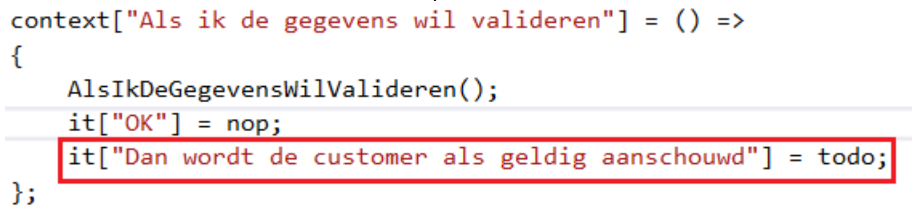
How To's

Abbreviations

- RMC: Right mouse click
- Spec: Specification

Mark steps as pending

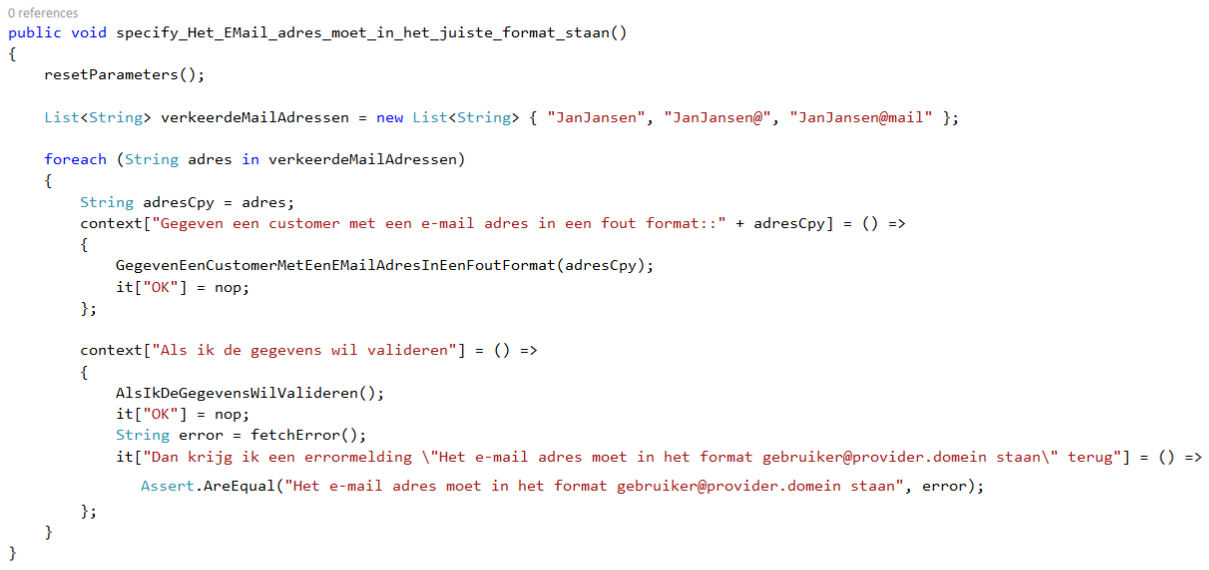
1) Navigate to the step that should be pending
2) Write "todo" after the lambda expression



```
context["Als ik de gegevens wil valideren"] = () =>
{
    AlsIkDeGegevensWilValideren();
    it["OK"] = nop;
    it["Dan wordt de customer als geldig aanschouwd"] = todo;
};
```

Create a cyclic(data driven) test

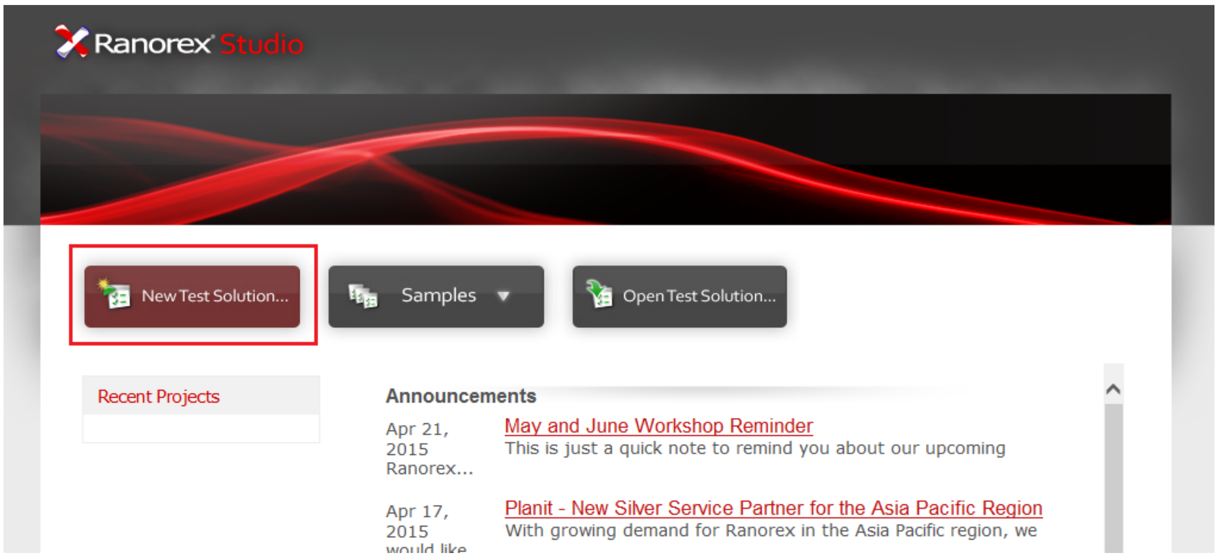
1) Write a scenario as described in the manual
2) Write your steps
3) Build a loop around the steps



```
0 references
public void specify_Het_EMail_adres_moet_in_het_juiste_format_staan()
{
    resetParameters();

    List<String> verkeerdeMailAdressen = new List<String> { "JanJansen", "JanJansen@", "JanJansen@mail" };

    foreach (String adres in verkeerdeMailAdressen)
    {
        String adresCpy = adres;
        context["Gegeven een customer met een e-mail adres in een fout format::" + adresCpy] = () =>
        {
            GegevenEenCustomerMetEenEMailAdresInEenFoutFormat(adresCpy);
            it["OK"] = nop;
        };

        context["Als ik de gegevens wil valideren"] = () =>
        {
            AlsIkDeGegevensWilValideren();
            it["OK"] = nop;
            String error = fetchError();
            it["Dan krijg ik een errormelding \"Het e-mail adres moet in het format gebruiker@provider.domein staan\" terug"] = () =>
               Assert.AreEqual("Het e-mail adres moet in het format gebruiker@provider.domein staan", error);
        };
    }
}
```

Connect to Ranorex

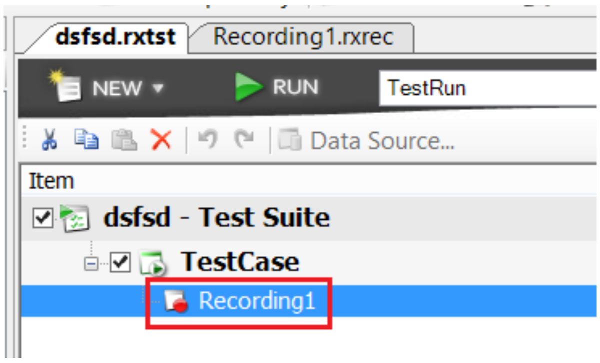
1) Install Ranorex
2) Create a new test project
3) Install the Nspec nuget
4) Add Ranorex.Core to the reference list
5) Create your specifications
6) Generate your stepdefinitions
7) Start Ranorex
8) Generate a new test solution



Apr 21, 2015 May and June Workshop Reminder
This is just a quick note to remind you about our upcoming Ranorex...

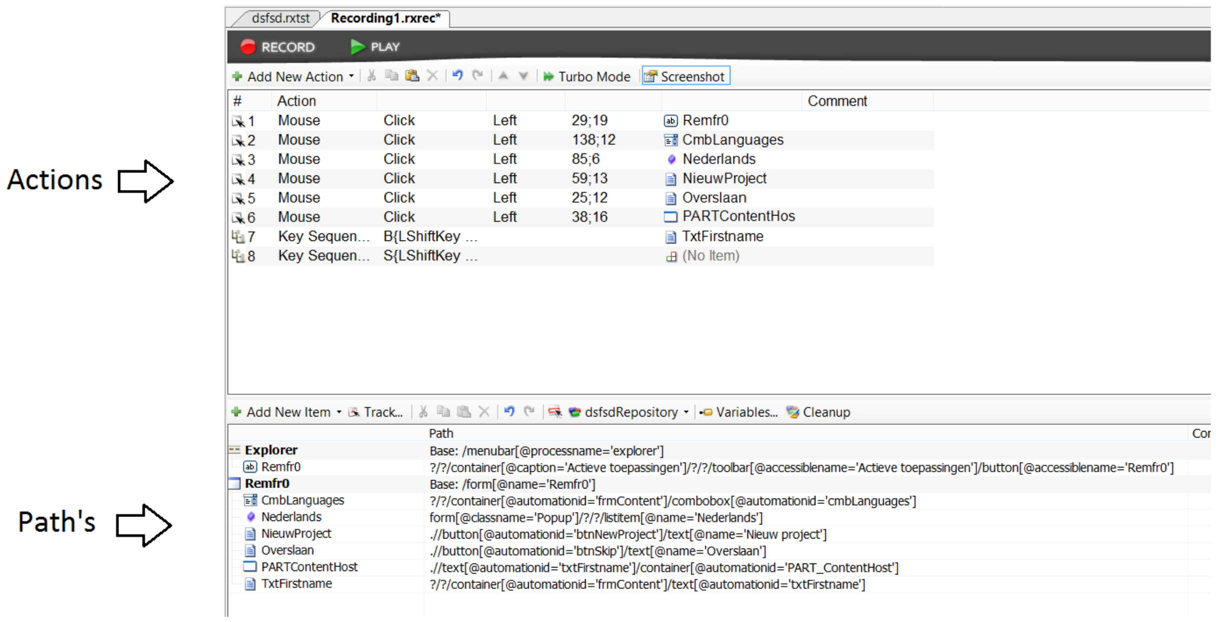
Apr 17, 2015 Planit - New Silver Service Partner for the Asia Pacific Region
With growing demand for Ranorex in the Asia Pacific region, we would like

9) Double click on the recording



10) Start your application
11) Press record in Ranorex
12) Set the record settings to “Global recording”
13) Perform all actions(clicking ,modifying tekst,...) that need to be done to perform your test. Make sure you have at least clicked on all items related to the test, otherwise the element can’t be found when connecting to specflow
14) Press stop recording
15) You go back to the ranorex main window

16) Now you see all the performed actions and the UI path to the controls

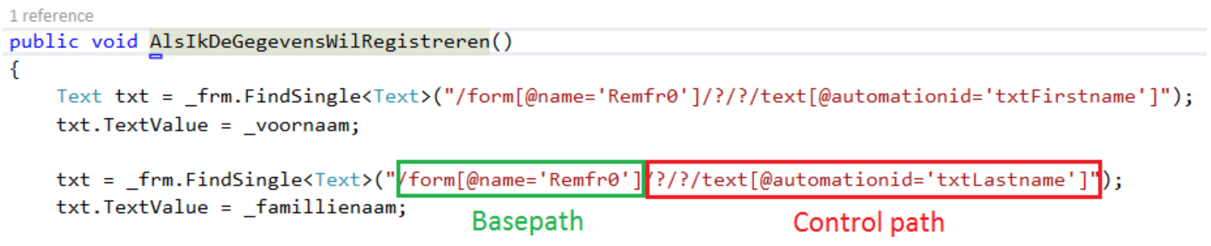


#	Action				Comment
1	Mouse	Click	Left	29;19	Remfr0
2	Mouse	Click	Left	138;12	CmbLanguages
3	Mouse	Click	Left	85;6	Nederlands
4	Mouse	Click	Left	59;13	NieuwProject
5	Mouse	Click	Left	25;12	Overslaan
6	Mouse	Click	Left	38;16	PARTContentHos
7	Key Sequen...	B{LShiftKey ...			TxtFirstname
8	Key Sequen...	S{LShiftKey ...			(No Item)

	Path	Cor
Explorer	Base: /menubar[@processname='explorer']	
Remfr0	?/?/container[@caption='Actieve toepassingen']/?/?/toolbar[@accessiblename='Actieve toepassingen']/button[@accessiblename='Remfr0']	
Remfr0	Base: /form[@name='Remfr0']	
CmbLanguages	?/?/container[@automationid='frmContent']/combobox[@automationid='cmbLanguages']	
Nederlands	form[@classname='Popup']/?/?/listitem[@name='Nederlands']	
NieuwProject	.//button[@automationid='btnNewProject']/text[@name='Nieuw project']	
Overslaan	.//button[@automationid='btnSkip']/text[@name='Overslaan']	
PARTContentHost	.//text[@automationid='txtFirstname']/container[@automationid='PART_ContentHost']	
TxtFirstname	?/?/container[@automationid='frmContent']/text[@automationid='txtFirstname']	

17) Go back to your steps

18) Via the Ranorex core, wire up the data/actions to your application using the paths you can find in Ranorex. The path to a control is made up out of 2 parts. The base path and the control path. An example is displayed in the image below.



```
1 reference
public void AlsIkDeGegevensWilRegistreren()
{
    Text txt = _frm.FindSingle<Text>("/form[@name='Remfr0']/?/?/text[@automationid='txtFirstname']");
    txt.TextValue = _voornaam;

    txt = _frm.FindSingle<Text>("/form[@name='Remfr0']/?/?/text[@automationid='txtLastname']");
    txt.TextValue = _famillienaam;
```

Attachments

A. Code for CodedUiGeneratorProvider

```
using CodedUIGeneratorProvider.Generator.SpecFlowPlugin;
using TechTalk.SpecFlow.Infrastructure;
[assembly:GeneratorPlugin(typeof(CodedUIGeneratorPlugin))]

namespace CodedUIGeneratorProvider.Generator.SpecFlowPlugin
{
    using System.CodeDom;

    using BoDi;

    using TechTalk.SpecFlow.Generator;
    using TechTalk.SpecFlow.Generator.Configuration;
    using TechTalk.SpecFlow.Generator.Plugins;
    using TechTalk.SpecFlow.Generator.UnitTestProvider;
    using TechTalk.SpecFlow.UnitTestProvider;
    using TechTalk.SpecFlow.Utils;

    /// <summary>
    /// The CodedUI generator plugin.
    /// </summary>
    public class CodedUIGeneratorPlugin : IGeneratorPlugin
    {
        /// <summary>
        /// The register dependencies.
        /// </summary>
        /// <param name="container">
        /// The container.
        /// </param>
        public void RegisterDependencies(ObjectContainer container)
        {
        }

        /// <summary>
        /// The register customizations.
        /// </summary>
        /// <param name="container">
        /// The container.
        /// </param>
        /// <param name="generatorConfiguration">
        /// The generator configuration.
        /// </param>
        public void RegisterCustomizations(ObjectContainer container,
SpecFlowProjectConfiguration generatorConfiguration)
        {
            container.RegisterTypeAs<CodedUIGeneratorProvider,
IUnitTestGeneratorProvider>();
            container.RegisterTypeAs<MsTest2010RuntimeProvider,
IUnitTestRuntimeProvider>();
        }

        /// <summary>
        /// The register configuration defaults.
        /// </summary>
        /// <param name="specFlowConfiguration">
        /// The spec flow configuration.
        /// </param>
        public void RegisterConfigurationDefaults(SpecFlowProjectConfiguration
specFlowConfiguration)
        {
        }
    }

    /// <summary>
    /// The CodedUI generator.
    /// </summary>
    public class CodedUIGeneratorProvider : MsTest2010GeneratorProvider
    {
        /// <summary>
        /// Initializes a new instance of the <see cref="CodedUiGeneratorProvider"/>
class.
        /// </summary>
        /// <param name="codeDomHelper">
        /// The code dom helper.
        /// </param>
        public CodedUIGeneratorProvider(CodeDomHelper codeDomHelper)
            : base(codeDomHelper)
        {
        }

        /// <summary>
        /// The set test class.
        /// </summary>
        /// <param name="generationContext">
        /// The generation context.
        /// </param>
        /// <param name="featureTitle">
        /// The feature title.
        /// </param>
        /// <param name="featureDescription">
        /// The feature description.
        /// </param>
        public override void SetTestClass(TestClassGenerationContext
generationContext, string featureTitle, string featureDescription)
        {
            base.SetTestClass(generationContext, featureTitle, featureDescription);

            foreach (CodeAttributeDeclaration declaration in
generationContext.TestClass.CustomAttributes)
            {
                if (declaration.Name ==
"Microsoft.VisualStudio.TestTools.UnitTesting.TestClassAttribute")
                {
                    generationContext.TestClass.CustomAttributes.Remove(declaration);
                    break;
                }
            }

            generationContext.TestClass.CustomAttributes.Add(new
CodeAttributeDeclaration(new
CodeTypeReference("Microsoft.VisualStudio.TestTools.UITesting.CodedUITestAttribute")))
;
        }
    }
}
```