**Инструкция 3**

Мы остановились на том, что создали отдельные классы с тестами, которые наследуются от TestBase. Тесты ничего не знают про драйвер, они всего лишь вызывают вспомогательные методы, находящиеся в TestBase. Дополнительно к этому мы создали классы для сущностей, которые нам нужны: AccountData, ContactData, GroupData.

Давайте теперь немного структурируем.

ВНИМАНИЕ! С этого момента у вас какое-то время не будут запускаться тесты, пока мы не закончим строить нашу архитектуру. Не пугайтесь, это нормально. Все заработает, когда мы закончим.

Начнем с перемещений по страницам. У нас есть несколько методов в TestBase, которые предназначены только для переходов по страницам (например, OpenHomePage, OpenGroupsPage и т.д.)

Создадим новый класс NavigationHelper.

Не забывайте про обязательные вещи при создании нового класса:

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

Перенесем в него методы, связанные с навигацией (вырезать из TestBase, вставить в NavigationHelper).

Создадим новый класс LoginHelper и перенесем в него все, что связано с авторизацией (метод Login и Logout, если имеется).

Создадим новый класс GroupHelper и перенесем в него все, что связано с группами.

Создадим новый класс ContactHelper и перенесем в него все, что связано с контактами.

На данный момент в TestBase у вас должны остаться только автоматически сгенерированные методы: SetupTest(),TeardownTest(),CloseAlertAndGetItsText(),IsAlertPresent(),IsElementPresent(By by)

Разберемся с ними.

У тестов есть свой общий родительский класс TestBase.

Пусть и у классов-помощников будет свой родительский класс – HelperBase. Обратим внимание, что все методы в хелперах используют драйвер, поэтому объявим его как поле, чтобы в дальнейшем нигде его не объявлять:

protected IWebDriver driver;

Создадим его и укажем родителем у всех Helper’ов:

public class ContactHelper : HelperBase

Перенесем туда protected bool acceptNextAlert = true;

А также автоматически созданные методы CloseAlertAndGetItsText(),IsAlertPresent(),IsElementPresent(By by)

Теперь объединим все это. Нам поможет ApplicationManager – класс, который будет всем управлять.

1. Создадим класс ApplicationManager и проведем стандартные для новых классов манипуляции
2. Из TestBase перенесем

private IWebDriver driver;

private StringBuilder verificationErrors;

private string baseURL;

1. Объявим наши классы помощники

private NavigationHelper navigation;

private LoginHelper auth;

private GroupHelper group;

private ContactHelper contact;

1. Теперь создадим конструктор нашего менеджера, вырезав команды из SetUp в TestBase, и инициализируем наших помощников

public AppManager()

{

driver = new ChromeDriver(@"D:\distr");

driver.Manage().Window.Maximize();

baseURL = "http://localhost";

verificationErrors = new StringBuilder();

group = new GroupHelper(this);

contact = new ContactHelper(this);

auth = new LoginHelper(this);

navigation = new NavigationHelper(this, baseURL);

}

1. Добавим нашему менеджеру Property

public IWebDriver Driver

{

get

{

return driver;

}

}

public NavigationHelper Navigation

{

get

{

return navigation;

}

}

public LoginHelper Auth

{

get

{

return auth;

}

}

public GroupHelper Group

{

get

{

return group;

}

}

public ContactHelper Contact

{

get

{

return contact;

}

}

Эти Property позволят тестам обращаться к классам-помощникам через менеджера.

1. На шаге 4 мы инициализировали наших помощников и говорили им про менеджера в this (group = new GroupHelper(this);). Сделаем связь двусторонней. В классах-помощниках добавим конструкторы:

В GroupHelper поместим

public GroupHelper(AppManager manager)

: base(manager)

{

}

В ContactHelper

public ContactHelper(AppManager manager)

: base(manager)

{

}

В LoginHelper

public LoginHelper(AppManager manager)

: base(manager)

{

}

В NavigationHelper (он от менеджера получает еще и baseURL, который ему необходим для выполнения своего прямого назначения)

private string baseURL;

public NavigationHelper(AppManager manager, string baseURL)

: base(manager)

{

this.baseURL = baseURL;

}

1. Вроде бы всех познакомили друг с другом? Не совсем. Наши базовые классы еще не подозревают о существовании менеджера. Сообщим же HelperBase

protected AppManager manager;

protected IWebDriver driver;

public HelperBase(AppManager manager)

{

this.manager = manager;

this.driver = manager.Driver;

}

И TestBase (из него удаляем все, что уже имеется в Application Manager, посмотрите выше)

protected AppManager app;

[SetUp]

public void SetupTest()

{

app = new AppManager();

}

Благодаря тому, что здесь объявляется AppManager, во всех тестах не нужно прописывать объявление AppManager, т.к. все тесты наследуются от TestBase.

1. Поскольку запуском и работой тестов у нас заведует Application Manager, то завершение работы тоже перенесем в него. В TestBase в методе TeardownTest() содержится код, который закрывает браузер.

Создадим в ApplicationManager метод

public void Stop()

{

driver.Quit();

}

А в TestBase просто вызовем его

[TearDown]

public void TeardownTest()

{

app.Stop();

}

Вроде бы все готово. Но нннет. Тесты у нас совсем сломались и красные-красные. Почему? А потому что они не могут найти вспомогательные методы, которые перенесены в классы-помощники.

Так давайте сообщим тестам новое расположение (TestBase уже сказали на шаге 8 ;) )

[Test]

public void GroupCreationTest()

{

app.Navigation.GoToGroupsPage();

GroupData group = new GroupData("New Group") { Header = "sds", Footer = "dsfsd"};

app.Group.CreateNewGroup(group);

app.Navigation.GoToGroupsPage();

}

И так же поступим с остальными тестами.

