

```
3 require File.expand_path("../config/initializers/spec_helper.rb", __FILE__)
4 # Prevent database truncation if the environment is production
5 abort("The Rails environment is running in production mode!")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
```

Specflow & Selenium for Developers

20.06.2018

Share the passion

SPECFLOW & SELENIUM

Ziel und Zweck

Akzeptanztests sind zentraler Bestandteil der Definition of Done und sind in der Summe End-to-End Tests.

Dabei sind im PBI die **Akzeptanzkriterien** im **Gherkin** Syntax definiert, aus welchem nun durch die Entwickler die entsprechenden Tests generiert werden.

Diese Tests werden **automatisiert ausgeführt** und sobald alle Tests erfolgreich durchlaufen, sind die Akzeptanzkriterien erfüllt und das PBI **abgenommen**. Zusätzlich laufen diese Tests danach als **Regressionstests** bei **jedem Build** oder anhand von Tags auch nach einem **bestimmten Zeitplan** durch.



SPECFLOW & SELENIUM

Vorbereitung

In der Dateiablage der Testgilde unter *Dokumente/Workshop Unterlagen/* das Dokument "Work with SpecFlow and Selenium" öffnen und die Vorbereitungen installieren.



Specflow

Specflow verarbeitet die Gherkin Feature Dateien und generiert daraus den sogenannten Glue-Code. Dies sind einzelnen Testschritte, welche dann durch den xUnit Testrunner ausgeführt und geprüft werden.

When I go to the Info Hub



```
[When(@"I go to the Info Hub")]
public void WhenIGoToTheInfoHub()
{
    ScrollDown();
}
```

MyGherkinCode.feature



xUnit

xUnit stellt einerseits den Testrunner zur Verfügung, welcher die Tests danach ausführt. Zusätzlich werden auch die Assertions damit ausgeführt, welche die erwarteten Resultate überprüfen.

```
[Then(@"I see the Side Bar")]
public void ThenISeeTheSideBar()
{
    Assert.True(IsSideBarVisible);
}
```

MyGherkinCode.feature.steps.ch



Selenium

Selenium ist die «Fernsteuerung» für die verschiedenen Browser. Damit werden die PageObjects und Methoden bereitgestellt, welche dann aus dem Glue-Code angesprochen werden.

```
public void ScrollDown()
{
    Driver.FindElement(By.TagName("body")) ...
}
```

MyPageOrSection.cs

SPECFLOW & SELENIUM

Easy Flow with Specflow



Feature Files

Die Feature Files werden durch den PO oder BA erstellt und in den Akzeptanz Kriterien des PBI verknüpft.

```
Scenario: Test with Chrome as default browser
    #Given is set by Background
    When I go to the Info Hub
    Then I see the Info Hub Page
    And The title contains Content Hub
```



Glue Code

Die **Step Definitions** (der Verbindungscode zwischen Feature File und Page Objects) wird von Specflow automatisch generiert.

```
[When(@"I go to the Info Hub")]
public void WhenIGoToTheInfoHub()
{
    Hooks.Driver.NavigateToPath("/de/hub.html");
}

[Then(@"The title contains Content Hub")]
public void ThenTheTitleContainsContentHub()
{
    Assert.Contains("Content Hub", _contentHub.title);
}
```



Hooks

Hooks sind **Step Definitions**, welche immer wieder gebraucht und an einer gewissen Position immer wieder ausgeführt werden sollen.

```
[BeforeTestRun]
[BeforeFeature]
[BeforeScenario]
[BeforeScenarioBlock]
[BeforeStep]
[AfterStep]
[AfterScenarioBlock]
[AfterScenario]
[AfterFeature]
[AfterTestRun]
```

Scenario Outline

Feature File

```
Scenario Outline: Open the Swiss Life homepage and check the title
#Given is set by Background
When I change the language to <lang>
Then The Homepage has the right <title>
Examples:
| lang | title|
| de   | Willkommen bei der Swiss Life-Gruppe|
| en   | Welcome to the Swiss Life Group|
```

Step Definitions

```
[When(@"I change the language to (.*)")]
public void WhenIChangeTheLanguageToDe(string lang)
{
    _homepage.ChangeLang(lang);
}

[Then(@"The Homepage has the right (.*)")]
public void ThenTheHomepageHasTheRight (string title)
{
    Assert.Equal(title, _homepage.HomepageTitle);
}
```

Scenario Info

Feature File

Scenario: Check German search tabs as example...

#Given is set by Background

When I enter the phrase Vorsorge

Then I see the following four tabs

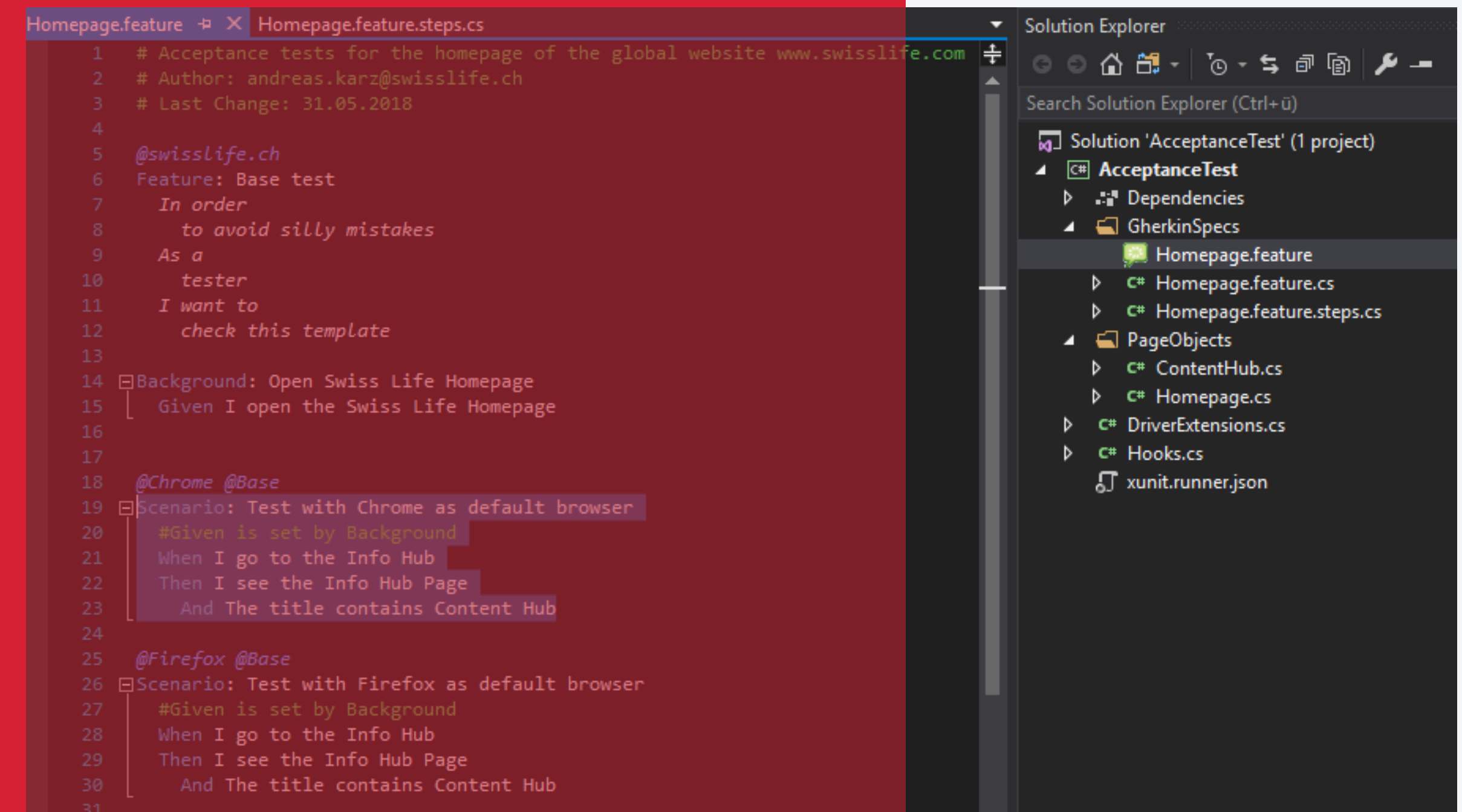
label	refinement	
Alle	all	
Seiten	pages	
Dokumente/Downloads	documents	
Medienmitteilungen	news	

Step Definitions

```
[Then(@"I see the following four tabs")]
public void ThenISeeTheFollowingFourTabs(Table table)
{
    IEnumerable<Tab> tabs = _searchPage.Tabs().CreateSet<Tab>();
    table.CompareToSet(tabs, false);
    Assert.True(true);
}
```

Unbedingt beachten!

- Gherkin immer in der gleichen Sprache
- Gleiche Schritte => gleicher Syntax (case-sensitive)
- Scenario Background in Hooks auslagern
- Neue Hooks dokumentieren für POs und BAs
- Syntax ist Feature File übergreifend
- Zahlen ohne 1000-er Trennzeichen und mit . als Dezimaltrennzeichen



The screenshot shows an IDE with two files open: `Homepage.feature` and `Homepage.feature.steps.cs`. The `Homepage.feature` file contains Gherkin syntax for acceptance tests for the Swiss Life homepage. The `Homepage.feature.steps.cs` file contains the corresponding C# implementation for the steps defined in the feature file. The Solution Explorer on the right shows the project structure, including the `AcceptanceTest` solution, `GherkinSpecs` folder, and the `Homepage.feature` file.

```

1  # Acceptance tests for the homepage of the global website www.swisslife.com
2  # Author: andreas.karz@swisslife.ch
3  # Last Change: 31.05.2018
4
5  @swisslife.ch
6  Feature: Base test
7    In order
8      to avoid silly mistakes
9    As a
10     tester
11     I want to
12     check this template
13
14  Background: Open Swiss Life Homepage
15  | Given I open the Swiss Life Homepage
16
17
18  @Chrome @Base
19  Scenario: Test with Chrome as default browser
20  | #Given is set by Background
21  | When I go to the Info Hub
22  | Then I see the Info Hub Page
23  | And The title contains Content Hub
24
25  @Firefox @Base
26  Scenario: Test with Firefox as default browser
27  | #Given is set by Background
28  | When I go to the Info Hub
29  | Then I see the Info Hub Page
30  | And The title contains Content Hub
31

```


Hooks

In den Hooks sind die Schritte definiert, welche immer wieder an der gleichen Position ausgeführt werden sollen.

Z.B. beim Start eines Szenarios **[BeforeScenario]** soll ein neuer Browser geöffnet werden. Am Ende eines jeden Szenarios **[AfterScenario]** soll dieser Browser natürlich wieder geschlossen werden. Damit mit verschiedenen Browsern getestet werden kann, kann dies über **Tags gesteuert** werden.

Ebenfalls soll nach jedem Szenario ein **Screenshot** erstellt und gespeichert werden.

SPECFLOW & SELENIUM

Asserts

Bei den *Then Steps* muss das erwartete Resultat geprüft werden. Dies wird mit *xUnit Asserts* geprüft.

Bei diesem Punkt entscheidet sich, ob ein Test erfolgreich ist oder nicht.

[XUNIT ASSERT](#)

SPECFLOW & SELENIUM

Pause



SPECFLOW & SELENIUM

Verify the behaviors

Selenium ist vereinfacht gesagt eine Fernsteuerung für gängige Browser.

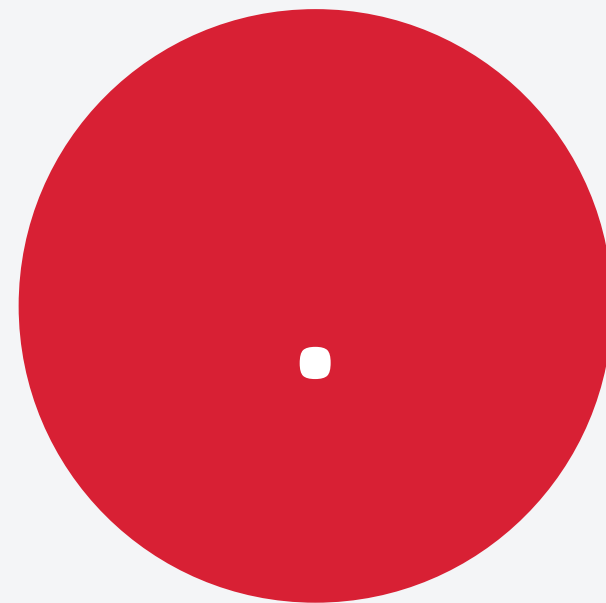
Kernkomponente ist der WebDriver, mit welcher die unterschiedlichen Browser angesprochen werden.

Diverse Erweiterungen vorhanden, z.B. für Screenshots oder Waiters für Komponenten.

Selektoren suchen Elemente – darauf lassen sich dann Aktionen ausführen oder Werte einlesen.

Selektoren arbeiten mit verschiedenen Filtern, z.B. CSS, XPath oder Tag Names.

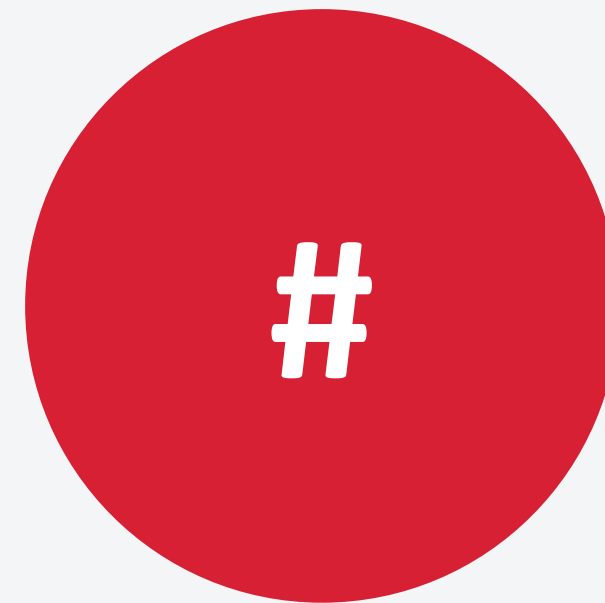




CSS

Mit dem CSS Selektor kann einerseits der Pfad aufgelöst werden (jQuery like) oder direkt nach dem Klassennamen gesucht werden.

```
By.CssSelector(".hidden-xsa[hreflang='de']")
By.ClassName("mod-teaser-home")
```



ID

Über die ID lassen sich Elemente direkt anhand der HTML ID eindeutig finden. Wird meist in Kombination mit dem CSS Selektor verwendet, d.h. das äussere Container Element wird über die ID geholt und dann in diesem Objekt via CSS die einzelnen Elemente gesucht.

```
By.Id("teaserSection")
```



Tag

Innerhalb eines Container Elements kann auch direkt nach dem Tag Namen gesucht werden. In diesem Fall sollte jedoch immer mit FindElements gesucht werden.

```
FindElements(By.TagName("button"))
```



xPath

Mit XPath steht sicher das mächtigste aber auch komplexeste Werkzeug zur Verfügung. Folgender Filter findet z.B. ein Inputfeld vom Typ Button mit dem Namen «continue».

```
By.XPath("//input[@name='continue'][@type='button']")
```

Das sind nur die wichtigsten Selektoren – eine komplette Liste gibt es hier: <https://www.guru99.com/locators-in-selenium-ide.html>

Der richtige Aufbau spart Zeit

Page

Formular

Element

Element

Element

Sektion

Titel

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Aenean commodo ligula eget
dolor. Aenean massa. Cum sociis
natoque penatibus et

Flexibel bleiben

```
<body>
  <div id="contact">
    <h1>Welcome</h1>
    <p>Der Weg zu uns</p>
    <a href="https://maps.google.com">Route</a>
    <form class="contactform">
      <input type="text" name="firstname" />
      <input type="text" name="lastname" />
      <input type="submit" value="Senden" />
    </form>
  </div>
</body>
```

```
var ContactSection = Driver.FindElement(By.Id("contact"));
var Title = ContactSection.FindElement(By.TagName("h1"));

var RouteLink = ContactSection.FindElement(By.TagName("a"));
var ContactForm = ContactSection.FindElement(By.TagName("form"));
var FirstName = ContactForm.FindElement(By.XPath("//input[@name='firstname']"));
var LastName = ContactForm.FindElement(By.XPath("//input[@name='lastname']"));
var Submit = ContactForm.FindElement(By.XPath("//input[@type='submit']"));
```

wait.Until – ExpectedConditions

Beim 1. Start der Seite oder nach einem Sprachwechsel etc. unbedingt mit *wait.Until* die initialen Elemente suchen, sonst sind die Elemente nicht ansprechbar.

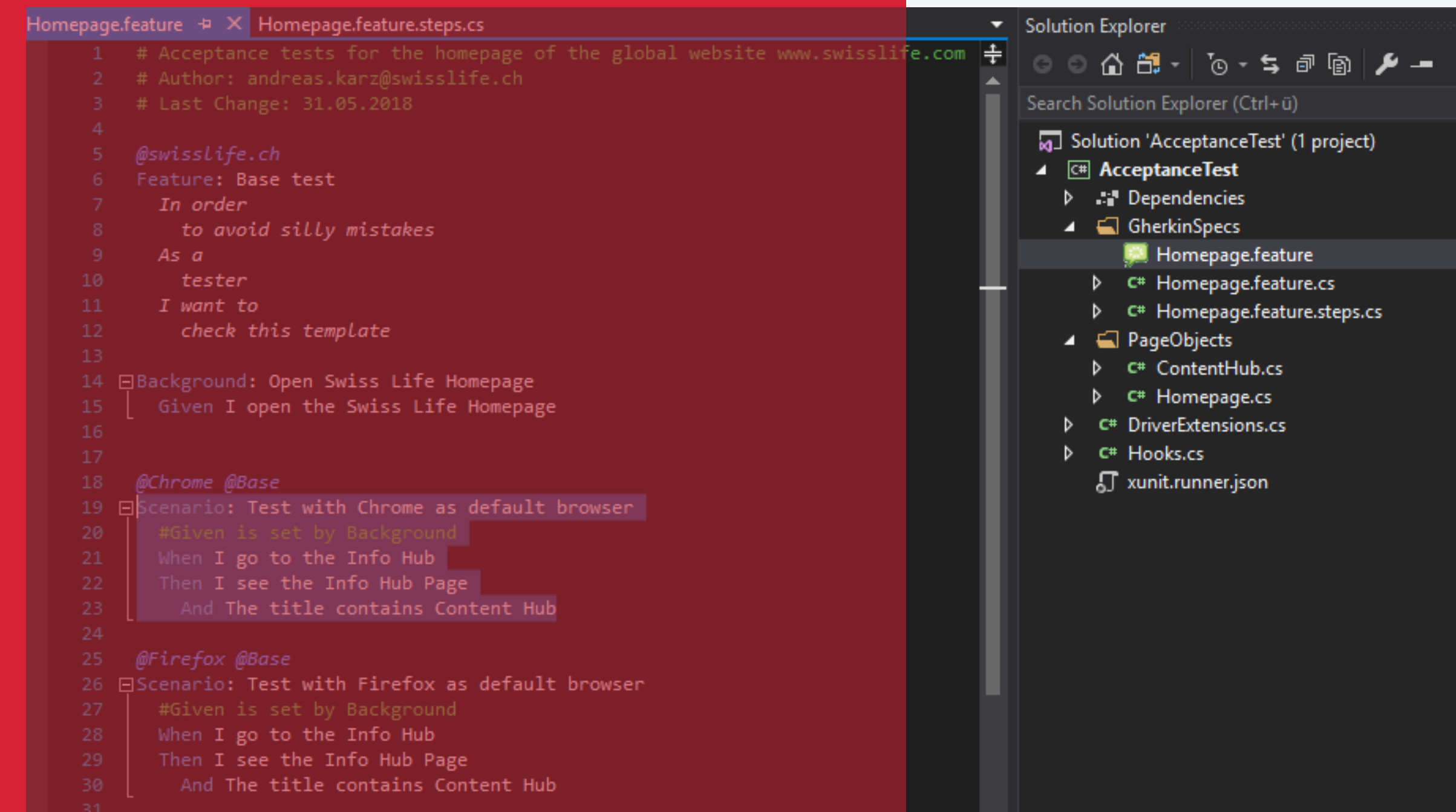
Dabei stehen verschiedene *ExpectedConditions* zur Verfügung – siehe [Github](#)

Für das einfache Handling im Alltag ist in den Swiss Life DriverExtensions die Methode *GetElementSafe* implementiert. Bei diesem kann als 2. Parameter optional ein Selektor mitgegeben werden, der nicht mehr sichtbar sein darf, bevor der 1. Selektor mit der Suche beginnt.

```
var MyElement = Driver.GetElementSafe(By.Id("contact"));  
var MyElement = Driver.GetElementSafe(By.Id("contact"), By.Id("AjaxSpinner"));
```

Unbedingt beachten!

- Nur Elemente initialisieren, welche immer wieder benötigt werden. Ein einfacher Klick, der nur einmal ausgeführt wird, kann direkt an *FindElement* angefügt werden.
- Elemente erst initialisieren, wenn endgültige Seite geladen ist. Das ist vor allem bei Sprachwechseln zu beachten, da sonst die Elemente *stale* sind.
- Dynamisch nachgeladene Elemente immer mit *wait.until* suchen, da diese nicht unmittelbar vorhanden sind.



The screenshot shows an IDE with two files open: `Homepage.feature` and `Homepage.feature.steps.cs`. The `Homepage.feature` file contains Gherkin syntax for acceptance tests. The `Homepage.feature.steps.cs` file contains the corresponding C# implementation for these steps.

```

Homepage.feature
1 # Acceptance tests for the homepage of the global website www.swisslife.com
2 # Author: andreas.karz@swisslife.ch
3 # Last Change: 31.05.2018
4
5 @swisslife.ch
6 Feature: Base test
7   In order
8     to avoid silly mistakes
9   As a
10    tester
11    I want to
12    check this template
13
14 Background: Open Swiss Life Homepage
15 | Given I open the Swiss Life Homepage
16
17
18 @Chrome @Base
19 Scenario: Test with Chrome as default browser
20 | #Given is set by Background
21 | When I go to the Info Hub
22 | Then I see the Info Hub Page
23 | And The title contains Content Hub
24
25 @Firefox @Base
26 Scenario: Test with Firefox as default browser
27 | #Given is set by Background
28 | When I go to the Info Hub
29 | Then I see the Info Hub Page
30 | And The title contains Content Hub
31

```

The Solution Explorer on the right shows the project structure for 'AcceptanceTest' (1 project). It includes folders for 'Dependencies' and 'GherkinSpecs'. Under 'GherkinSpecs', there is a folder 'Homepage.feature' containing files: `Homepage.feature.cs`, `Homepage.feature.steps.cs`, `PageObjects`, `ContentHub.cs`, `Homepage.cs`, `DriverExtensions.cs`, `Hooks.cs`, and `xunit.runner.json`.

Getter & Action

?

Getter

Von den gefunden Elementen kann man diverse Eigenschaften abfragen:

- Text
- HTML-Attribute
- CSS Value
- Type
- Selected
- Enabled
- Visible
- Tag Name
- Size

```
public string HomepageTitle => _homepageTitle.Text;
```

!

Actions

Auf den gefunden Elementen können folgende Methoden ausgeführt werden:

- Click()
- Submit()
- Clear()
- SendKeys()

```
_langSelector["de"].Click();  
_firstName.SendKeys("Andreas");
```

SL Driver Extensions (Helpers)

NavigateToPath

Navigiert zu einem relativen Link. Als optionaler zweiter Parameter kann ein Selektor mit gegeben werden. Dann wird ein neues IWebElement zurück gegeben, sobald es der Selektor gefunden hat.

LocalStorage

Für die Manipulation des Local Storage stehen die folgenden Methoden zur Verfügung:

- LocalStorageSetItem
- LocalStorageGetItem
- LocalStorageClear
- LocalStorageLength

IsElementPresent

Gibt einen Boolean zurück, ob das gewünschte Element vorhanden ist.

Element kann dabei auch unsichtbar oder verdeckt sein.

GetElementSafe

Liefert das gewünschte IWebElement zurück, sobald dieses vorhanden, sichtbar und anklickbar ist.

Als optionaler 2. Selektor kann das Element angegeben werden, welches vorher ausgeblendet sein muss, z.B. ein Ajax Spinner.

ExecuteScript

Führt einen JavaScript Befehl aus.

MakeScreenshot

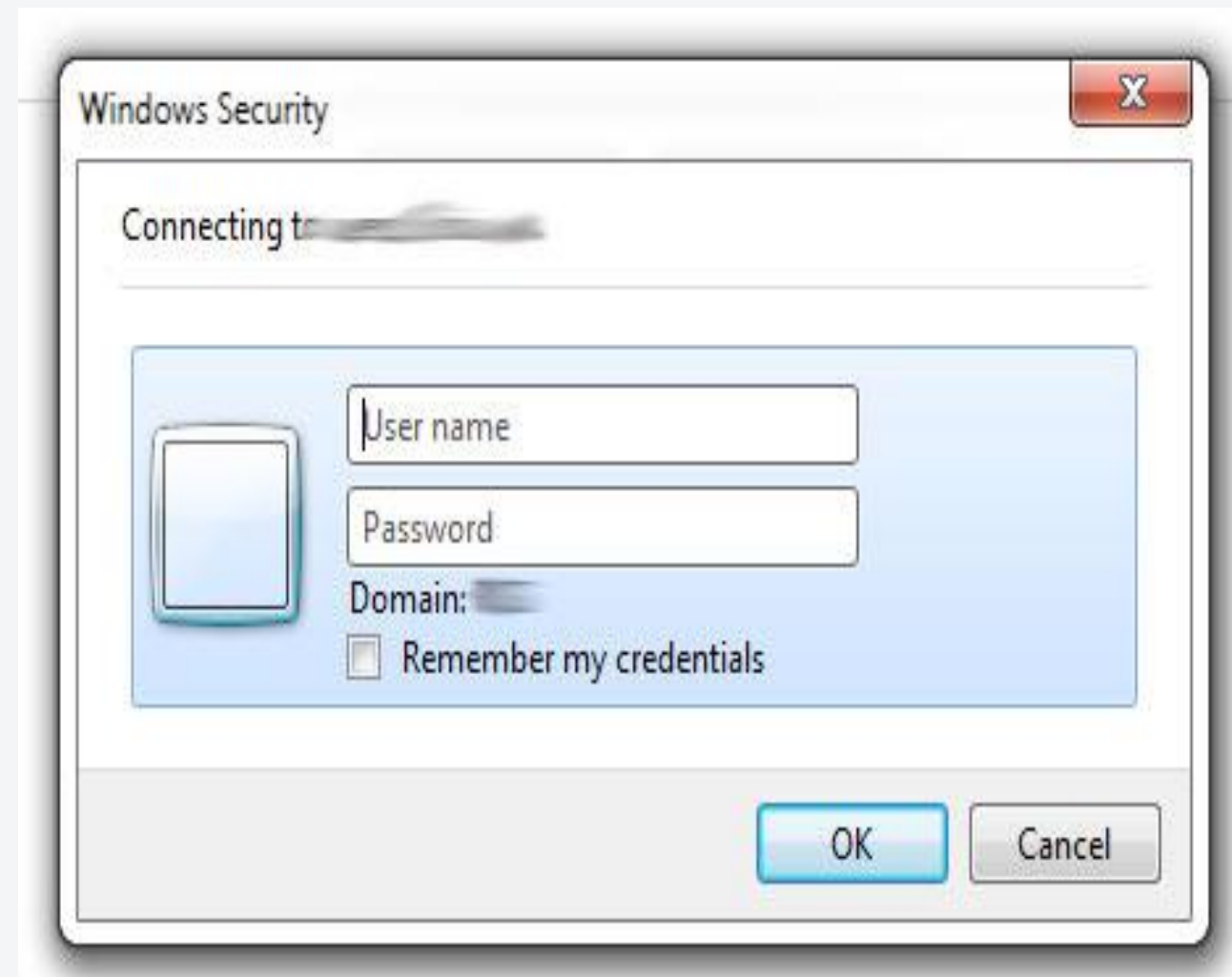
Erstellt einen Screenshot des Browser, ergänzt den Dateinamen mit der Versionsnummer und speichert das Bild in das Verzeichnis *SeleniumResults*.

SetMobileSize

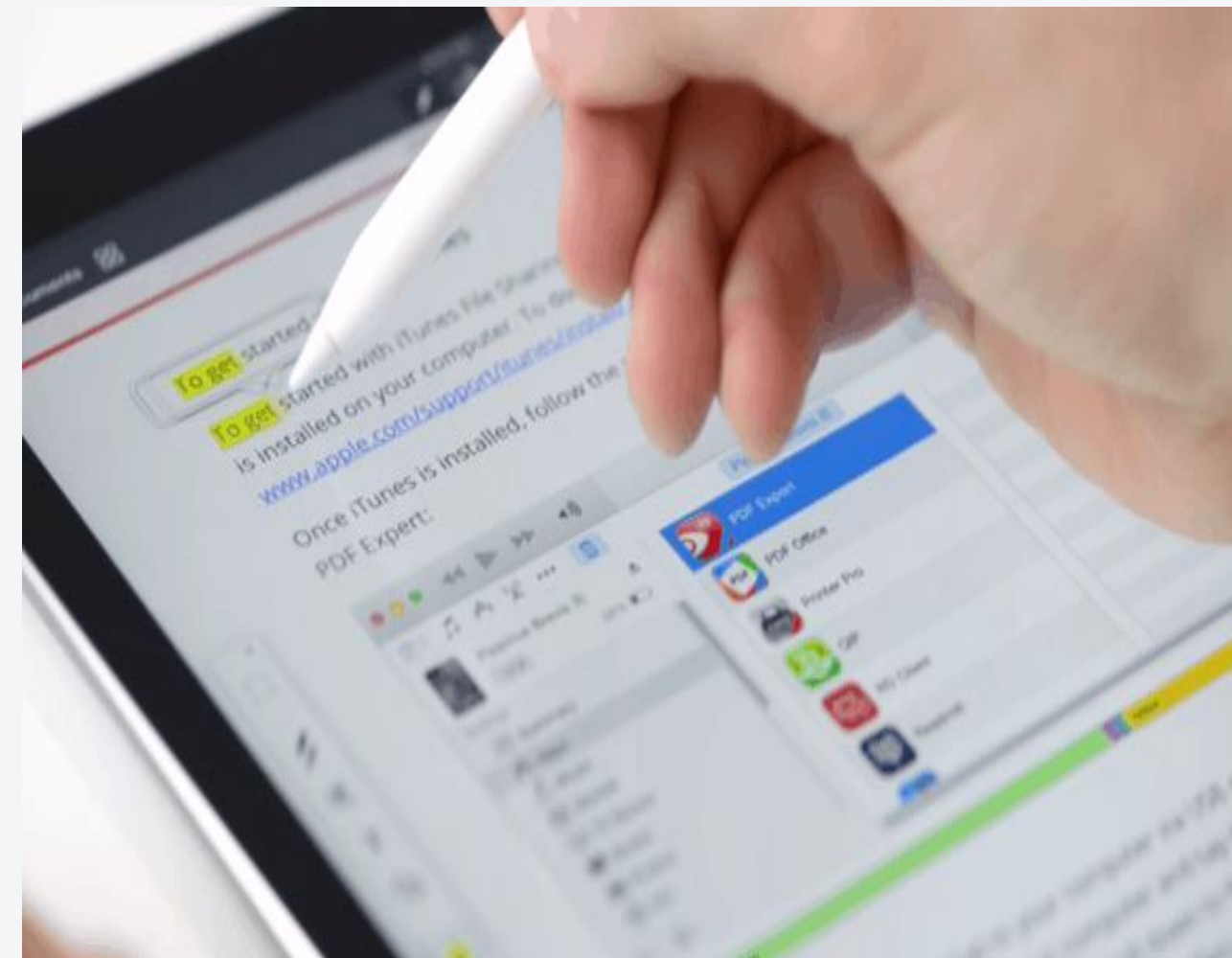
Setzt die Grösse des Browsers auf 750px * 1024px. Die Pixelgrösse kann auch optional angegeben werden.

SELENIUM & SPECFLOW

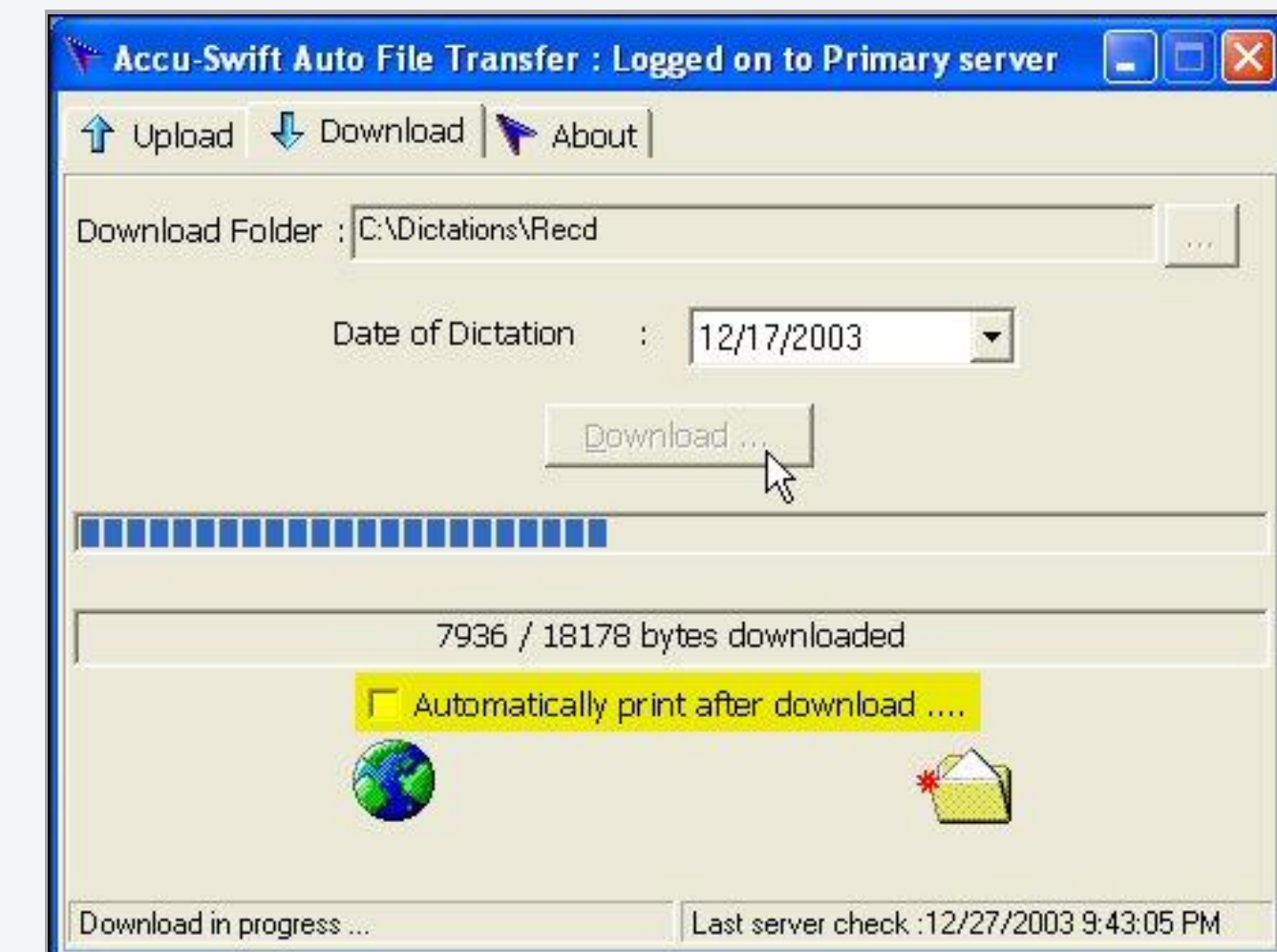
Trickshots



Window Authentication
und Pop-Ups allgemein



PDF prüfen



Downloads validieren

SELENIUM & SPECFLOW

FAQ

Sind
Selenium
und Specflow
nur für C#
verfügbar?

Gibt es
Support bei
der
Umsetzung?

Wie werden die
Tests
automatisiert?

?

TEST GILDE

Microsoft Yammer



SPECFLOW & SELENIUM

Herzlichen Dank
