# **Lab: Selenium Waits**

## Objectives

* Observe the **consequences of not using waits**.
* Understand the **different types of waits in Selenium**.
* Learn how to **use implicit, explicit, and fluent waits** effectively.

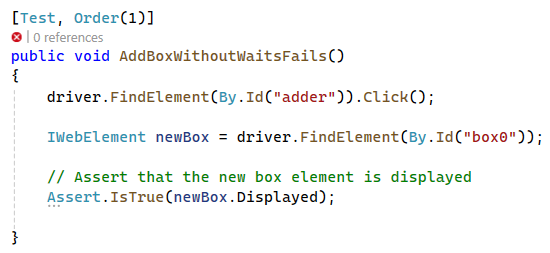
## Prerequisites

* Create a new NUnit test project.
* Install the necessary Selenium packages via NuGet:   
  Selenium.WebDriver, Selenium.WebDriver.ChromeDriver, Selenium.Support.
* Initialize the ChromeDriver and navigate to the application URL.
* All the tests will be performed on the following URL: [**https://www.selenium.dev/selenium/web/dynamic.html**](https://www.selenium.dev/selenium/web/dynamic.html)

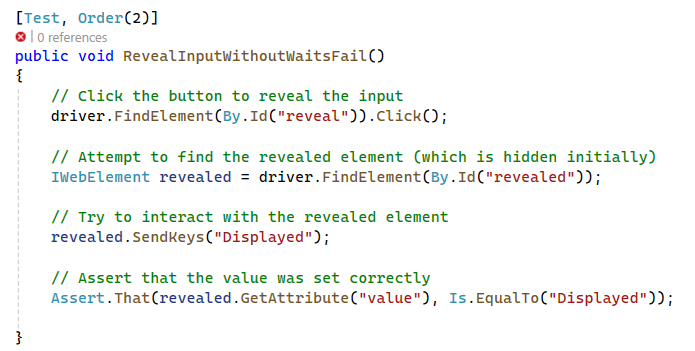
## Test Without Waits

**Observe failures due to elements not being immediately available.**

* Open your test project and create a new test called **AddBoxWithoutWaitsFails**().
* Click the button to add a new box element.
* Attempt to find the new box element.
* Assert that the new box element is displayed.



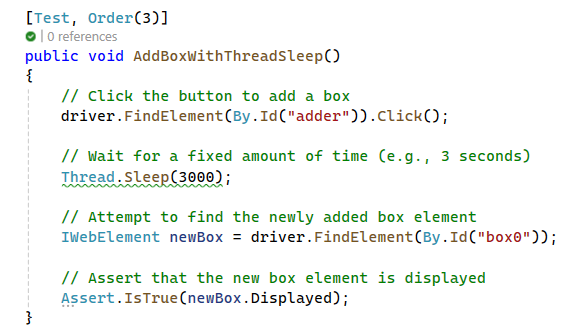
* Create a new test called **RevealInputWithoutWaitsFail().**
* Click the button to reveal the input element.
* Attempt to find the input element.
* Interact with the input element and assert its value.



**Note:** These tests will fail because the elements are not ready for interaction without waits.

## Using Thread.Sleep

* Create a new test called **AddBoxWithThreadSleep().**
* Click the button to add a new box element.
* Use **Thread.Sleep** to wait for a fixed amount of time (e.g., 3 seconds).
* Attempt to find the new box element after the sleep period.
* Assert that the new box element is displayed.

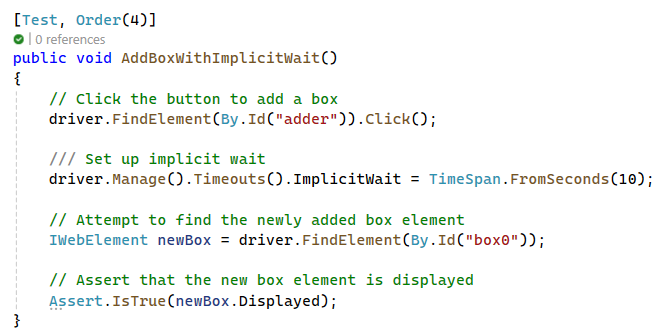


**Note:** While Thread.Sleep can make tests pass by introducing fixed delays, it is inefficient and can make tests slow and unreliable.

## Implicit Waits

You are already familiar with Implicit Waits and have used them a couple of times, but let's practice once again.

* Create a new test method called **AddBoxWithImplicitWait().**
* Set an implicit wait of 10 seconds.
* Click the button to add a new box element.
* Attempt to find the new box element.
* Assert that the new box element is displayed.



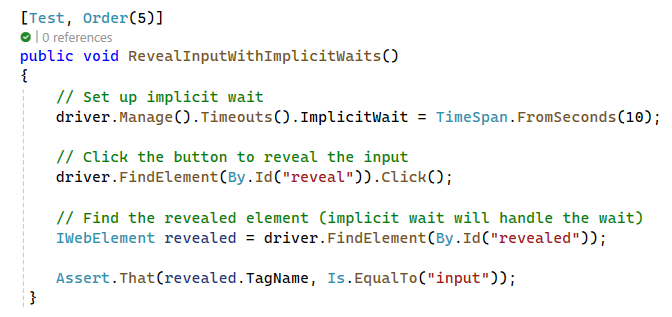
Create a new test method called **RevealInputWithImplicitWaits().**

Set an implicit wait of 10 seconds.

Click the button to reveal the input element.

Attempt to find the input element.

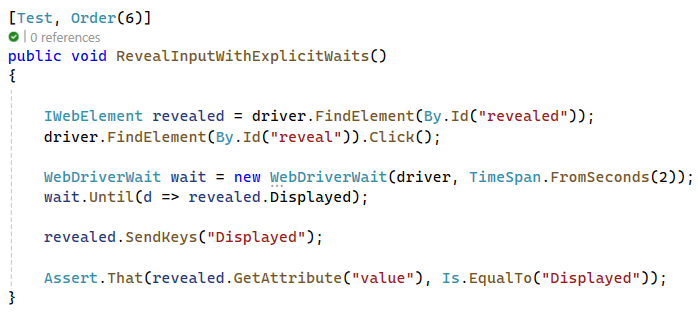
Assert that the input element exists and is interactable by checking its tag name.



**Note:** Implicit waits instruct Selenium to wait up to a specified amount of time before throwing an exception if the element is not found, improving reliability for dynamic content. This wait is set globally, meaning it applies to all elements in the WebDriver session.

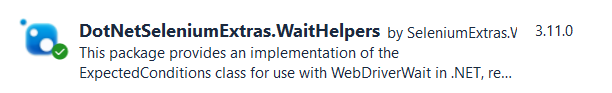
## Explicit Waits

* Create a new test called **RevealInputWithExplicitWaits().**
* Click the button to reveal the input element.
* Use an explicit wait to wait for the input element to be displayed.
* Interact with the input element and assert its value.

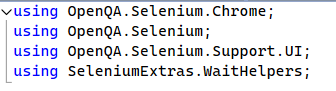


## Fluent Wait with Expected Conditions and Ignored Exceptions

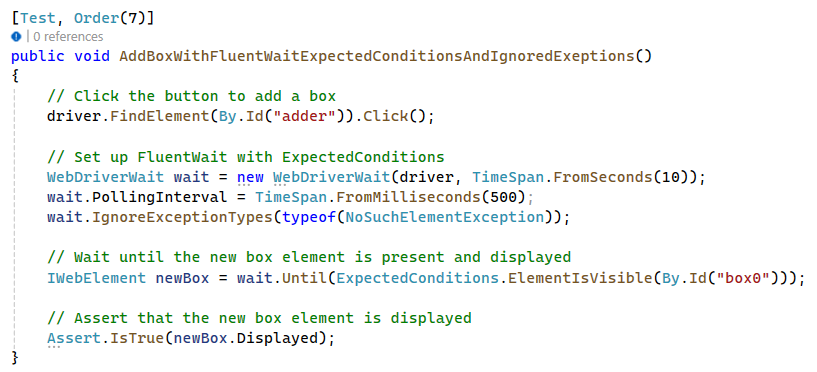
In order to use Expected Conditions, you need to install SeleniumExtras.WaitHelpers via Nuget.



Add it to the necessary namespaces.

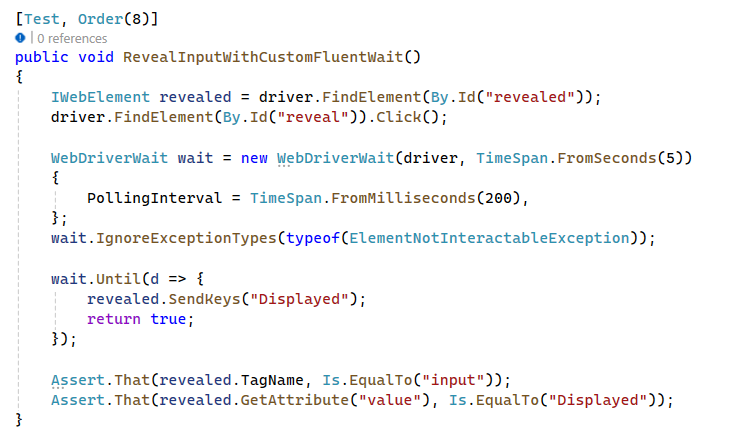


* Create a new test called **AddBoxWithFluentWaitExpectedConditionsAndIgnoredExceptions()**
* Click the button to reveal the input element.
* Create a WebDriverWait instance with a timeout of 10 seconds and a polling interval of 500 milliseconds.
* Configure Fluent Wait to ignore specific exceptions.
* Use ExpectedConditions to wait until the new box element is present and visible.
* Verify that the newly added box element is displayed.



## Custom Waits Conditions

* Create a new test called **RevealInputWithCustomFluentWait().**
* Click the button to reveal the input element.
* Create a WebDriverWait instance with a timeout of 5 seconds and a polling interval of 200 milliseconds.
* Configure Fluent Wait to ignore specific exceptions.
* Use a custom wait condition to send keys to the revealed element.
  + The custom wait condition within wait.Until should be a lambda expression that tries to send keys to the revealed element. The wait should continue until the element is interactable and the keys are sent successfully.
* Verify that the revealed element is displayed and its value is set correctly.



Note: This demonstrates how to create custom conditions that are not covered by predefined expected conditions, providing flexibility in handling unique scenarios in web automation.

## Exceptions

**Modify the First Two Tests to Assert Exceptions**

* Modify the AddBoxWithoutWaitsFails test to assert the NoSuchElementException thrown when the new box element is not found.
* Modify the RevealInputWithoutWaitsFail test to assert the ElementNotInteractableException thrown when the input element is not interactable.