**Инструкция 2**

***Рефакторизация кода***

В результате выполнения первого задания, у вас в проекте должен был оказаться всего один класс с тестом.

Давайте немного структурируем. Предположим, наш тест сейчас выглядит вот так:

[Test]

public void TheUntitledTest()

{

driver.Navigate().GoToUrl(baseURL + "addressbook/");

driver.FindElement(By.Name("user")).Clear();

driver.FindElement(By.Name("user")).SendKeys("admin");

driver.FindElement(By.Name("pass")).Clear();

driver.FindElement(By.Name("pass")).SendKeys("secret");

driver.FindElement(By.CssSelector("input[type=\"submit\"]")).Click();

driver.FindElement(By.LinkText("groups")).Click();

driver.FindElement(By.XPath("(//input[@name='new'])[2]")).Click();

driver.FindElement(By.Name("group\_name")).Clear();

driver.FindElement(By.Name("group\_name")).SendKeys("name");

driver.FindElement(By.Name("group\_header")).Clear();

driver.FindElement(By.Name("group\_header")).SendKeys("header");

driver.FindElement(By.Name("group\_footer")).Clear();

driver.FindElement(By.Name("group\_footer")).SendKeys("footer");

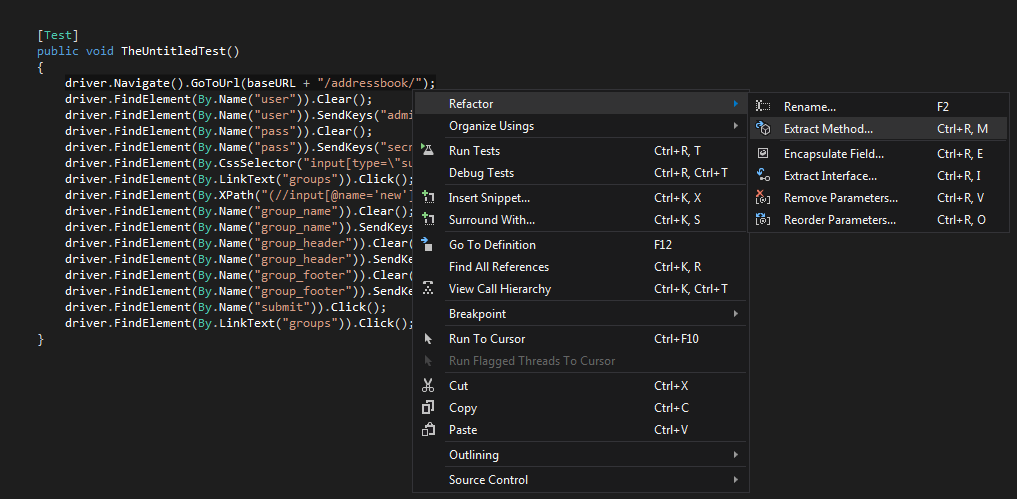
driver.FindElement(By.Name("submit")).Click();

driver.FindElement(By.LinkText("groups")).Click();

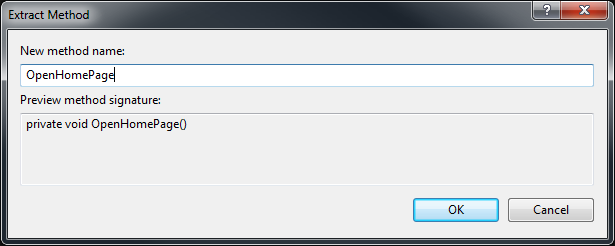
}

Нам нужно сделать его более читабельным.

Первая строка производит переход на домашнюю страницу. Давайте выделим его в отдельный метод. Для этого выделим его, кликнем правой кнопкой мыши и выберем Refactor -> Extract method.



Дадим ему понятное название



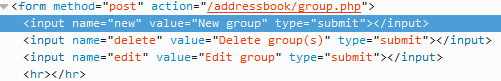
Что же мы получим? В самом тесте шаг выглядит почти как на естественном языке, а само действие выполняется во вспомогательном методе. Сделаем то же самое с остальными шагами.



Что же мы получили? «Читая» метод самого теста, мы сразу понимаем что происходит. При этом, обратите внимание, перехода на страницу групп у нас 2. Выделив его в отдельный метод, мы убрали дублирование. Например, если локатор ссылки на эту страницу изменится, в коде его нам нужно менять лишь в одном месте.

*Что такое локатор? Это способ уникально идентифицировать элемент на странице, чтобы драйвер не запутался и сделал то, что нам нужно.*

*Например, возьмем кнопку «New group» на странице со списком групп. Кликаем правой кнопкой и выбираем Inspect Element. Что мы видим?*

**

*Теперь давайте искать способы кликнуть по ней.*

*Мы можем обратиться по имени: driver.FindElement(By.Name("new")).Click();*

*По CssSelector так driver.FindElement(By.CssSelector("input[name='new']")).Click(); или так driver.FindElement(By.CssSelector("input[value='New group']")).Click();*

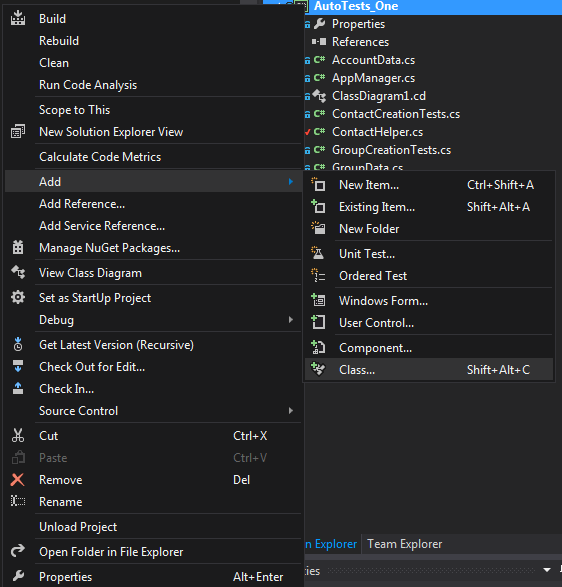
*По XPath так driver.FindElement(By.XPath("(//input[@name='new'])")).Click(); или так driver.FindElement(By.XPath("(//input[@value='New group'])")).Click();*

Что же мы имеем теперь? 1 класс, содержащий и тест, и методы инициализации, и вспомогательные методы.

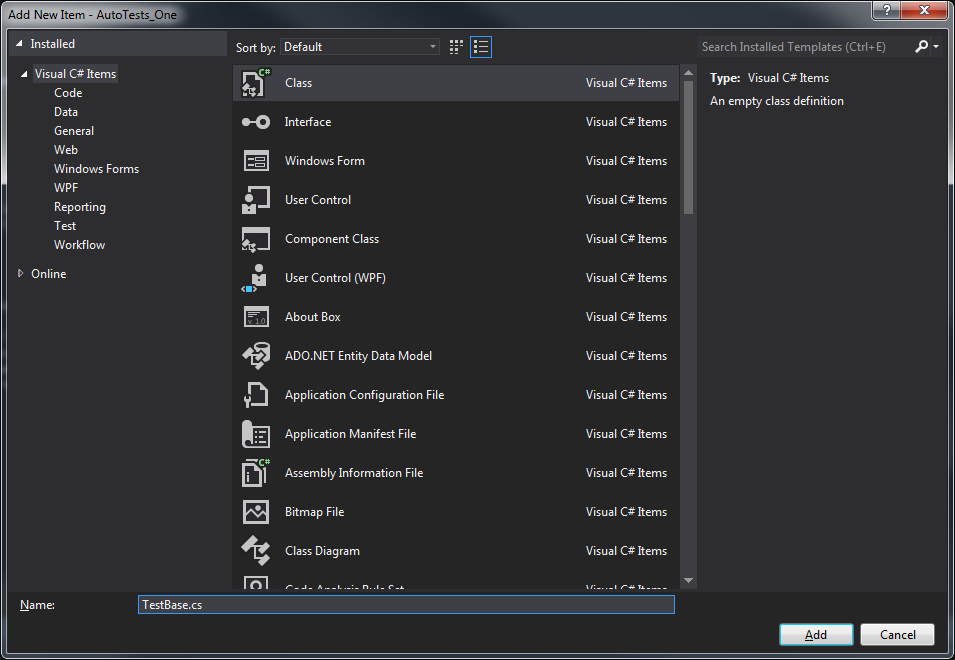
***Создание базового для всех тестов класса***

Давайте еще немного структурируем. Добавим базовый класс для всех тестов.

В Solution Explorer правый клик по имени проекта, Add -> Class.



Назовем его TestBase.



Теперь 3 обязательных действия, которые нужно проделать при создании нового класса сейчас и в будущем:

1. Скопировать список библиотек
2. namespace везде должен совпадать. У меня namespace AutoTests\_One (замените на свой)
3. Видимость класса сделать public public class TestBase

Теперь укажем TestBase родителем нашего класса с тестом

[TestFixture]

public class GroupCreationTests : TestBase

И перенесем в TestBase все, что не является с тестом. Вырезаем из класса с тестом все, что не помечено [Test] и вставляем внутрь TestBase.

Если у вас что-то подчеркивается красным в тесте, значит, тест не видит вспомогательные методы. Почему? Потому что они в другом классе и видимость у них private. Меняем ее в TestBase на public для всех вспомогательных методов:

private void Login() -> public void Login()

***Создание классов для объектов сайта.***

На этом наша работа не закончена. Нужно создать классы для объектов нашего сайта. Например, у нас есть пользователь с логином и паролем. Создаем для него класс AccountData (тут библиотеки копировать необязательно, нужен одинаковый namespace и видимость public).

Задаем ему 2 поля: логин и пароль. Выглядеть будет примерно так:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AutoTests\_One

{

public class AccountData

{

public AccountData(string username, string password)

{

Username = username;

Password = password;

}

public string Username {get; set;}

public string Password { get; set; }

}

}

Создаем класс для объекта группы. Назовем GroupData. Перечисляем все поля, которые есть на форме создания новой группы.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AutoTests\_One

{

public class GroupData

{

public GroupData(string name)

{

Name = name;

}

public string Name { get; set; }

public string Header { get; set;}

public string Footer { get; set; }

}

}

Почему Header и Footer я написала только как свойство внизу? Потому что они будут считаться необязательными. Обязателен только Name.

Теперь давайте используем наши объекты сайта. В тесте перед вызовом метода добавляем

AccountData user = new AccountData("admin", "secret");

Передаем user как параметр нашему методу авторизации

Login(user);

В сам метод авторизации добавляем параметр

public void Login(AccountData user)

{

driver.FindElement(By.Name("user")).Clear();

driver.FindElement(By.Name("user")).SendKeys(user.Username);

driver.FindElement(By.Name("pass")).Clear();

driver.FindElement(By.Name("pass")).SendKeys(user.Password);

driver.FindElement(By.CssSelector("input[type=\"submit\"]")).Click();

}

Таким же образом поступаем с группой

Добавляем в тест GroupData group = new GroupData("New Group") { Header = "sds", Footer = "dsfsd"}; , в метод создания группы добавляем параметр CreateNewGroup(group); , и в сам метод добавляем этот параметр.

public void CreateNewGroup(GroupData group)

{

driver.FindElement(By.Name("new")).Click();

FillTheField("group\_name", group.Name);

if (group.Header != null)

{

FillTheField("group\_header", group.Header);

}

if (group.Footer != null)

{

FillTheField("group\_footer", group.Footer);

}

driver.FindElement(By.Name("submit")).Click();

}

Дополнительно я добавила проверки на наличие Header и Footer в данных для заполнения, потому что они у нас необязательны.

**Домашнее задание:**

Повторить то, что я перечислила выше:

1. Отрефакторить код
2. Добавить базовый класс для тестов
3. Добавить классы для объектов сайта (если имеется форма с полями, с которыми вы в тесте взаимодействуете, то для этой формы нужно создать класс …Data, достаточно указать только те поля, которые вы используете, остальные можно добавить при необходимости) – AccountData, GroupData, ContactData
4. Добавить новый тест на добавление какой-либо сущности на сайт(зависит от сайта, может быть группа, заметка, сообщение и тд), то есть снова записать рекордером и снова проделать все пункты выше аналогично для нового теста.   
   (В итоге получится два теста, дважды [Test])