MisconfiguredComponents Unit Test

# Unit Test kod:

## Vystrihovadlo test kod

using System.Linq;

using System.Text;

using Caliburn.Micro;

using Castle.Facilities.TypedFactory;

using Castle.MicroKernel.Handlers;

using Castle.MicroKernel.Registration;

using Castle.Windsor;

using Castle.Windsor.Configuration.Interpreters;

using Castle.Windsor.Diagnostics;

using Castle.Windsor.Installer;

using Mediaresearch.Framework.DataAccess.BLToolkit.Dao;

using Mediaresearch.Framework.Mapping;

using Mediaresearch.Framework.Mapping.Castle;

using Mediaresearch.Framework.Utilities.Configuration;

using MIR.Media.Cutting2.Gui.Configuration;

using MIR.Media.Cutting2.Shell.Configuration;

using Moq;

using NUnit.Framework;

namespace MIR.Media.Cutting2.Core.Tests

{

[TestFixture]

public class MisconfiguredComponentsTest

{

[SetUp]

public void SetUp()

{

const string containerConfig = "MIR.Media.Cutting2.Shell.Container.config";

const string installersConfig = "MIR.Media.Cutting2.Shell.Installers.config";

var provider = new Mock<IConfigurationProvider>();

provider.Setup(d => d.GetConfig<CuttingConfiguration>()).Returns(new CuttingConfiguration {ApplicationName = "MIR.Media.Cutting2"});

provider.Setup(d => d.GetConfig<ColorRecognitionConfiguration>()).Returns(new ColorRecognitionConfiguration {GaietyGrayThreshold = 5});

provider.Setup(d => d.GetConfig<ColorRecognitionConfiguration>()).Returns(new ColorRecognitionConfiguration {GaietyColorThreshold = 0});

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(containerConfig));

m\_windsorContainer.Register(Component.For<IConfigurationProvider>().Instance(provider.Object).IsDefault().Named("TestImplementation"));

m\_windsorContainer.Register(Component.For<IWindsorContainer>().Instance(m\_windsorContainer));

m\_windsorContainer.Register(Component.For<IWindowManager>().ImplementedBy<WindowManager>());

m\_windsorContainer.Install(Configuration.FromXmlFile(installersConfig));

m\_windsorContainer.AddFacility<TypedFactoryFacility>();

m\_windsorContainer.ResolveAll<IDaoSource>();

var configurator = new CastleDependencyMappingConfigurator(m\_windsorContainer, m\_windsorContainer.ResolveAll<MappingConfiguratorBase>());

configurator.Configure();

}

private WindsorContainer m\_windsorContainer;

[Test]

public void CheckForMisconfiguredComponents()

{

var diagnostic = new PotentiallyMisconfiguredComponentsDiagnostic(m\_windsorContainer.Kernel);

var handlers = diagnostic.Inspect();

if (handlers?.Any() == true)

{

var builder = new StringBuilder();

builder.AppendFormat("Misconfigured components ({0})\r\n", handlers.Count());

foreach (var handler in handlers)

{

var info = (IExposeDependencyInfo) handler;

var inspector = new DependencyInspector(builder);

info.ObtainDependencyDetails(inspector);

}

Assert.Fail(builder.ToString());

}

}

}

}

## Admin test kod

using System;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using Castle.MicroKernel.Handlers;

using Castle.MicroKernel.Resolvers.SpecializedResolvers;

using Castle.Windsor;

using Castle.Windsor.Diagnostics;

using Mediaresearch.Framework.DataAccess.BLToolkit.Dao;

using Mediaresearch.Framework.Utilities.Castle.ModelsInteractions.Castle;

using MIR.Media.Admin.Container.Installers;

using NUnit.Framework;

namespace MIR.Media.Admin.Tests

{

[TestFixture]

[Apartment(ApartmentState.STA)]

public class MisconfiguredComponentsTest

{

[SetUp]

public void SetUp()

{

m\_windsorContainer = Container.Container.Current;

m\_windsorContainer.AddFacility(new ListenerRegistrationFacility());

m\_windsorContainer.Kernel.Resolver.AddSubResolver(new CollectionResolver(Container.Container.Current.Kernel));

m\_windsorContainer.Install(Castle.Windsor.Installer.Configuration.FromXmlFile("MIR.Media.Admin.Container.config"));

//m\_windsorContainer.Install(FromAssembly.Containing<AdminAssemblyIdentificator>());

m\_windsorContainer.Install(new ConnectionInstaller());

m\_windsorContainer.Install(

new SplashScreenInstaller(),

new CommonComponentsInstaller(),

new RuntimeResolvedInstaller(),

new ScreensInstaller(),

new OtherComponentsInstaller()

);

m\_windsorContainer.ResolveAll<IDaoSource>();

}

private WindsorContainer m\_windsorContainer;

private void TestMe()

{

m\_windsorContainer.Install(

new SplashScreenInstaller(),

new CommonComponentsInstaller(),

new RuntimeResolvedInstaller(),

new ScreensInstaller(),

new OtherComponentsInstaller()

);

}

public Task<int> RunOnUiAsync(Func<int> f)

{

var dispatcherOperation = Application.Current.Dispatcher.InvokeAsync(f);

return dispatcherOperation.Task;

}

[Test]

public void CheckForMisconfiguredComponents()

{

var diagnostic = new PotentiallyMisconfiguredComponentsDiagnostic(m\_windsorContainer.Kernel);

var handlers = diagnostic.Inspect();

if (handlers?.Any() == true)

{

var builder = new StringBuilder();

builder.AppendFormat("Misconfigured components ({0})\r\n", handlers.Count());

foreach (var handler in handlers)

{

var info = (IExposeDependencyInfo) handler;

var inspector = new DependencyInspector(builder);

info.ObtainDependencyDetails(inspector);

}

Assert.Fail(builder.ToString());

}

}

}

}

## Zmenovadlo test kod

using System.Linq;

using System.Text;

using Castle.MicroKernel.Handlers;

using Castle.MicroKernel.Registration;

using Castle.Windsor;

using Castle.Windsor.Configuration.Interpreters;

using Castle.Windsor.Diagnostics;

using Castle.Windsor.Installer;

using Mediaresearch.Framework.DataAccess.BLToolkit.Dao;

using MIR.Media.Changing2.Core;

using MIR.Media.Changing2.Shell;

using NUnit.Framework;

namespace UnitTests

{

[TestFixture]

public class MisconfiguredComponentsTest

{

[SetUp]

public void SetUp()

{

const string path = "MIR.Media.Changing2.Shell.Container.config";

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(path));

m\_windsorContainer.Register(Component.For<IWindsorContainer>().Instance(m\_windsorContainer).LifestyleSingleton());

m\_windsorContainer.Install(FromAssembly.Containing<ShellAssemblyIdentificator>(), FromAssembly.Containing<CoreAssemblyIdentificator>());

m\_windsorContainer.Resolve<IMediaDataDaoSource>();

}

private WindsorContainer m\_windsorContainer;

[Test]

public void CheckForMisconfiguredComponents()

{

var diagnostic = new PotentiallyMisconfiguredComponentsDiagnostic(m\_windsorContainer.Kernel);

var handlers = diagnostic.Inspect();

if (handlers?.Any() == true)

{

var builder = new StringBuilder();

builder.AppendFormat("Misconfigured components ({0})\r\n", handlers.Count());

foreach (var handler in handlers)

{

var info = (IExposeDependencyInfo) handler;

var inspector = new DependencyInspector(builder);

info.ObtainDependencyDetails(inspector);

}

Assert.Fail(builder.ToString());

}

}

}

}

# Postup:

Je nekolik druhu postupu. Zmenovadlo napr tvori container kteremu dava pomoci xmlInterpretera cestu ke configu. Admin ma zase container.Current. Tam je bezparametricky konstruktor. Viz testy vyse.

## Admin:

V adminovi jsem mel vyjimku, ze MediumVersionDialogViewModel ma zavislost na IMediumVersion. Tato trida se ale tvori jen pomoci new . Nekdo ji chtel resolvovat pomoci factory, ktera se nepouzivala:

namespace MIR.Media.Admin.Screens.MediumAdministration.MediumVersions.MediumVersionDialog

{

public interface IMediumVersionDialogFactory

{

IMediumVersionDialogViewModel Create(IMediumVersion mediumVersion);

void Release(IMediumVersionDialogViewModel model);

}

Ve spravovadle byl RuntimeResolvedInstaller, kde bylo zaregistrovano a taky se to nikde nepouzivalo:

public void Install(IWindsorContainer container, IConfigurationStore store)

{

//container.Register(

// Component.For<IMediumVersionDialogViewModel>().ImplementedBy<MediumVersionDialogViewModel>().LifeStyle.Transient

// );

}

Odstranil jsem Factory i registraci. Zavislost na rozhrani IMediumVersionDialogViewModel neni v zadnem konstruktoru

# Problemy:

**Problem:**

App.config (viz nasledujici problem) Vystrihovadlo ma tridu CuttingConfiguration a v app.configu ma konfiguracni sekce. V jednom installeru si z takoveto sekce bere propertu. V aplikaci to normalne funguje, jenze v testu ne. App.config se containeru ani davat nema to je proste spatne.

**Reseni:**

Filip mi poradil v testu si zaregistrovat instanci IConfigurationProvidera jako default, ktery bude jen pro potreby testu a pokud se spusti test, container pouzije tohoto defaultniho providera.

var provider = new Mock<IConfigurationProvider>();

provider.Setup(d => d.GetConfig<CuttingConfiguration>()).Returns(new CuttingConfiguration { ApplicationName = "MIR.Media.Cutting2" });

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(containerConfig));

m\_windsorContainer.Register(Component.For<IConfigurationProvider>().Instance(provider.Object).IsDefault().Named("TestovaciImplementace"));

m\_windsorContainer.Install(Castle.Windsor.Installer.Configuration.FromXmlFile(installersConfig)); // Spusti installery ve kterych se bere config z Castle

**Problem:**

**(**Viz predchazejici problem**)**

System.Configuration.ConfigurationErrorsException : Could not find section 'castle' in the configuration file associated with this domain.

**Reseni:**

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(config)); Tohle by melo dat app.config. Jenze v testovaci assembly mi to neda ani kdyz mu to dam explicitne (jako tam davam container.config).

**Problem:**

Neprojde installerem kt. je definovany v configu. Ve Vystrihovadle je Installers.config, ve kterem jsou definovany installery, ktere se maji nainstalovat. Neco jako je v Shedule.

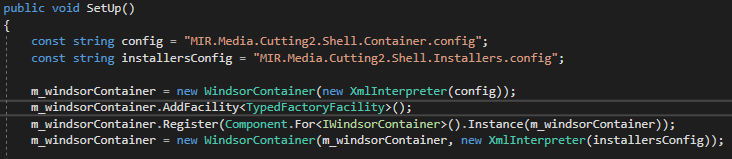
Tohle (je v kodu) nefunguje:

//m\_windsorContainer.Install(Castle.Windsor.Installer.Configuration.FromXmlFile(new XmlInterpreter(installersConfig).ToString()));

Oprava: funguje to! Viz test ve Vystrihovadle

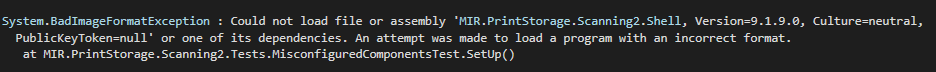
**Reseni:**

Pokud chci v unit testu pouzit tento installers.config, musim si vytvorit novy container, ktery vezme toho prvniho (ktery uz zna container.config) jako parenta a jako druhy argument dostane configuration interpretera.

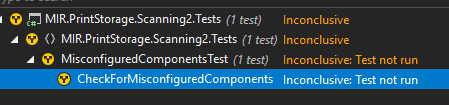


**Problem:**

BadImageFormatException

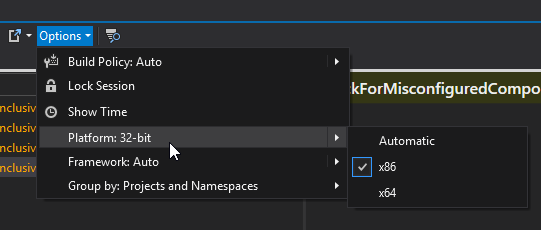
****

**Inconclusive:**

****

**Reseni:**

Stalo se to u skenovadla. Tam je Shell projekt porad na x86. Timpadem musi byt x86 i testovaci projekt jinak dostanu inconclusive error. V nastaveni unit testu musim zmenit platformu a test projde. Jak se s tim popere TFS nevim.



[‎11.‎03.‎2019 13:14]  Peter Hlavenka:

Cau, je nejaky duvod proc je projekt MIR.Printstorage.Scanning2.Shell na  platform target x86 ?  Muzu zmenit na x64?

[‎11.‎03.‎2019 13:14]  Petr Mitrofan:

musi zustat x32 kvuli ovladacum scaneru

[‎11.‎03.‎2019 13:14]  Peter Hlavenka:

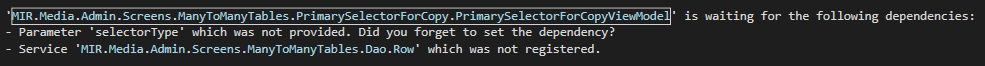
ok

Skenovadlo dostane svuj prvni unit test :D

[‎11.‎03.‎2019 13:16]  Petr Mitrofan:

zazraky e deji

**Problem:**

****

Pri resolvovani instance tridy (za rozhranim) se castlu predaji typy, na ktere si stezuje unit test:

IPrimarySelectorForCopyViewModel primarySelectorForCopyViewModel =

m\_container.Resolve<IPrimarySelectorForCopyViewModel>(new { selectorType = PrimarySelectorForCopyType.CopyTo, targetRow = m\_primaryTableViewModel.SelectedRow });

**Reseni:**

**Problem:**

STA Registrace ktere probihaji v aplikaci jsou v main threadu. Registrace v UnitTestu jsou ve worker threadu. Nekdy se v installeru vytvari instance trid a pak se zaregistruji jako .Instance() . Admin.ColumnHiderControl vola uz v konstruktoru NotifyOfPropertyChange(). Musi byt teda STA

**Reseni:**

Do Unit testu pridat atribut :

[TestFixture]

[Apartment(ApartmentState.STA)]

**Problem:**

Container.current => pri vytvareni currentu v testu se musi container vytvorit takto:

public void SetUp()

{

m\_windsorContainer = Container.Container.Current;

**Reseni:**

**Problem:**

Zhulena reference , nebo neni mozne pouzit nejaky typ.

**Reseni:**

Reference – Testovaci assembly musi mit reference na vsechny projekty, ktere chce pouzivat a to dokonce ve vyssi nebo rovne verzi .netu nez je pouzivana assembly

**Proble****m:**

Changing: Unit test se pousti nad appdata a proto nevidi na container.config. Tohle jsem prepral a skladam si cestu.

**Reseni:**

**Nejprve jsem se snazil poskladat stejny string jako dostavam v bootstrapperu. Na TFS to stejne padalo**

private static string GetProjectPath()

{

var codeBase = Assembly.GetExecutingAssembly().CodeBase;

var pathItems = codeBase.Split(Separator);

var start = pathItems.ToList().FindIndex(x => string.Equals("C:", x));

var end = pathItems.ToList().FindIndex(x => string.Equals("MIR.Media.Changing2", x)) + 1;

var projectPath = string.Join(Path.DirectorySeparatorChar.ToString(), pathItems.Skip(start).Take(end - start));

return projectPath;

}

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(Path.Combine(GetProjectPath(), "MIR.Media.Changing2.Shell\\bin\\Debug\\MIR.Media.Changing2.Shell.Container.config")));

**Nakonec stacilo misto sloziteho skladani stringu jen tohle:**

const string path = "MIR.Media.Changing2.Shell.Container.config";

m\_windsorContainer = new WindsorContainer(new XmlInterpreter(path));

**Problem:**

Tim, ze je test v jiné assembly nez je config, nedokaze zaregistrovat komponenty z configu, protože ty mají definovanou relativni cestu : type="MIR.Media.Changing2.Shell.Installers.MediaDataDbConfiguration, MIR.Media.Changing2.Shell">

**Reseni:**

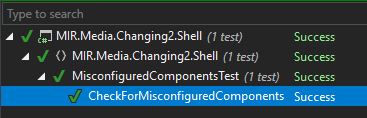
Resenim je presunout tento test do assembly, ve ktere jsou komponenty.

**Problem**:

Changing: Vytvorim instanci containeru, pustim installer, resolvnu dependencyDaoSource a dostanu error ze viewModel není registrovany. Pritom registrace VM a všech zavislosti probiha v installeru, kterym to urcite projde.

**Reseni:**

Musel jsem vytvorit novy installer – ShellInstaller. IWindowManager se totiz registroval v bootstrapperu, cimz ho unit test nenasel, protoze ten instaluje jen z installeru.



**Problem:**

Spusteni aplikace (ne testu) : ShellInstaller se mi instaloval dvakrat, protoze nejprve ho pustim v bootstrapperu kvuli IWindowManagerovi, ktereho potrebuju v main threadu, a pak se pousti jeste jednou asynchronne v akci ktera bezi na pozadi za splashscreenem:

m\_globalContainer.Install(new ShellInstaller());

void InitAction()

{

m\_globalContainer.Register(Component.For<IWindsorContainer>().Instance(m\_globalContainer).LifestyleSingleton());

m\_globalContainer.Install(FromAssembly.This(), FromAssembly.Containing<CoreAssemblyIdentificator>());

**Reseni:**

Vytvori si klic ktery se kontroluje pri instalaci. Dvakrat se vola jen instalace, instance ShellInstalleru se vytvari jen jednou.

public class ShellInstaller : IWindsorInstaller

{

private static string ShellInstallerKey { get; set; }

private static readonly object Lock = new object();

public ShellInstaller()

{

ShellInstallerKey = Guid.NewGuid().ToString();

}

public void Install(IWindsorContainer container, IConfigurationStore store)

{

if (!container.Kernel.HasComponent(ShellInstallerKey))

{

container.Register(Component.For<IWindowManager>().ImplementedBy<WindowManager>().Named(ShellInstallerKey));

}

}

}

**Problem:**

Zmenovadlo. U me to zbuildit slo a unit test prosel, u Karla taky, u Filipa a na TFS ne.

Viz [Problem1](#Problem1)

**Reseni:**

Viz [Problem1](#Problem1)

**Problem:**

Assembly has no entry point. (nema main metodu statickou) Kdyz jsem ve vystrihovadle odstranil z assembly s testama tridu UnitTests , zacal problem

class UnitTest

{

static void Main(string[] args)

{

}

}

**Reseni:**

Vrati ho zpet.

# Hotove softy - checkboxy:

Seznam softu, aplikaci a sluzeb

Admin

Approving

Catching

Coding

Cutting

Changing

Importing

Norming

Pricing

Scanning

SimLog

TvLogGenerator

Videomatching

DuplicityHunter

Picturematching

Reprocessing registration service

Import service

Schedule service

Export thumbnail service

PostOffice service

Export creative service

Pricing service

Admin

Admin

Admin

Admin

Admin

Admin

Admin

Admin