Automated Business Readable Web Tests with Selenium and SpecFlow

<https://app.pluralsight.com/library/courses/selenium-specflow-automated-business-readable-web-tests/table-of-contents>

Also see Selenium Web Driver.docx in ..\dev\training foloder

# Introducing Selenium

## Selenium

Automates browsers

Simulates human interaction in the UI

## IDE

Record automation test scripts

Playback scripts

Edit scripts

## WebDriver

Code-based

More maintainable

Full power of programming languages

# SpecFlow

Bridges the gap between devs and business users.

Write tests that are easily understood.

Generally non-technical.

Ensure different scenarios are covered.

## Example

Given I’m on a the home page 🡪 WebDrive API will go ..\home

When I choose more info 🡪 Clicks the “more info” link

Then it should take me to a new page 🡪 WebDriver gets the browser curr URL, and run assert

# Module: Getting Started with Selenium

## Overview of WebDriver

public interfact ISearchContext{

IWebElement **FindElement**(**By** by);

ReadOnlyCollection<IWebElement> FindElements(By by);

}

public class **By** {

// static methods to locate elements on the page

}

## Html Element Section

The By class is used to locate elements by:

Id, Name, ClassName, XPath, LinkText, PartialLinkText, TagName, CssSelector

## Creating the first test

**Open Loan sln file** in ..\pluralsight\selenium-specflow-tests\2-exercise-files\after\GettingStarted\

***packages.config***

Selenium.Webdriver, xunit and xunit.runner packages already installed via NuGet.

Create new LoanApplicationTest class in test project DemoWebApp.Tests

***FYI***: **“Test Explorer” window** can be found under menu Tests-Windows-TestExplorer

Default Port: rt-click on Loan project and go to Properties, then Web

<http://localhost:40077/>

# Module: Adding Business Readability with SpecFlow

## Business Readable UI Automation

* Document the features of the system we’re building
* Documents written in a non-technical manner
* It may help in internal/external auditing when it comes to non-tech language
* Executable as tests
* The buns readable tests are source controller, alongside the test code
* The documentation stays up-to-date; tests will fail if the code does not match the docs
* Helps to create better communication between developers and stakeholders
* Rreduces wasted effort on features the business doesn’t want

## Installing SpecFlow Visual Studio Extension

Manage NuGet packages, search for “SpecFlow.xUnit” package

Also install ***ChromeDriver*** NuGet package, and use it in ***LoanApplicationSteps.cs***

# Module: Creating More Maintainable Web Automation

## Page Object Models

Improve readability by using private getters/setters as follows :

Change:

IWebElement firstNameInput = \_driver.FindElement(By.Id("FirstName"));

firstNameInput.SendKeys(firstName);

To:

\_loanApplicationPage.FirstName = firstName;

\*\*\* See LoadApplicationPage.cs for private vars

## Creating a Loan Application Object Model

LoadApplicationPage.cs sets a bunch of private vars.

FYI: LoanApplicationTests.cs was the old way of testing, prior to refactoring with the page object model.

Ex/

private IWebElement \_firstName;

private IWebElement \_lastName;

public string FirstName

{

set

{

\_firstName.SendKeys(value);

}

}

public string SecondName

{

set

{

\_secondName.SendKeys(value);

}

## Refactoring To Use Page Object Models

In LoanApplicationSteps.cs, change GivenIAmOnTheLoanApplicationScreen()

\_driver.Navigate().GoToUrl() is now replaced with:

**LoanApplicationPage.NavigateTo**(\_driver)

Entering the first name of “Sarah”

[Given(@"I enter a first name of (.\*)")]

public void GivenIEnterAFirstNameOf(string firstName)

{

//IWebElement firstNameInput = \_driver.FindElement(By.Id("FirstName"));

//firstNameInput.SendKeys(firstName);

\_loanApplicationPage.FirstName = firstName;

}

[Then(@"I should see the application complete confirmation for Sarah")]

public void ThenIShouldSeeTheApplicationCompleteConfirmationForSarah()

{

//IWebElement confirmationNameSpan = \_driver.FindElement(By.Id("firstName"));

//string confirmationName = confirmationNameSpan.Text;

Assert.Equal("Sarah", \_confirmationPage.FirstName);

}

***FYI***: See **LoanApplication.feature.cs** for code generated by SpecFlow

i.e. testRunner.Then("I should see the application complete confirmation for ***Sarah***"

## Install Selenium.Support package

Install from NuGet

In ApplicationConfirmationPage.cs

[FindsBy(How = How.Id, Using = "firstName")]