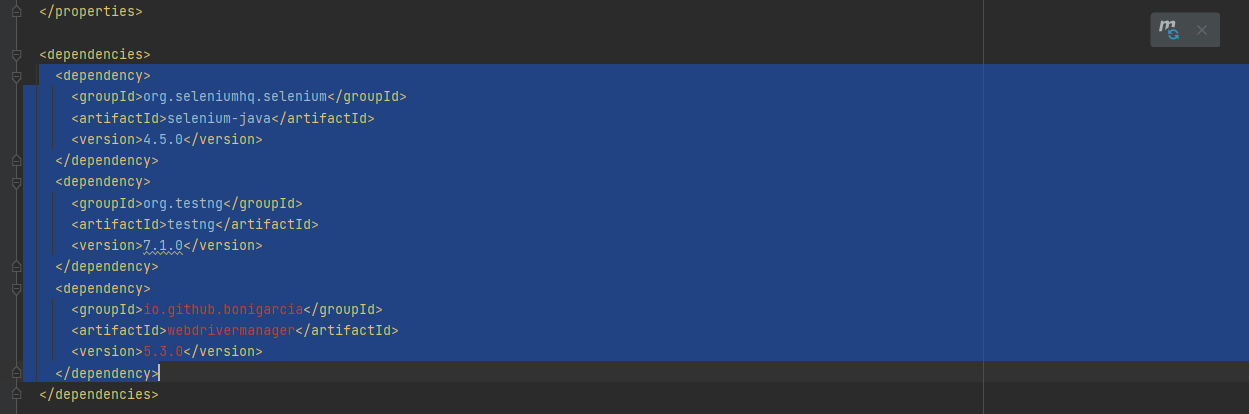
* Відкрити pom.xml;



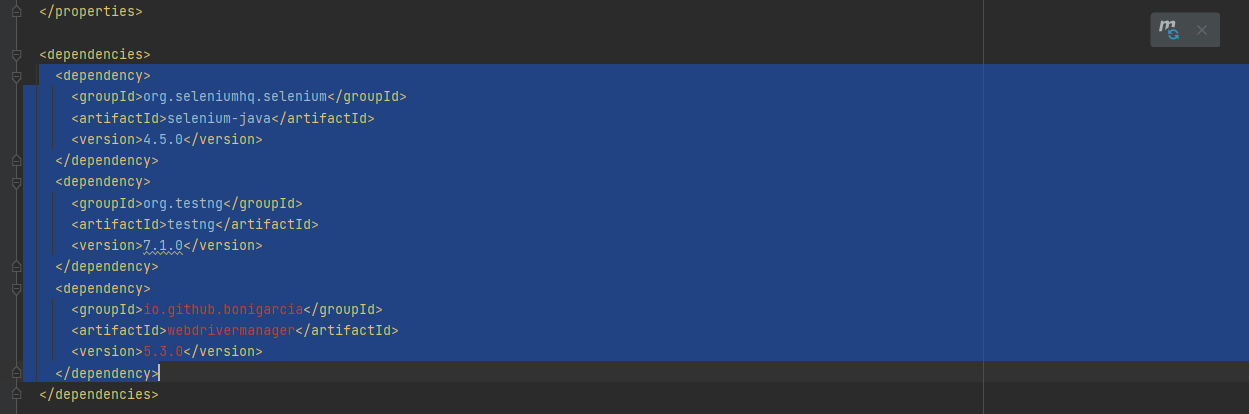
* Перейти за посиланням та скопіювати xml код;

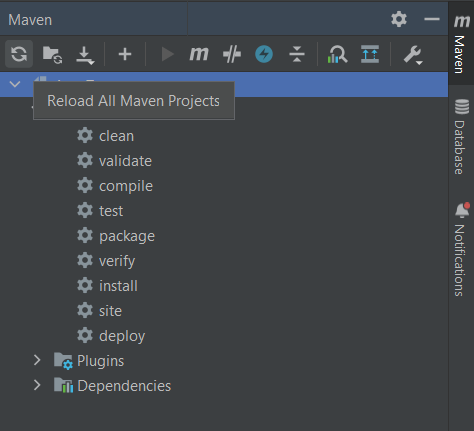


* Вставити усі залежності в блок <dependencies></dependencies>



* Щоб Maven підтягнув усі необхідні файли треба натиснути кнопку, яка з’явится після кожної зміни pom.xml файлу. Або відкрити зліва вкладинку Maven та натиснути на “Reload All Maven Projects”





Надалі можна буде обрати будь-який сайт для тестування, але для прикладу було використано сайт нашого університету – Національний технічний університет «Дніпровська політехніка», <https://www.nmu.org.ua/ua/> (станом на 03.10.22).

У створеному проєкті на основі артефакту вже є необхідна структура папок, тому створимо файл для першої лабораторної роботи. Для цього натисніть правою кнопкою миші на папку, у якій треба створити файл. Оберіть пункт “New”, а в новому випадаючому меню оберіть формат файлу “Java Class”.

Зображення, що містить текст

Автоматично згенерований опис

В новому файлі необхідно створити клас для методів, які будуть тестувати обраний сайт. В цьому класі будуть знаходитись змінні, які потрібні в декількох методах, серед них – базова адреса сайту та веб драйвер, який саме буде імітувати користувача.

Зображення, що містить текст

Автоматично згенерований опис

Далі використаємо анотації TestNG для того, щоб зробити передумови для запуску тестів. Щоб не створювати кожного разу нове вікно браузера будемо створювати декілька передумов, а саме:

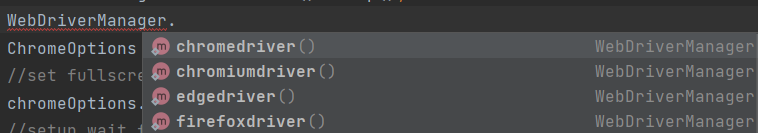
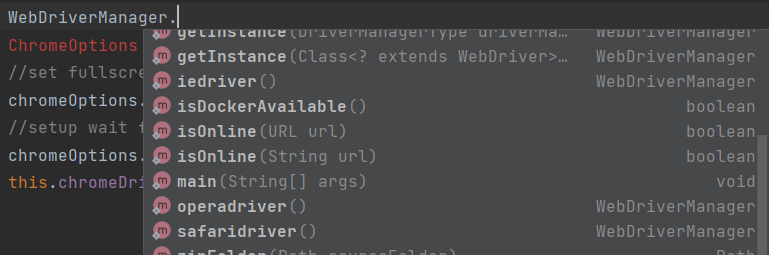
* Умови для відкриття браузера – один раз на початку тестування;
* Умови для переходу на початкову сторінку сайту – для кожного тесту;

Отже перші умови реалізуються наступним чином:

Зображення, що містить текст

Автоматично згенерований опис

Тут використовується WebDriverManager для того, щоб полегшити відкриття браузеру. За відсутності такого менеджера необхідно було б завантажувати собі драйвер певного браузера та вказувати шлях до нього в ChromeOptions. Як вже можна було помітити, наразі використовується GoogleChrome у якості браузера, але можна обрати будь-який з тих, що існують в менеджері.



Зображення, що містить текст

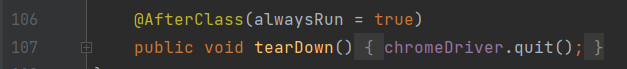
Автоматично згенерований опис

Наступні передумови реалізовані наступним чином:

Зображення, що містить текст

Автоматично згенерований опис

Також після завершення тестування необхідно закрити вікно браузера:

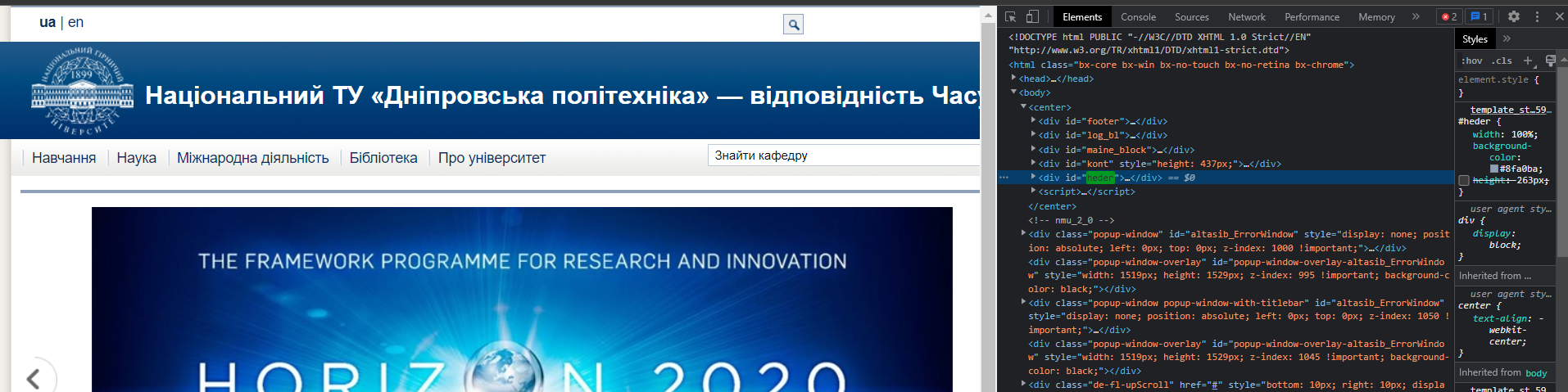


Наступним кроком є написання тестів. Як приклад знаходження елементу за його ідентифікаційним словом, спробуємо знайти хедер на сайті:

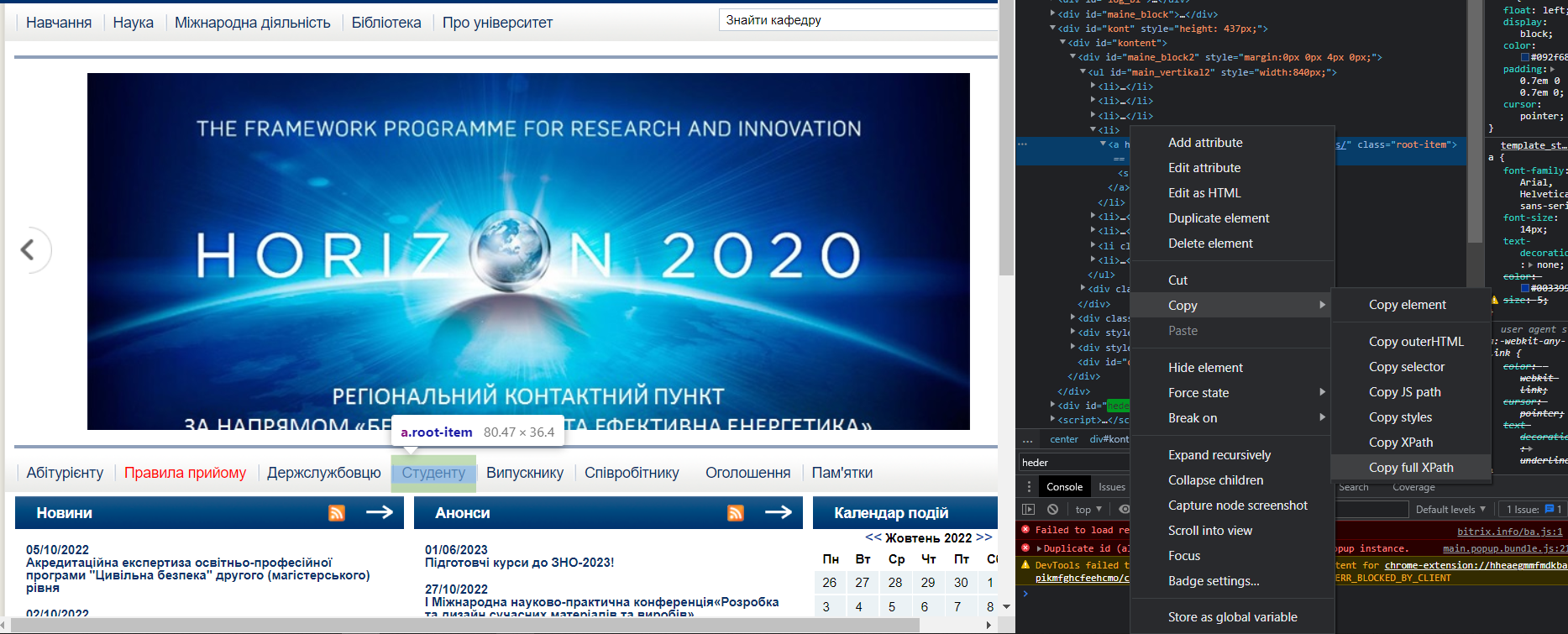
Зображення, що містить текст

Автоматично згенерований опис

Цей тест впаде. Якщо піти на сайт та за допомогою DevTools (Ctrl+Shift+I) спробувати знайти шапку у HTML коді сторінки, то виявиться, що насправді шапка – це нижня частина сторінки, яка має помилку у слові.



Далі спробуємо знайти елемент за допомогою його XPath. Для цього необхідно знайти елемент у DOM та натиснути на нього правою кнопкою миші і обрати пункт «Копіювати повний XPath».



Цей шлях до елементу необхідно буде додати до наступного тесту, в якому ми перевіримо перехід на іншу сторінку по кліку на цю кнопку:

Зображення, що містить текст

Автоматично згенерований опис

Наступний тест буде демонструвати знаходження елементу за назвою його тегу. Знайдемо будь-яке перше поле для введення по тегу «input» на сторінці «Студентам» та виведемо його параметри, які має кожен WebЕлемент Selenium-у. А потім намагатимемось ввести щось в це поле та натиснути «Enter», щоб підтвердити введення. Однак, цей тест покаже негативний результат на етапі перевірки тексту в полі.

Зображення, що містить текст

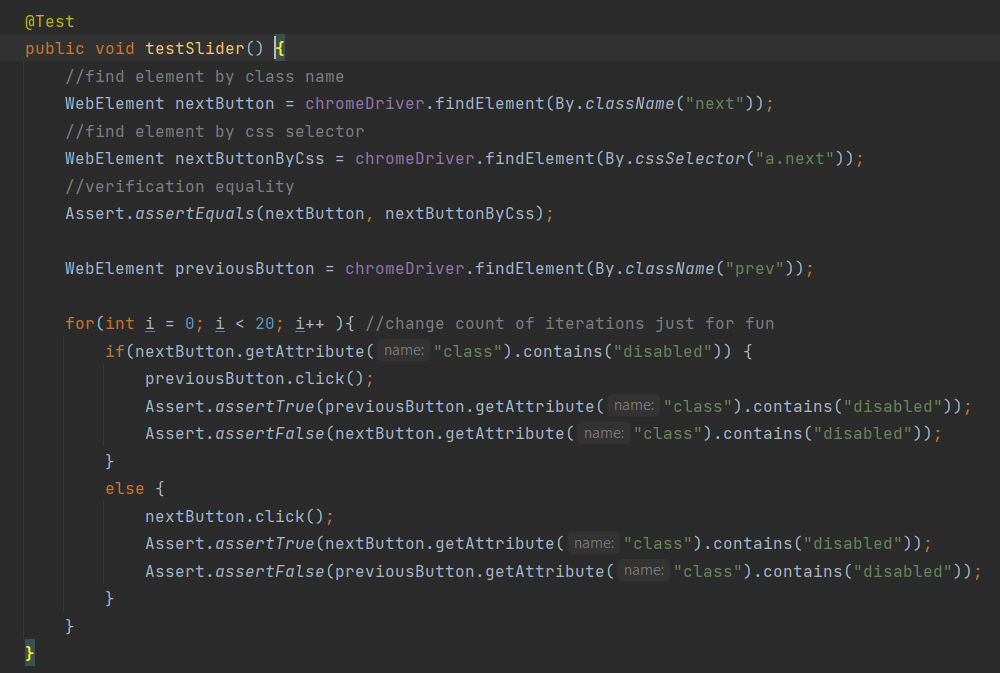
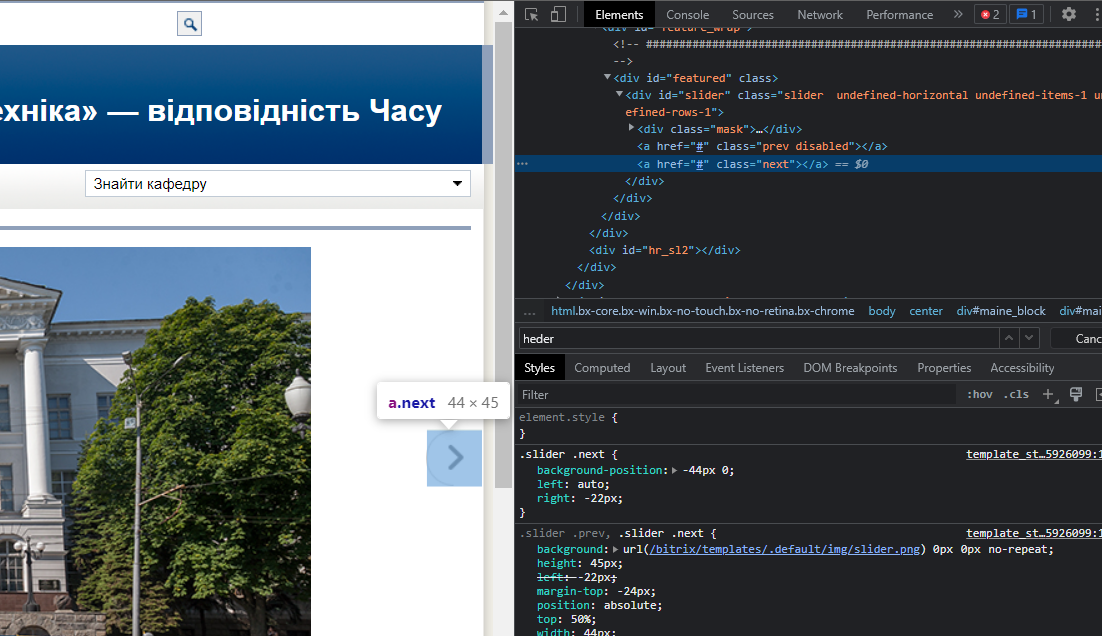
Автоматично згенерований опис

Якщо піти до сторінки сайту та знайти поле введення, то одразу стає зрозуміло, що це не те поле введення, яке ми очікували знайти на сторінці – це поле не буде мати жодного тексту ніколи, а на будь які дії буде відкривати сторінку для пошуку. Тобто насправді це не поле для вводу, а лише кнопка, яку зробили таким чином.

Зображення, що містить текст

Автоматично згенерований опис

Наступний тест демонструватиме як можна знаходити по назві класу елемента або за його СSS селектором. Знайдемо кнопку слайдера на сторінці, яка має назву класу та будемо намагатися знайти її обома способами та порівняти знайдені елементи. Потім спробуємо тестування в циклі – будемо натискати кнопки почергово в залежності від того, яка з них є активною.



**Завдання для виконання**

У ході цієї лабораторної необхідно відтворити усі приклади тестів, які були наведені вище та створити тестовий сценарій для будь-якого веб-ресурсу. Тестовий сценарій для виконання повинен містити такі дії:

* Клік по елементу;
* Введення даних у поле та перевірка поля на наявність даних;
* Знаходження елементу за допомогою не прямого XPath (використовуючи функції або унікальні ідентифікатори);
* Перевірку будь-якої умови.

Для виконання такого сценарію найбільш підходять сайти, які мають в собі пошукове поле, тому варто обрати інтернет-магазин або сайт, який має великі обсяги даних.

Звіт до лабораторної роботи повинен містити:

* Титульну сторінку;
* Мету та тему роботи;
* Опис та скріншоти роботи над власним тестовим сценарієм;
* Посилання на репозиторій з кодом лабораторної роботи;
* Висновки та короткі відповіді на питання для контролю.

**Питання для контролю**

1. Що таке XPath? Наскільки доцільно використовувати повний шлях до елемента?
2. Для чого потрібен WebDriverManager?
3. Чи є сенс у автоматизованому тестуванні імітуванням користувача ігрових застосунків?
4. В чому різниця методів close() та quit() класу WebDriver?
5. Що таке Implicit wait?
6. Які опції для браузера можна використати? Назвіть щонайменше 3 таких.