# Ожидания

Ожидания - непременный атрибут любых UI тестов для динамических приложений. Скрипт выполняется намного быстрее реакции приложения на команды, поэтому часто в скриптах необходимо дожидаться определенного состояния приложения для дальнейшего с ним взаимодействия.

Ожидания бывают:

1. **Неявные ожидания** - Implicit Waits - конфигурируют экземпляр WebDriver делать многократные попытки найти элемент (элементы) на странице в течение заданного периода времени, если элемент не найден сразу. Tолько по истечении этого времени WebDriver бросит ElementNotFoundException.

Неявное ожидание указывает WebDriver опрашивать DOM в течение конкретного периода времени при поиске элементов, в случае если они не появились сразу. По умолчанию это значение равно 0. Если значение было однажды установлено, то оно будет использоваться перед выполнением каждой команды findElement, ожидая элемент, на протяжении жизни экземпляра объекта WebDriver, либо пока не будет изменено.

Однако, неявные ожидания не позволяют дожидаться таких событий как, например, исчезновение элемента из DOM или смены свойств элемента. Для таких событий нужно использовать явные ожидания.

**driver**.manage***()***.timeouts***()***.implicitlyWait***(***5, TimeUnit.***SECONDS)***;

Implicit Wait можно использовать для:

* ожидания полной загрузки страницы — pageLoadTimeout();
* ожидания появления элемента на странице — implicitlyWait();
* ожидания выполнения асинхронного запроса — setScriptTimeout();

Установка использования неявного ожидания будет выглядеть следующим образом:

**driver**.manage***()***.timeouts***()***.implicitlyWait***(***Duration.*ofSeconds*(10)***)***;  
**driver**.manage***()***.timeouts***()***.pageLoadTimeout***(***Duration.*ofSeconds*(10)***)***;  
**driver**.manage***()***.timeouts***()***.setScriptTimeout***(***5, TimeUnit.***SECONDS)***; - deprecated

1. **Явные ожидания** - Explicit Waits - это код, который ждет наступления какого-то события прежде чем продолжит выполнение команд скрипта. Такое ожидание срабатывает один раз в указанном месте. Для организации явных ожиданий Selenium предоставляет класс WebDriverWait и интерфейс ExpectedCondition.

О явном ожидании стоит помнить следующие вещи:

* ожидание сработает именно там, где оно указано;
* как и неявному ожиданию, ему необходимо указать лимит времени;
* ожидает выполнения необходимого условия;
* ждет завершения Ajax request.

Использовать явное ожидание можно через WebDriverWait. Инициализация будет происходить следующим образом:

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

где driver является референсом к нашему используемому WebDriver, а число 10 — TimeOut в секундах.

В тесте само ожидание уже будет выглядеть примерно так:

wait.until***(***ExpectedConditions.*visibilityOfElementLocated****(***inboxLocator***))***;

**new** WebDriverWait***(*driver**, Duration.ofSeconds(5)***)*** .until***(***ExpectedConditions.*visibilityOfElementLocated****(***inboxLocator***))***;

По умолчанию WebDriverWait вызывает ExpectedCondition каждые 500 миллисекунд до тех пор, пока условие не будет удовлетворено. Это значение можно изменить, установив свойство pollingEvery(sleepTimeOut, TimeUnit.MILLISECONDS) или используя конструктор WebDriverWait(WebDriver driver,long timeOutInSeconds,long sleepInMillis), принимающий 3-им параметром длительность в миллисекундах между опросами. По умолчанию WebDriverWait игнорирует NotFoundException, возникающее при вызове условия.

**new** WebDriverWait***(*driver**, Duration.ofSeconds(5), Duration.ofMillis(300)***)*** .until***(***ExpectedConditions.*visibilityOfElementLocated****(***inboxLocator***))***;

**Самым худшим вариантом** является использование ``` Thread.sleep(1000)```, в случае с которым скрипт просто будет ждать определенное количество времени. Это не гарантирует наступление нужного события, либо будет слишком избыточным и увеличит время выполнения теста.

Более предпочтительно использовать WebDriverWait и ExpectedCondition:

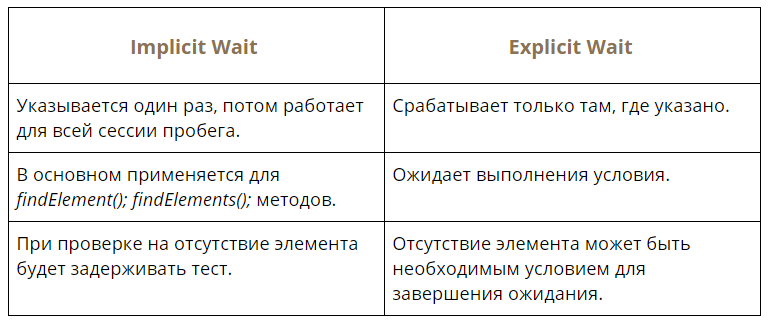
Следует также обратить внимание на класс ExpectedConditions, он предоставляет готовые условия ожидания. Вот некоторые из них:

* **stalenessOf**(WebElement element) — ожидание пока элемент исчезнет из DOM;
* **titleIs**(java.lang.String title) — ожидание заголовка страницы;
* **textToBePresentInElement**(By locator, java.lang.String text) — ожидание заданного текста в элементе;
* **visibilityOfElementLocated**(By locator) — ожидание, что элемент присутствует в DOM и виден.

Вот полный список всего, что вам предлагает подождать ExpectedConditions:

* alertIsPresent()
* elementSelectionStateToBe()
* elementToBeClickable()
* elementToBeSelected()
* frameToBeAvaliableAndSwitchToIt()
* invisibilityOfTheElementLocated()
* invisibilityOfElementWithText()
* presenceOfAllElementsLocatedBy()
* presenceOfElementLocated()
* textToBePresentInElement()
* textToBePresentInElementLocated()
* textToBePresentInElementValue()
* titleIs()
* titleContains()
* visibilityOf()
* visibilityOfAllElements()
* visibilityOfAllElementsLocatedBy()
* visibilityOfElementLocated()

## **Разница между Implicit и Explicit Wait**

[](https://tproger.ru/s3/uploads/2020/11/table.png)

## **Написание собственных ожиданий**

Ожидания Selenium не всегда способны удовлетворить потребности тестировщика. В таких случаях мы можем сами написать методы, которые удержат автотесты от падения. Работать кастомные ожидания будут по тому же принципу, что и Explicit Wait, т.е. срабатывать в той части теста, в которой указаны, а условие, которого необходимо дождаться, нужно переопределить метод apply() интерфейса ExpectedCondition, который наследуется от функционального интерфейса Function.

By newsLinksLocator = By.*cssSelector****(*".tabs-content\_\_item"*)***;  
**new** WebDriverWait***(*driver**, 5***)*** .until***(*new** ExpectedCondition***<***Boolean***>() {***   
 @Override  
 **public** Boolean apply***(***WebDriver driver***) {* return** driver.findElements***(***newsLinksLocator***)***.size***()*** == 10;  
 ***}  
 })***;

Или упростим и напишем все это через лямбду:

**new** WebDriverWait***(*driver**, Duration.ofSeconds(5)***)***.until***(***driver -> driver  
 .findElements***(***newsLinksLocator***)***.size***()*** == 10***)***;

**Тест для коллекции элементов с ожиданием условия видимсоти всех 10 локаторов:**

@Test  
**public void** elementCollectionTest***() {*** By newsLinksLocator = By.*cssSelector****(*".tabs-content\_\_item"*)***;  
 **new** WebDriverWait***(*driver**, 5***)***.until***(***driver -> driver  
 .findElements***(***newsLinksLocator***)***.size***()*** == 10***)***;  
 List***<***WebElement***>*** newsLinks = **driver**.findElements***(***newsLinksLocator***)***;  
 Assert.*assertEquals****(***newsLinks.size***()***, 10***)***;  
 newsLinks.forEach***(***link -> ***{*** Assert.*assertTrue****(***link.isDisplayed***())***;  
 Assert.*assertTrue****(***link.getText***()***.contains***(*" "*))***;  
 ***})***;  
***}***

**Прогоним тест.**

**Подведем итоги: Изучили ожидания явные и неявные, узнали о классе ExpectedConditions и его методах, узнали о том, что можно писать кастомные ожидалки переопределяя метод apply() интерфейса ExpectedCondition.**