## Работа с Selenium WebDriver

**Создание Xpath для поиска элемента на примере:**

Сайт: <https://music.yandex.ua/artist/12674/albums?sort=rating>

Альбом: The Day Is My Enemy

Нажать inspect in FirePath

Создали следующий правильный xpath: .//div[@class='page-artist']//div[@class='album album\_selectable']//img[@class='album-cover album-cover\_size\_L']

Div – тип элемента в дереве.

Class – атрибут элемента в дереве.

// - означает пропустить все предыдущее.

Сокращенный xpath: (.//img)[4]

[4] – всего четыре атрибута и этот четвертый.

(.//img)[@class = 'album-cover album-cover\_size\_L']

**Как найти родителя:** .//input[@id='lst-ib']//ancestor::\*

**Как найти родительский класс, содержащий букву "а":** .//input[@id='lst-ib']//ancestor::\*[contains(@class, "a")]

**Скрыть логотип яндекса на yandex.ua:** Ввести в консоли браузера: $(".home-logo\_\_default").hide("slow"); Для обратного показа логотипа hide необходимо заменить на show.

## Nunit Framework

Учет погрешности:

double expected = 2;

double actual = 1.999;

double delta = 0.001;

Assert.AreEqual(expected, actual, delta, "Погрешность больше 0.001");

Проверка равенства ссылок:

string input = "HELLO";

string expected = "HELLO";

Assert.AreSame(expected, input);

При этом если мы проверяем не ссылочный тип данных (например, int), то с помощью этого метода мы никогда не получим true.

Проверка, что в коллекции нет нулевых элементов:

List<string> employees = new List<string>();

CollectionAssert.AllItemsAreNotNull(employees, "Not null failed");

Проверка, что все элементы коллекции уникальные:

List<string> employees = new List<string>();

CollectionAssert.AllItemsAreUnique(employees, "Uniqueness failed");

Проверка равенства коллекций:

List<string> employees = new List<string>();

List<string> emp = new List<string>();

CollectionAssert.AreEqual(employees, emp);

Проверка равенства коллекций на значения, а не последовательность:

List<string> employees = new List<string>();

List<string> emp = new List<string>();

CollectionAssert.AreEquivalent(employees, emp);

Проверка, что элементы одной коллекции входят в другую коллекцию:

List<string> employees = new List<string>();

List<string> employees\_Subset = new List<string>();

CollectionAssert.AreEquivalent(employees\_Subset, employees, "failed!");

Проверка, что в одной строке содержится другая строка:

StringAssert.Contains("Assert samples", "sam");

Проверка, что в строку входят три цифры подряд:

StringAssert.IsMatch("123", new Regex(@"\d{3}"));

Виды тестовых объектов:

Dummy object – передается в тестируемый класс в качестве параметра. Не имеет поведения и с ним ничего не происходит (например, пустой объект класса или переменная).

Test stub – используется для получения данных от внешних зависимостей, подменяя их.

Test spy – используется для тестов взаимодействия. Основная задача – запись данных, которые поступают от тестируемого объекта.

Mock object – похож на test spy, но не записывает данные от тестируемого объекта, а самостоятельно проверяет их.

Fake object – замена тяжеловесной внешней зависимости легковесной реализацией. Эмулятор БД в памяти, фальшивый веб-сервис.

Inversion of Control (инверсия управления) – это важный абстрактный принцип, представляющий собой набор рекомендаций для написания слабо связанного кода, суть которого заключается в том, что каждый компонент системы должен быть как можно более изолированным от других и не полагаться в своей работе на детали конкретной реализации других компонентов.

Известные реализации принципа Inversion of Control:

* Dependency Injection.
* Service Locator.
* Factory Method.

Service Locator – паттерн, использующийся для отделения объекта от его зависимостей.

Dependency Injection – паттерн, описывающий технику внедрения внешней зависимости в программный компонент.

Способы внедрения зависимостей:

* через конструктор,
* через свойство,
* через интерфейс.

Способы создания экземпляров зависимостей: через контейнеры (Unity, Ninject, Spring.NET).

Модульные тесты можно условно разделить на две группы.

Тесты состояния (state-based tests) – тесты, проверяющие корректность отработки вызываемого метода. При этом проверяется состояние тестируемого объекта после вызова метода. State based tests базируются на Stub-объектах.

Тесты взаимодействия (interaction tests) – тесты, в которых тестируемый объект производит манипуляции с другими объектами. Они применяются, когда требуется удостовериться, что тестируемый объект корректно взаимодействует с другими объектами. Interaction tests базируются на Mock-объектах.

Mock-объект – управляемая замена существующих зависимостей в системе. Mock-объекты заменяют реальные объекты системы и позволяют проверить вызовы своих членов тестируемым классом. Mock-объекты отличаются от Stub-объектов тем, что они могут быть причиной неуспешного завершения юнит-теста.

Isolation Framework – инструмент, автоматизирующий процесс создания заглушек для классов, интерфейсов методов, используемых в тестируемом методе.

* Rhino Mocks.
* Moq.
* Typemock Isolator.
* Microsoft Fakes.
* NMock2.