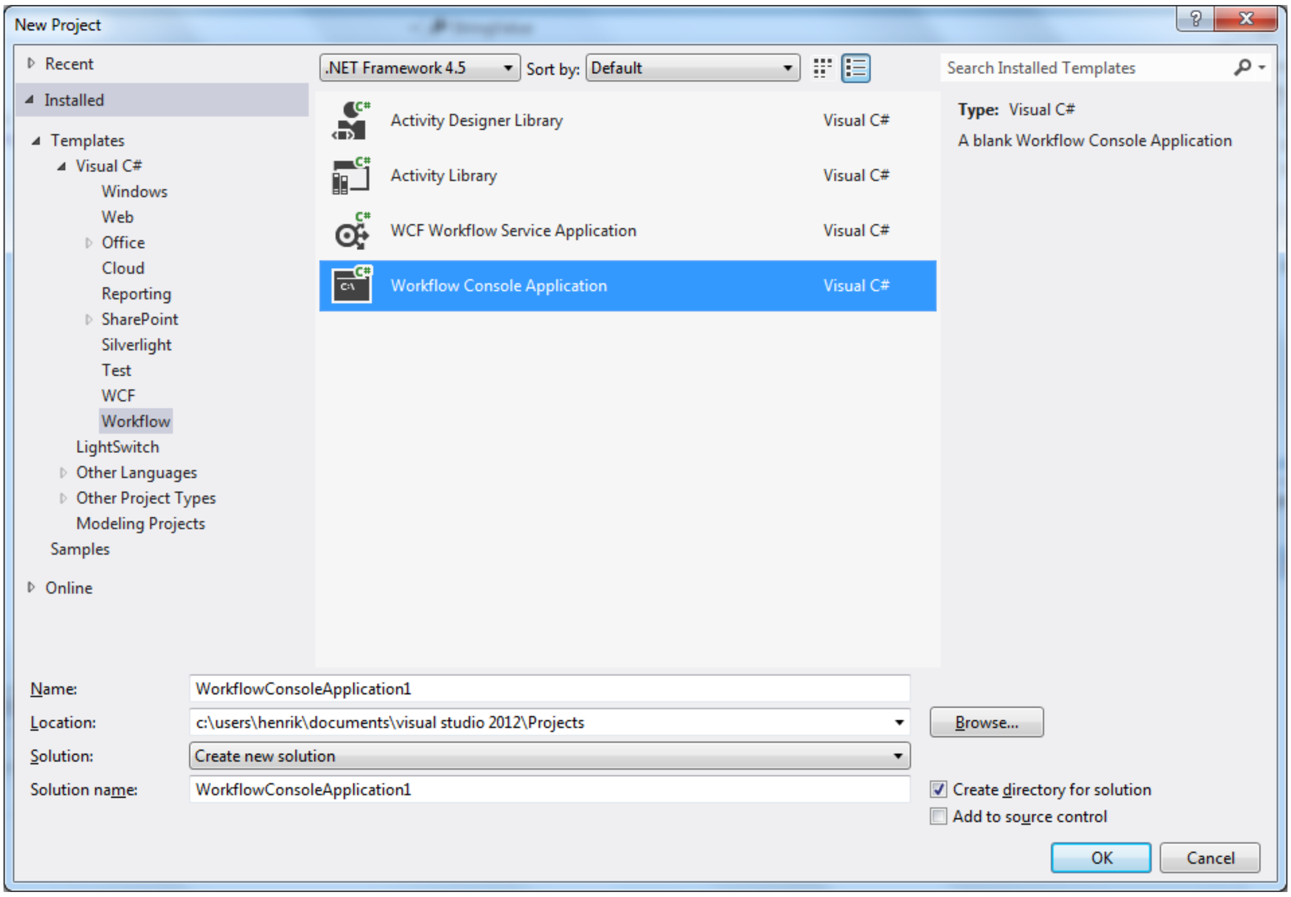
Introduction to Windows Workflow Foundation

- WF can
    - model
    - Configurate
    - maintain and
    - execute workflows



.NET Framework 4.5 Sort by: Default

Activity Designer Library	Visual C#
Activity Library	Visual C#
WCF Workflow Service Application	Visual C#
Workflow Console Application	Visual C#

Type: Visual C#
A blank Workflow Console Application

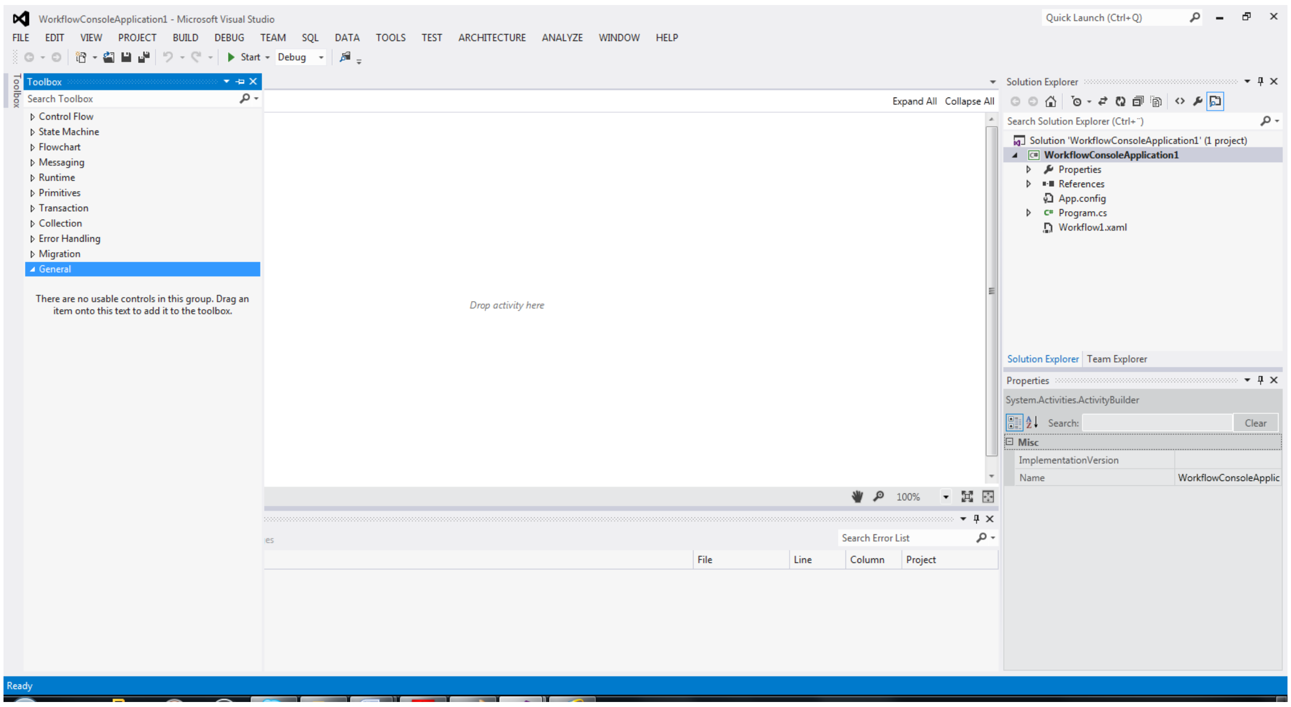
Name: WorkflowConsoleApplication1
Location: c:\users\henrik\documents\visual studio 2012\Projects
Solution: Create new solution
Solution name: WorkflowConsoleApplication1
[✓] Create directory for solution
[ ] Add to source control

- Several types
- Definition: Workflow is a sequence of related acts (or tasks)

Business Process modeling

- Is basic in many applications
- What is the order of a sequence of jobs?
- When?

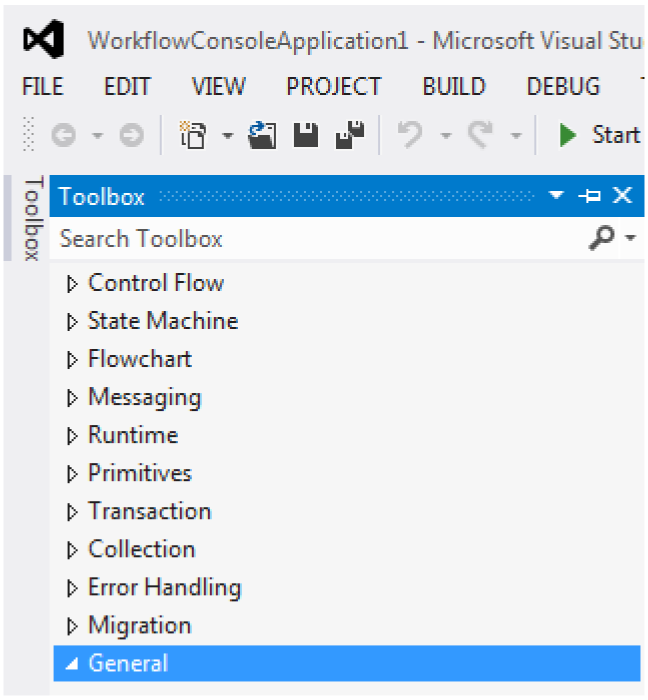
WF Building Blocks

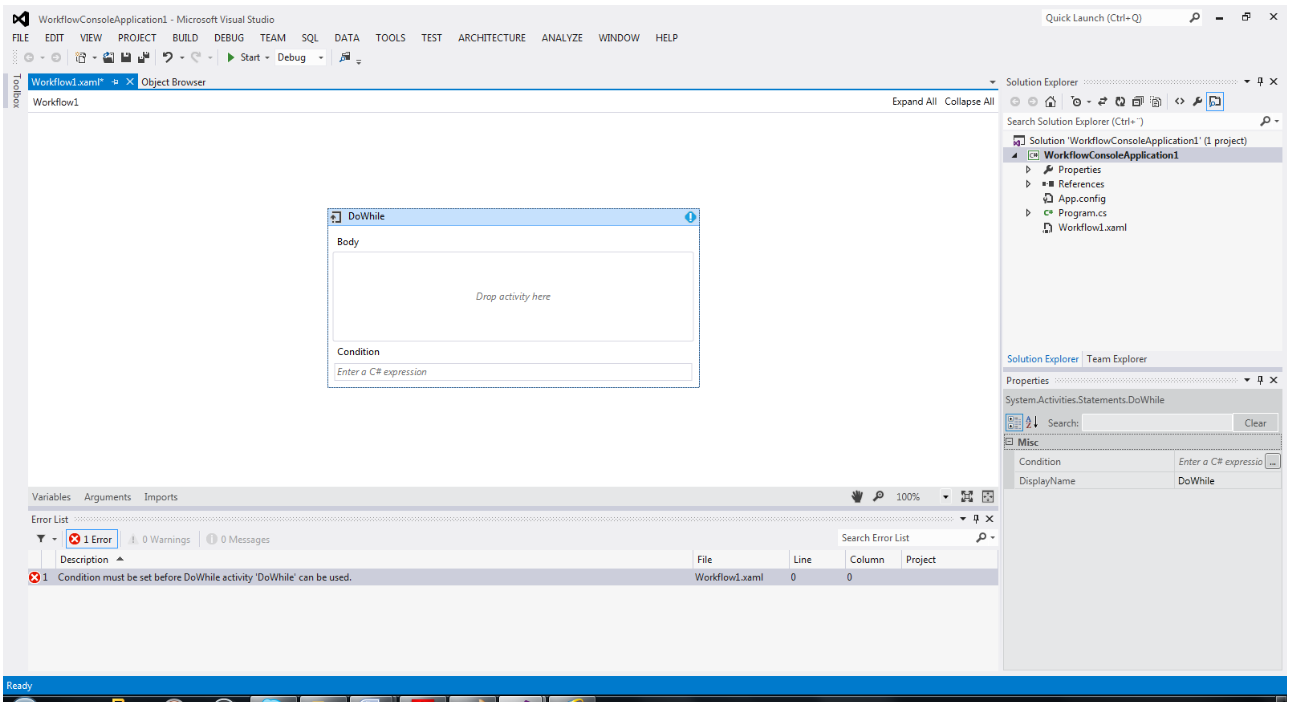


There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

- WF can be build as
  - console applications
  - Windows Forms or WPF
  - WCF Service
  - XML WEB Service

Use ToolBox





System.Activities.Statements.DoWhile

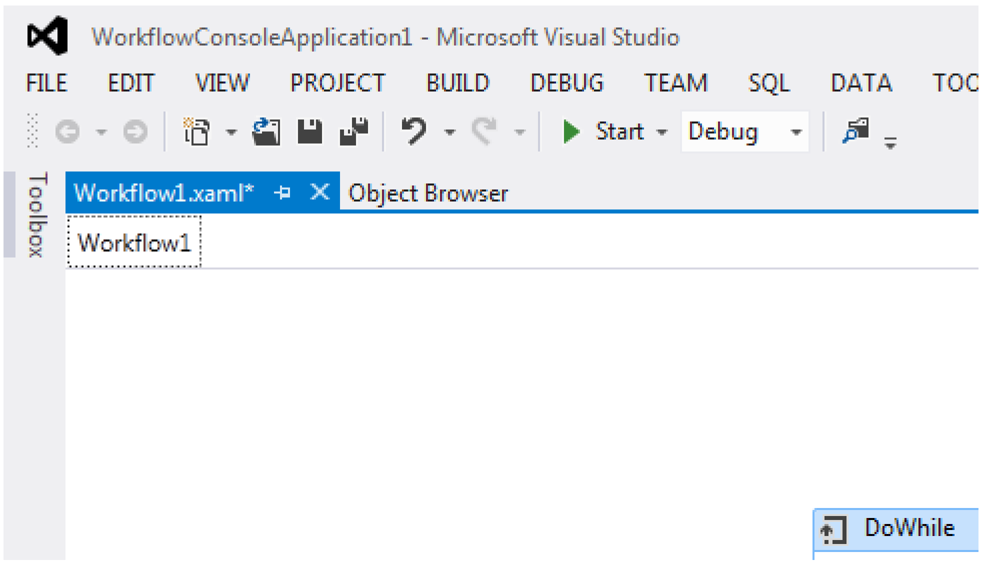
Misc

Condition	Enter a C# expressio
DisplayName	DoWhile

	Description	File	Line	Column	Project
1	Condition must be set before DoWhile activity 'DoWhile' can be used.	Workflow1.xaml	0	0	

XAML

- XAML is used


WorkflowConsoleApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL DATA TOC
Start
Debug
Toolbox
Workflow1.xaml*
Object Browser
Workflow1
DoWhile

Activities in WF

- Allows modeling of a business process
- A business process can be combined of any number of activities
    - 6 groups of activities
    - Control flow activities
    - Flowchart activities
    - messaging activities
    - Runtime activities
    - Transaction activities
    - Collection and exception activities

<activity> and objects

- All <Activity> elements are of course mapped to objects

WF engines

- Can be a simple console app
- GUI (Forms or WPF)
- Or a WCF Service

WorkflowInvoker

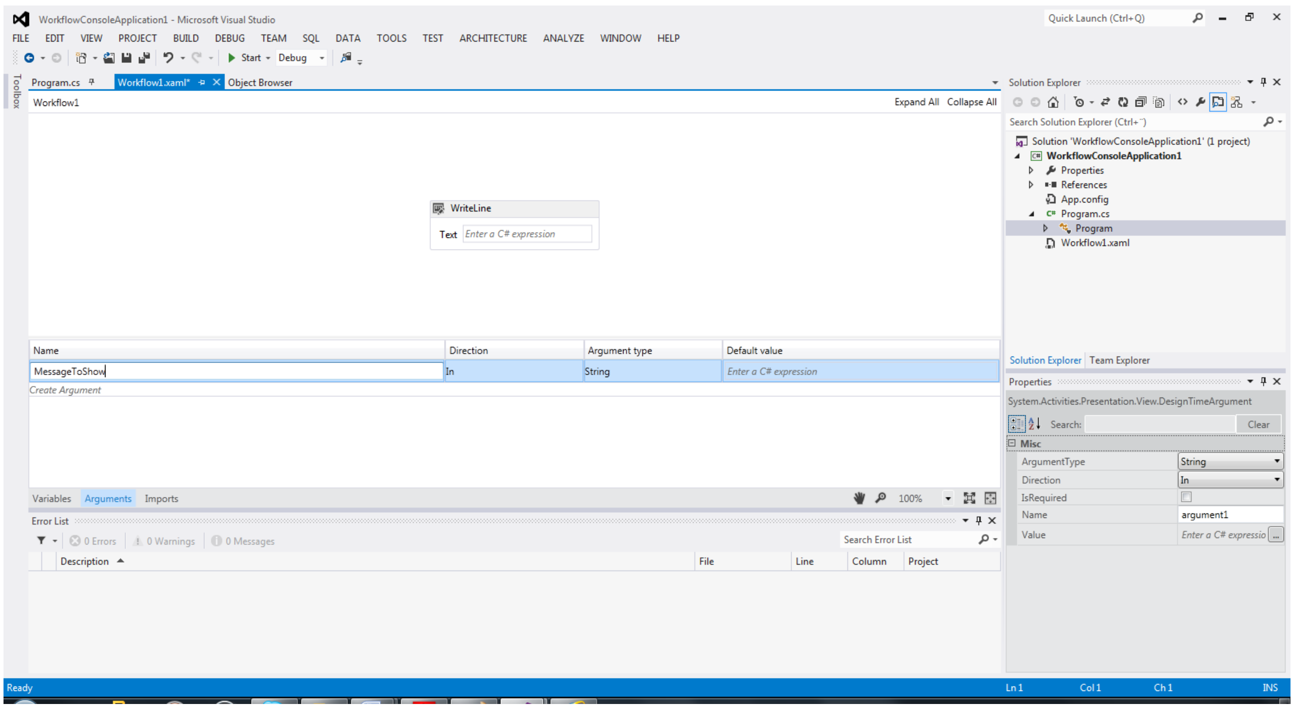
- Starts a workflow

```
using System;
using System.Linq;
using System.Activities;
using System.Activities.Statements;

namespace WorkflowConsoleApplication1
{

    class Program
    {
        static void Main(string[] args)
        {
            // Create and cache the workflow definition
            Activity workflow1 = new Workflow1();
            WorkflowInvoker.Invoke(workflow1);
        }
    }
}
```

- Invoke is synchronous (and blocks)
- Can take arguments: Use a Dictionary<string, object>

```
using System;
using System.Linq;
using System.Activities;
using System.Activities.Statements;


using System.Collections.Generic;

namespace WorkflowConsoleApplication1
{

    class Program
    {
        static void Main(string[] args)
        {

            // Get data from user, to pass to workflow.
            Console.Write("Please enter the data to pass the workflow: ");
            string wfData = Console.ReadLine();
            // Package up the data as a dictionary.
            Dictionary<string, object> wfArgs = new Dictionary<string, object>();
            wfArgs.Add("MessageToShow", wfData);
            // Create and cache the workflow definition
            Activity workflow1 = new Workflow1();
            WorkflowInvoker.Invoke(workflow1,wfArgs);
        }
    }
}
```

- Built-in workflow designer

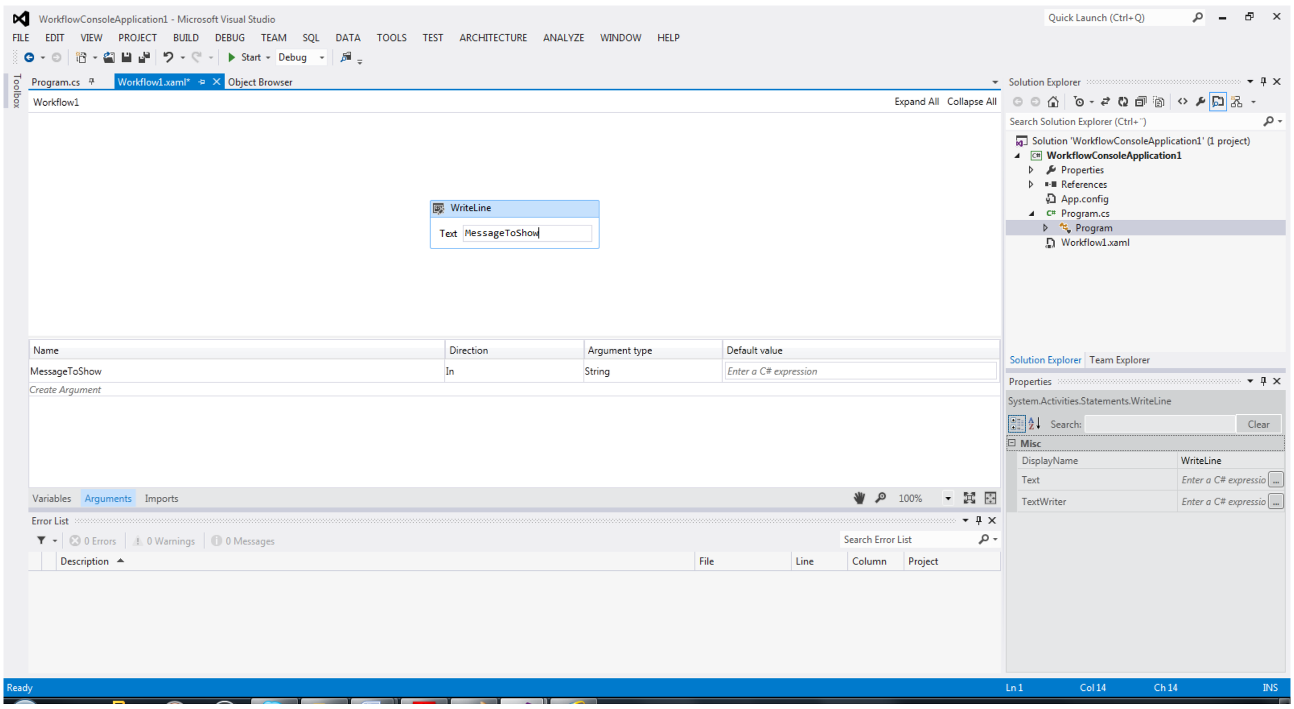

WorkflowConsoleApplication1 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL DATA TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start Debug
Toolbox
Program.cs Workflow1.xaml* Object Browser
Workflow1
Expand All Collapse All
WriteLine
Text Enter a C# expression
Variables Arguments Imports
100%
Error List
0 Errors 0 Warnings 0 Messages
Search Error List
Description File Line Column Project
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WorkflowConsoleApplication1' (1 project)
WorkflowConsoleApplication1
Properties
References
App.config
Program.cs
Program
Workflow1.xaml
Solution Explorer Team Explorer
Properties
Search: Clear
Misc
Ready Ln 1 Col 1 Ch 1 INS

Name	Direction	Argument type	Default value
MessageToShow	In	String	Enter a C# expression
Create Argument			

System.Activities.Presentation.View.DesignTimeArgument

ArgumentType	String
Direction	In
IsRequired	
Name	argument1
Value	Enter a C# expressio

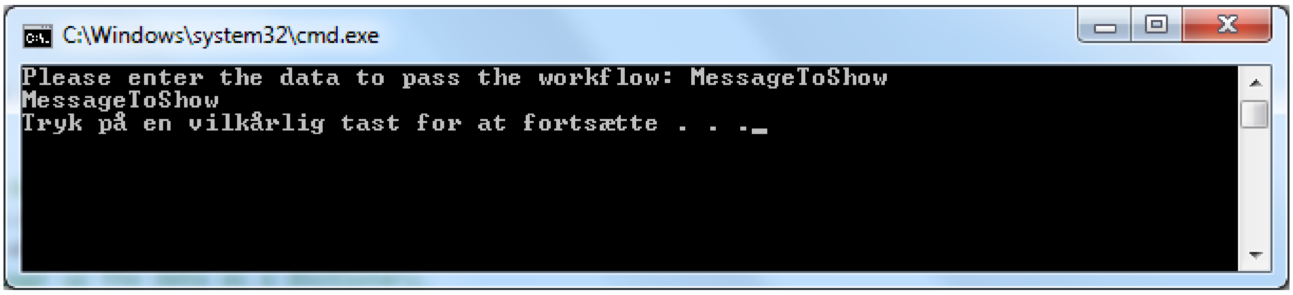
Notice: Arguments inbottom of designer


WorkflowConsoleApplication1 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL DATA TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start Debug
Toolbox
Program.cs Workflow1.xaml* Object Browser
Workflow1
Expand All Collapse All
WriteLine
Text MessageToShow
Variables Arguments Imports
100%
Error List
0 Errors 0 Warnings 0 Messages
Search Error List
Description File Line Column Project
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WorkflowConsoleApplication1' (1 project)
WorkflowConsoleApplication1
Properties
References
App.config
Program.cs
Program
Workflow1.xaml
Solution Explorer Team Explorer
Properties
Search: Clear
Misc
Ready Ln 1 Col 14 Ch 14 INS

Name	Direction	Argument type	Default value
MessageToShow	In	String	Enter a C# expression
Create Argument			

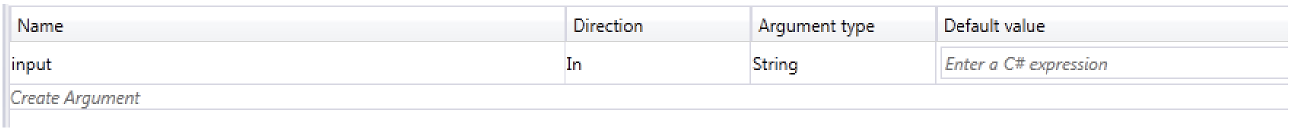
System.Activities.Statements.WriteLine

DisplayName	WriteLine
Text	Enter a C# expressio
TextWriter	Enter a C# expressio



```
Please enter the data to pass the workflow: MessageToShow
MessageToShow
Tryk på en vilkårlig tast for at fortsætte . . .
```

- Example 2:
- As argument is the string **input** created

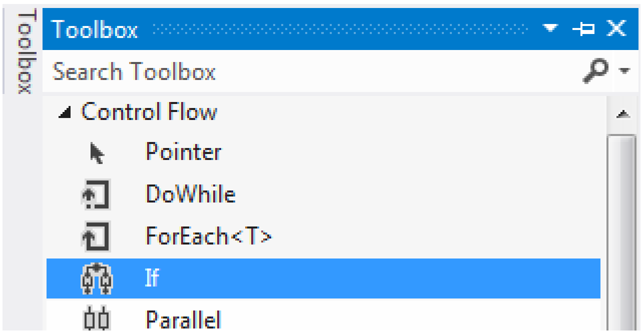


Name	Direction	Argument type	Default value
input	In	String	*Enter a C# expression*
Create Argument			

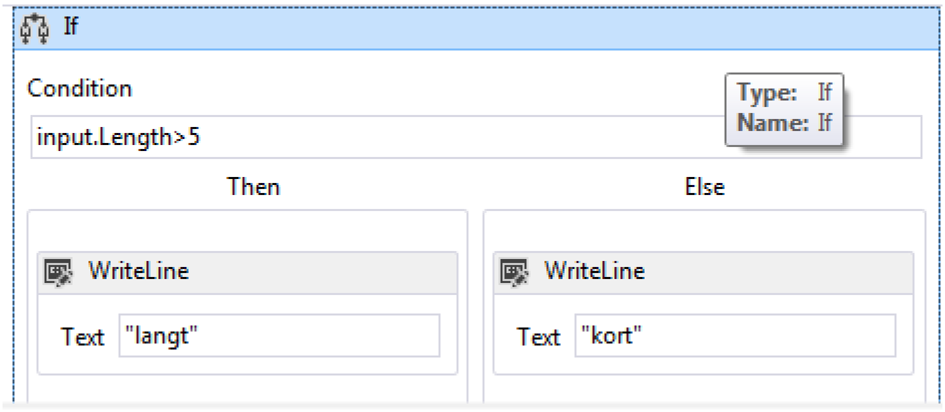
- Fetched (as above)

```
using System;
using System.Linq;
using System.Activities;
using System.Activities.Statements;

using System.Collections.Generic;

namespace WorkflowConsoleApplication1
{

    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Skriv et input ");
            string wfData = Console.ReadLine();
            Dictionary<string, object> wfArgs = new Dictionary<string, object>();
            wfArgs.Add("input", wfData);


            // Create and cache the workflow definition
            Activity workflow1 = new Workflow1();
            WorkflowInvoker.Invoke(workflow1, wfArgs);
        }
    }
}
```

- If as **Control Flow**



- Condition:

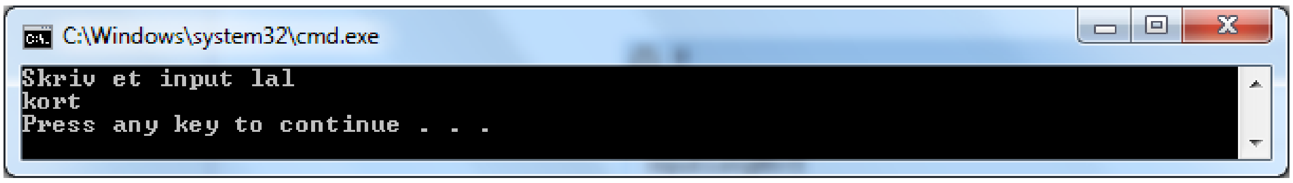


Condition
input.Length>5

Then: WriteLine — Text "langt"

Else: WriteLine — Text "kort"

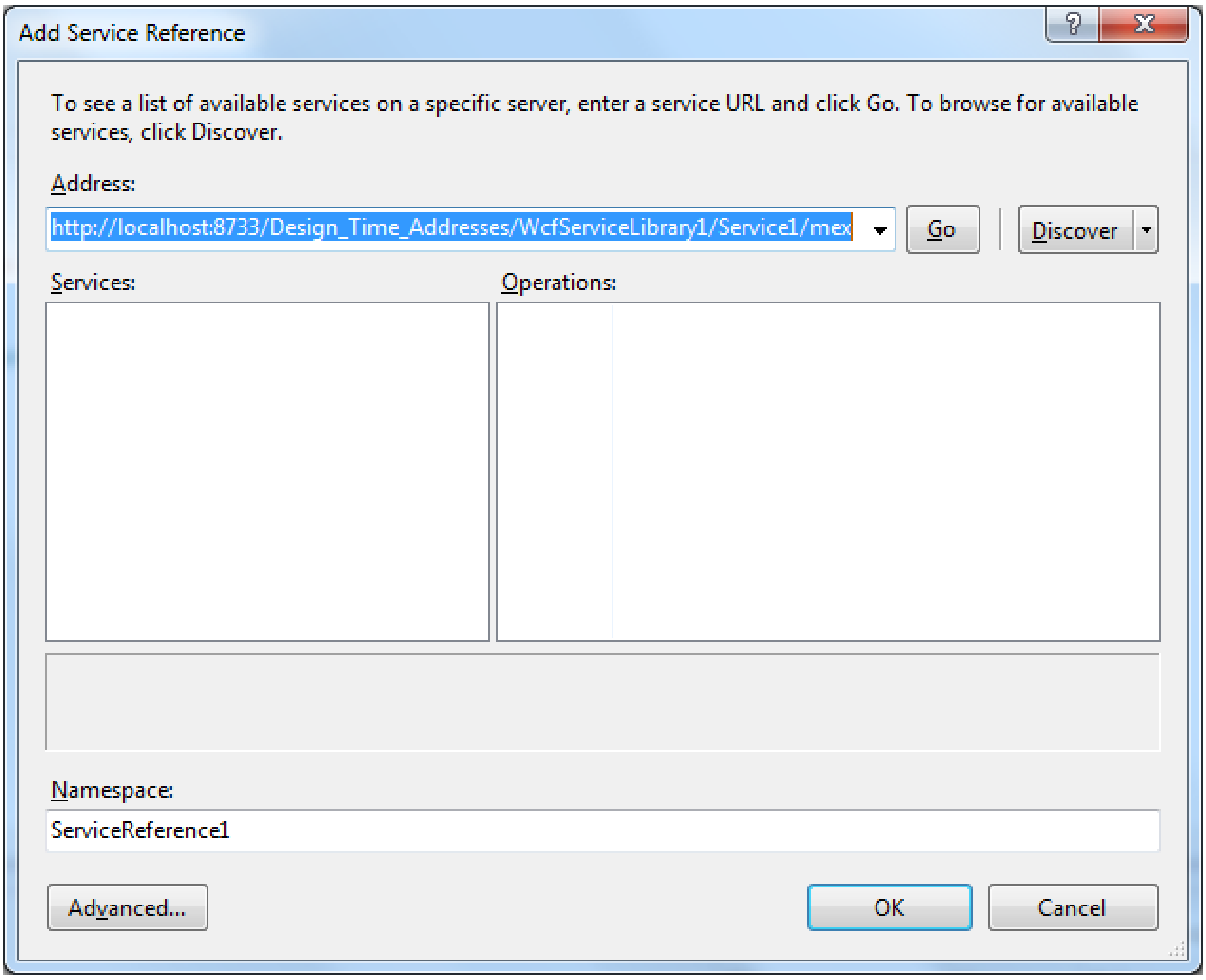
- Two WriteLine are used



```
Skriv et input lal
kort
Press any key to continue . . .
```

VS: Workflow as Service consumer

- Find the service (in the WF project)


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

http://localhost:8733/Design_Time_Addresses/WcfServiceLibrary1/Service1/mex

Namespace:

ServiceReference1

WF Assemblies and Namespaces

- Source: MSDN

Namespaces

Namespace	Description
System.Workflow.Activities	Defines activities that can be added to workflows to create and run an executable representation of a work process.
System.Workflow.Activities.Configuration	Provides classes that represent sections of the configuration file.
System.Workflow.Activities.Rules	Contains a set of classes that define the conditions and actions that form a rule.
System.Workflow.Activities.Rules.Design	Contains a set of classes that manage the **Rule Set Editor** and the **Rule Condition Editor** dialog boxes.
System.Workflow.ComponentModel	Provides the base classes, interfaces, and core modeling constructs that are used to create activities and workflows.
System.Workflow.ComponentModel.Compiler	Provides infrastructure for validating and compiling activities and workflows.
System.Workflow.ComponentModel.Design	Contains classes that developers can use to build custom design-time behavior for workflows and activities and user interfaces for configuring workflows and activities at design time.
System.Workflow.ComponentModel.Serialization	Provides the infrastructure for managing the serialization of activities and workflows to and from extensible Application Markup Language (XAML) and CodeDOM.
System.Workflow.Runtime	Contains classes and interfaces that you can use to control the workflow runtime engine and the execution of a workflow instance.
System.Workflow.Runtime.Configuration	Contains classes that are used to configure the workflow runtime engine.
System.Workflow.Runtime.DebugEngine	Contains classes and interfaces for use in debugging workflow instances.
System.Workflow.Runtime.Hosting	Contains classes that are related to services provided to the workflow runtime engine by the host application.
System.Workflow.Runtime.Tracking	Contains classes and an interface related to tracking services.

WF Activitetes

- Control Flow activities

Table 26-1. *The Control Flow Activities of WF*

Activities	Meaning in Life
DoWhile	A looping activity that executes contained activities at least once, until a condition is no longer true.
ForEach<T>	Executes an activity action once for each value provided in the ForEach<T>.Values collection.
If	Models an If-Then-Else condition.
Parallel	An activity that executes all child activities simultaneously and asynchronously.
ParallelForEach<T>	Enumerates the elements of a collection and executes each element of the collection in parallel.
Pick	Provides event-based control flow modeling.
PickBranch	A potential path of execution within a parent Pick activity.
Sequence	Executes a set of child activities sequentially.
Switch<T>	Selects one choice from a number of activities to execute, based on the value of a given expression of the type specified in this object's type parameter.
While	Executes a contained workflow element while a condition evaluates to true.

- Flowchart activities

Table 26-2. *The Flowchart Activities of WF*

Activities	Meaning in Life
Flowchart	Models workflows using the familiar "flow chart" paradigm. This is often the very first activity you will place on a new designer.
FlowDecision	A node that provides the ability to model a conditional node with two possible outcomes.
FlowSwitch<T>	A node that allows modeling a switch construct, with one expression and one outcome for each match.

- Messaging activities

Table 26-3. Common Messaging Activities of WF

Activities	Meaning in Life
CorrelationScope	Used to manage child message activities.
InitializeCorrelation	Initializes correlation without sending or receiving a message.
Receive	Receives a message from a WCF service.
Send	Sends a message to a WCF service.
SendAndReceiveReply	Sends a message to a WCF service and captures the return value.
TransactedReceiveScope	An activity that enables you to flow a transaction into a workflow or dispatcher-created server-side transactions.

- Runtime / Primitives activities

Table 26-4. The Runtime and Primitive Activities of WF

Activities	Meaning in Life
Persist	Requests that a workflow instance persist its state into a database using the WF persistence service.
TerminateWorkflow	Terminates the running workflow instance, raises the WorkflowApplication.Completed event in the host, and reports error information. Once the workflow is terminated, it cannot be resumed.
Assign	Allows you to set properties on an activity using the assignment values you defined via the workflow designer.
Delay	Forces a workflow to stop for a fixed amount of time.
InvokeMethod	Calls a method of a specified object or type.
WriteLine	Writes a specified string to a specified TextWriter-derived type. By default, this will be the standard output stream (a.k.a. the console); however, you can configure other streams, such as a FileStream.

- Transaction activities

***Table 26-5.** The Transaction Activities of WF*

Activities	Meaning in Life
CancellationScope	Associates cancellation logic within a main path of execution.
CompensableActivity	An activity that supports compensation of its child activities.
TransactionScope	An activity that demarcates a transaction boundary.

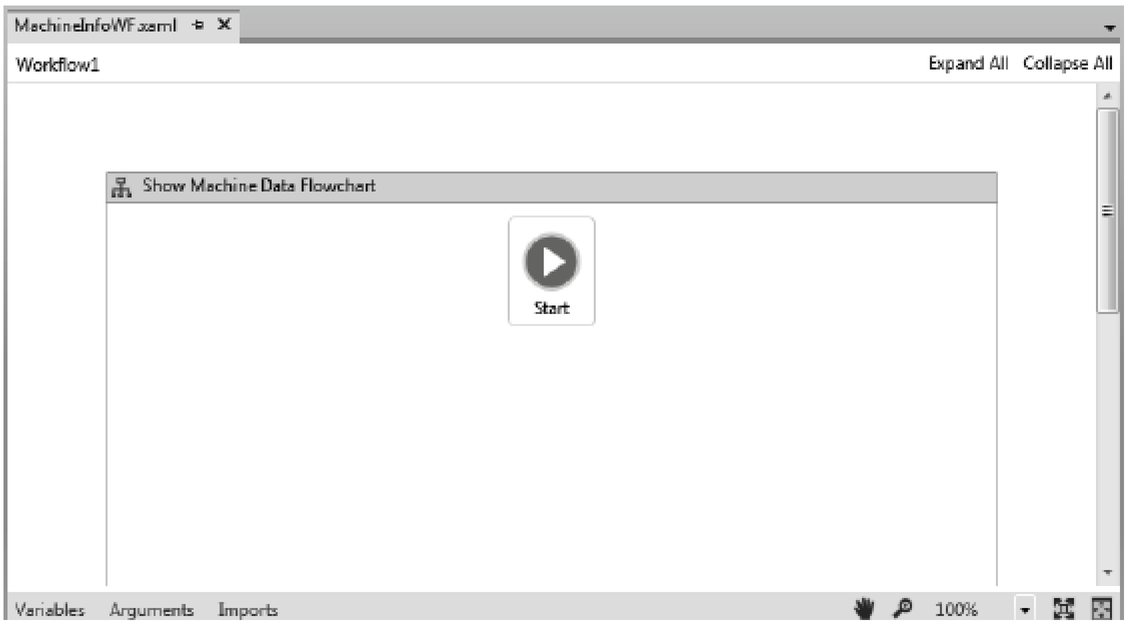
- Collection and exception activities

***Table 26-6.** The Collection and Error Handling Activities of WF*

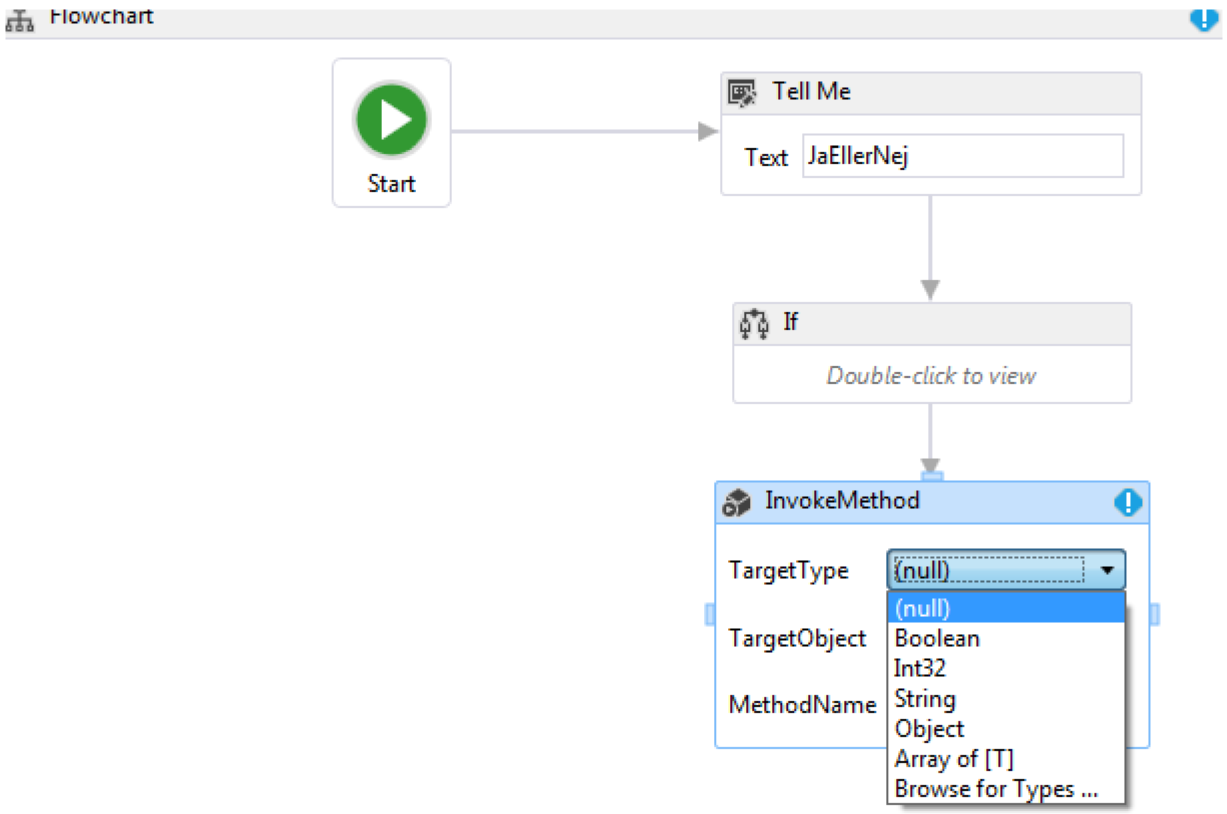
Activities	Meaning in Life
AddToCollection<T>	Adds an item to a specified collection.
ClearCollection<T>	Clears a specified collection of all items.
ExistsInCollection<T>	Indicates whether a given item is present in a given collection.
RemoveFromCollection<T>	Removes an item from a specified collection.
Rethrow	Throws a previously thrown exception from within a Catch activity.
Throw	Throws an exception.
TryCatch	Contains workflow elements to be executed by the workflow runtime within an exception handling block.

Builing a flowchart workflow

- Workflow service with sequential steps



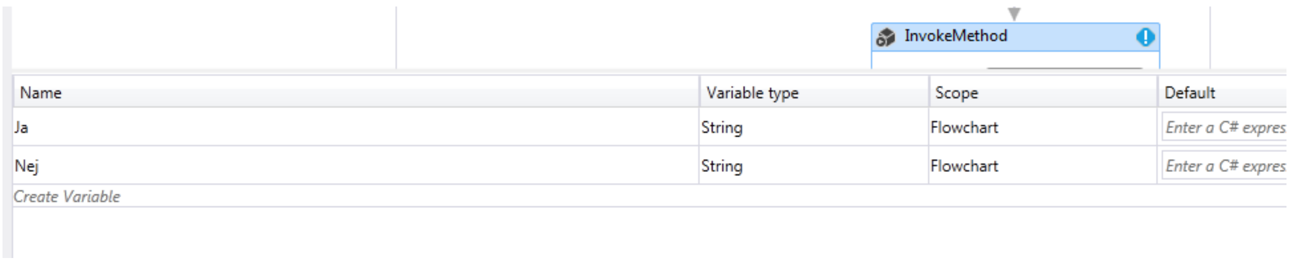
- Connect activities



Text: JaEllerNej

TargetType: (null)

- App wide variables

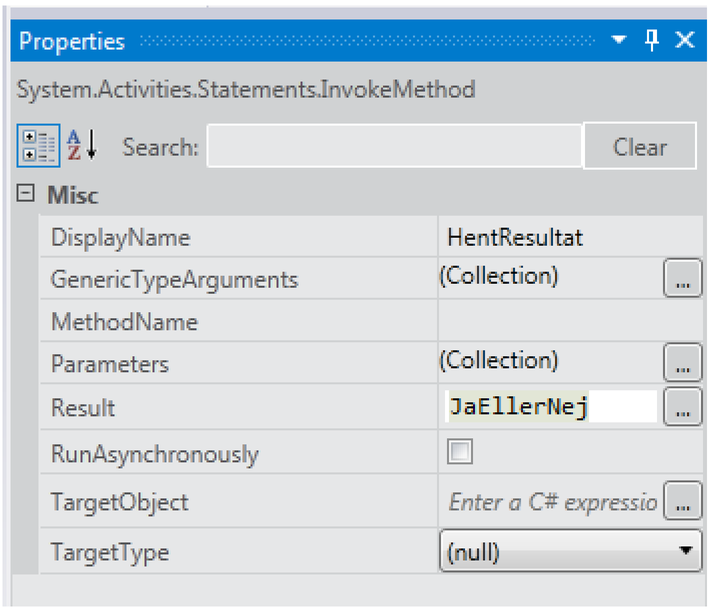


InvokeMethod

Name	Variable type	Scope	Default
Ja	String	Flowchart	Enter a C# expres
Nej	String	Flowchart	Enter a C# expres

Create Variable

- Use Properties



Properties

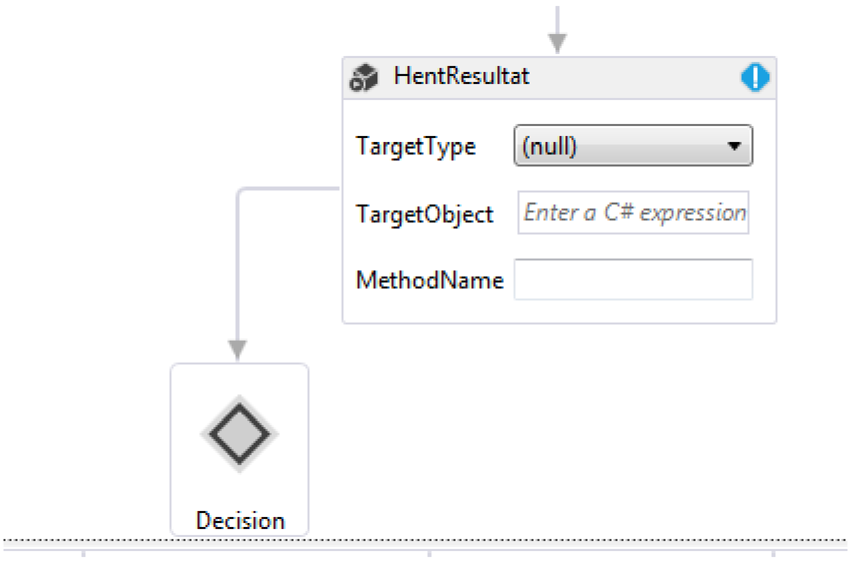
System.Activities.Statements.InvokeMethod

Search: Clear

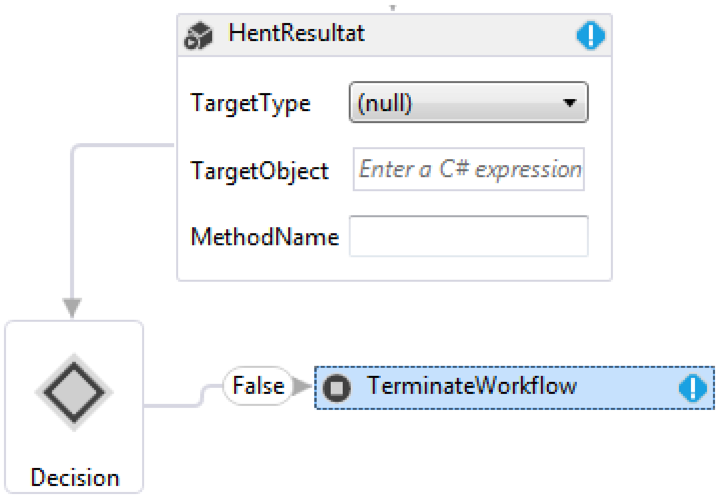
Misc

DisplayName	HentResultat
GenericTypeArguments	(Collection)
MethodName	
Parameters	(Collection)
Result	JaEllerNej
RunAsynchronously	☐
TargetObject	Enter a C# expressio
TargetType	(null)

  - Decision

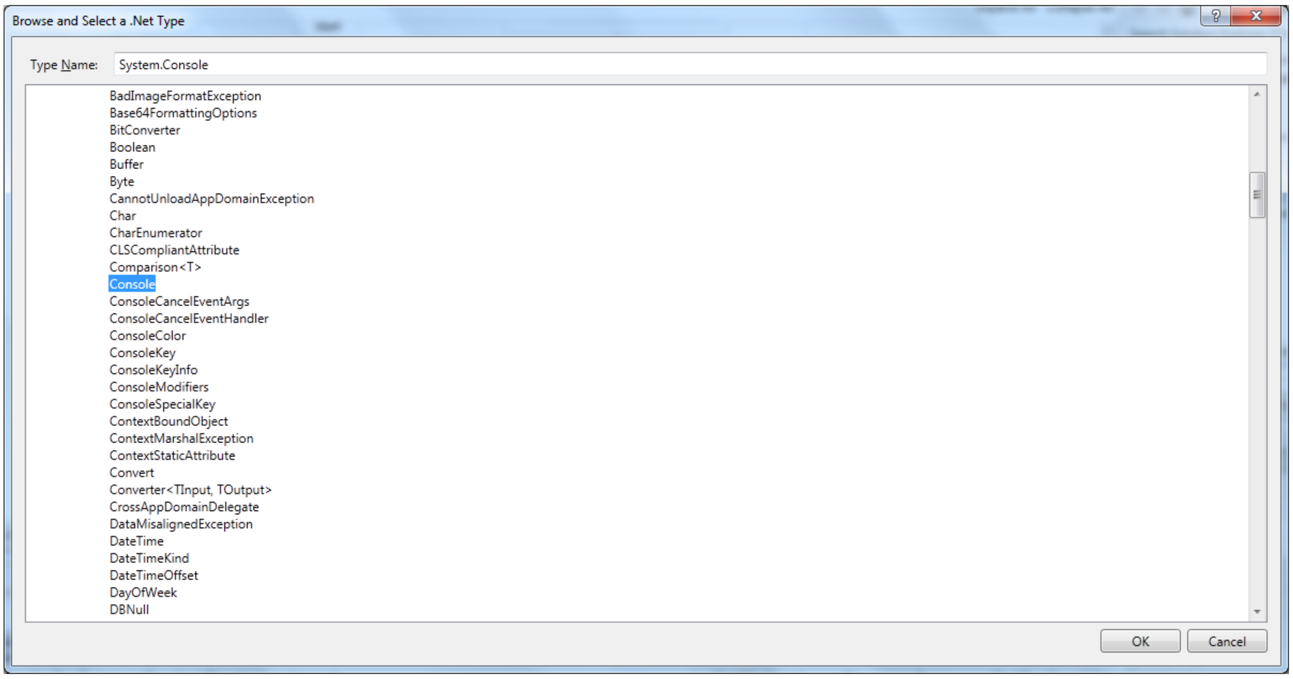


- Termination



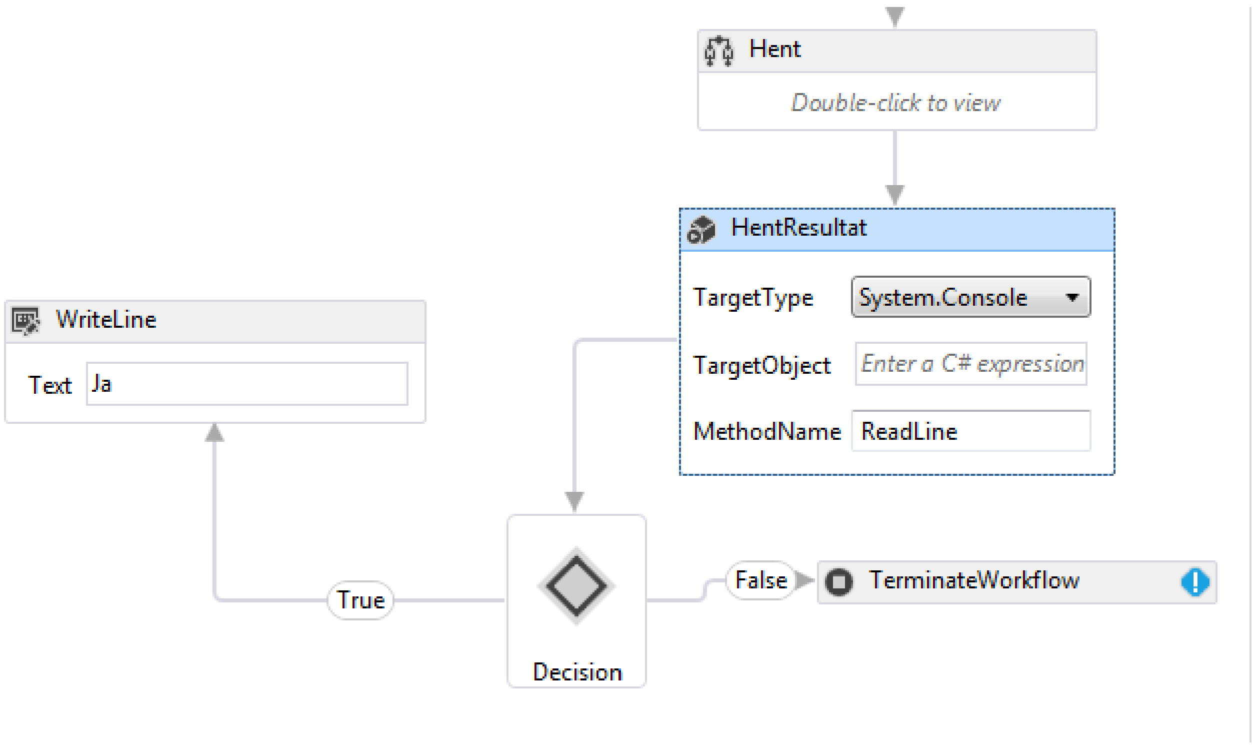
In *HentResultat* Find TargetObject (Here it is Console):

(browse)



Type Name: System.Console

BadImageFormatException
Base64FormattingOptions
BitConverter
Boolean
Buffer
Byte
CannotUnloadAppDomainException
Char
CharEnumerator
CLSCompliantAttribute
Comparison<T>
Console
ConsoleCancelEventArgs
ConsoleCancelEventHandler
ConsoleColor
ConsoleKey
ConsoleKeyInfo
ConsoleModifiers
ConsoleSpecialKey
ContextBoundObject
ContextMarshalException
ContextStaticAttribute
Convert
Converter<TInput, TOutput>
CrossAppDomainDelegate
DataMisalignedException
DateTime
DateTimeKind
DateTimeOffset
DayOfWeek
DBNull



TargetType System.Console

TargetObject Enter a C# expression

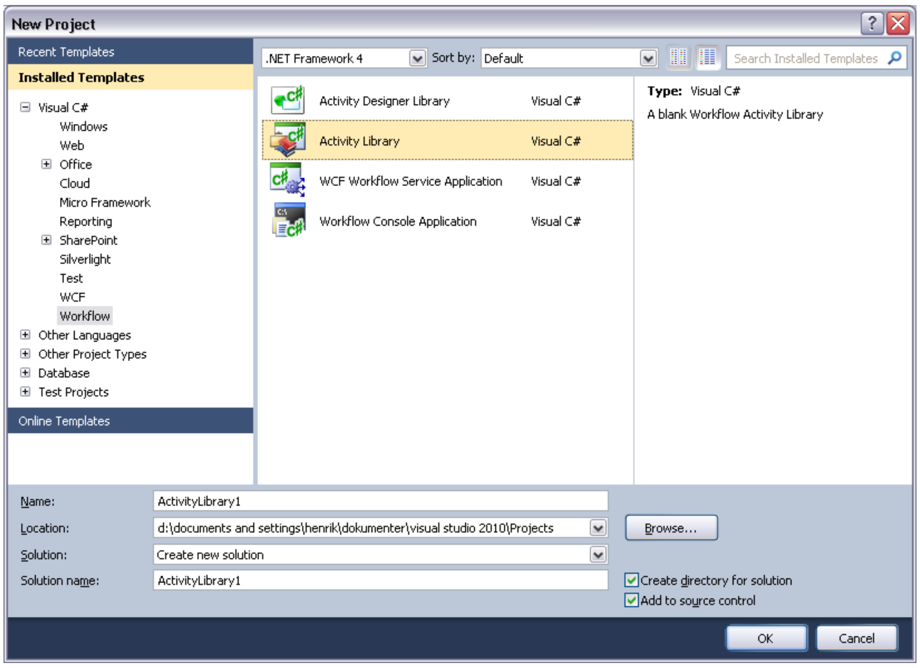
MethodName ReadLine

Text Ja

Placing a workflow in assemblies (DLL)

Practical example

- Create an Activity Library

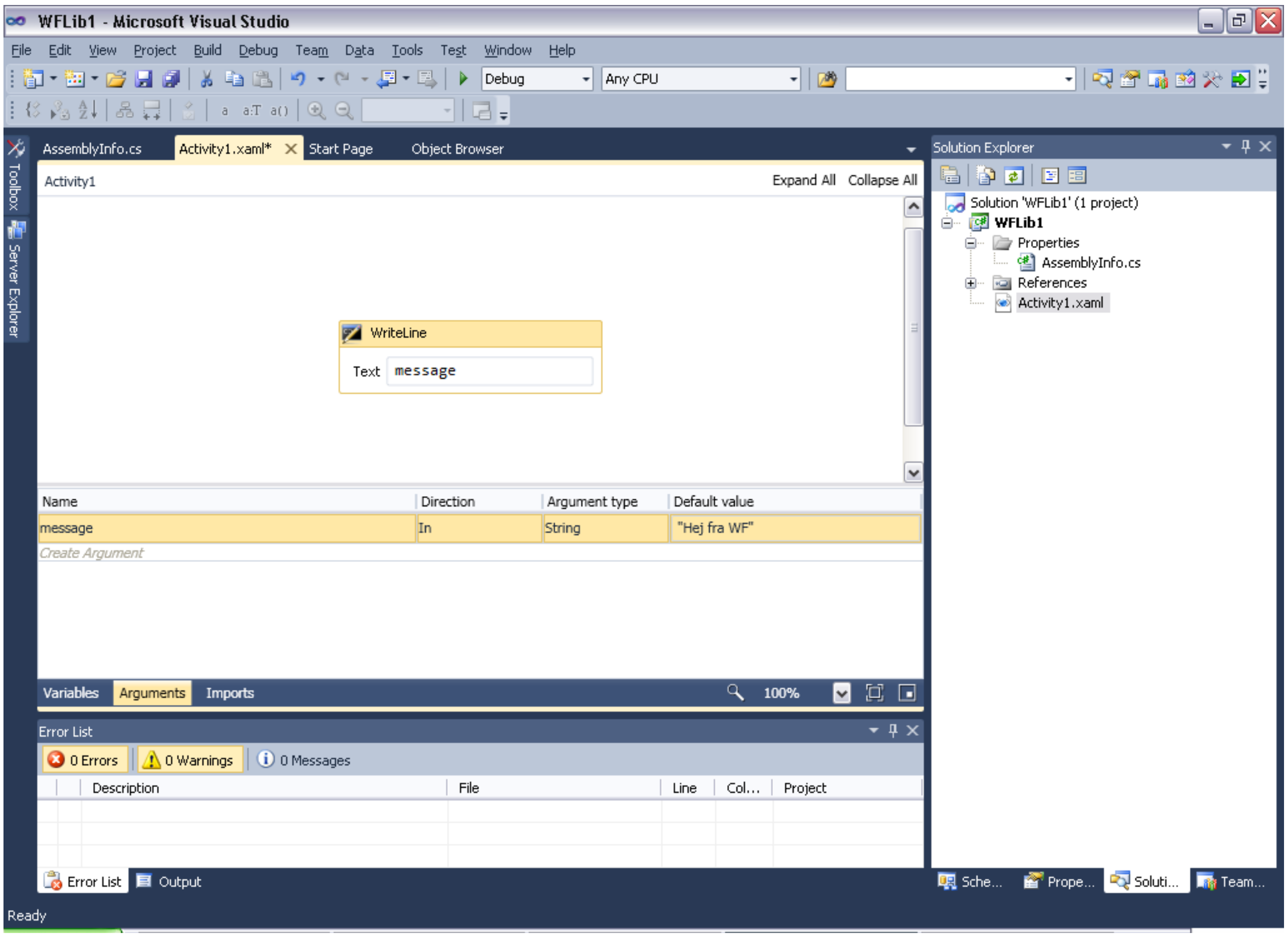


.NET Framework 4 Sort by: Default

Activity Designer Library	Visual C#
Activity Library	Visual C#
WCF Workflow Service Application	Visual C#
Workflow Console Application	Visual C#

Type: Visual C#
A blank Workflow Activity Library

Name: ActivityLibrary1
Location: d:\documents and settings\henrik\dokumenter\visual studio 2010\Projects
Solution: Create new solution
Solution name: ActivityLibrary1
[✓] Create directory for solution
[✓] Add to source control



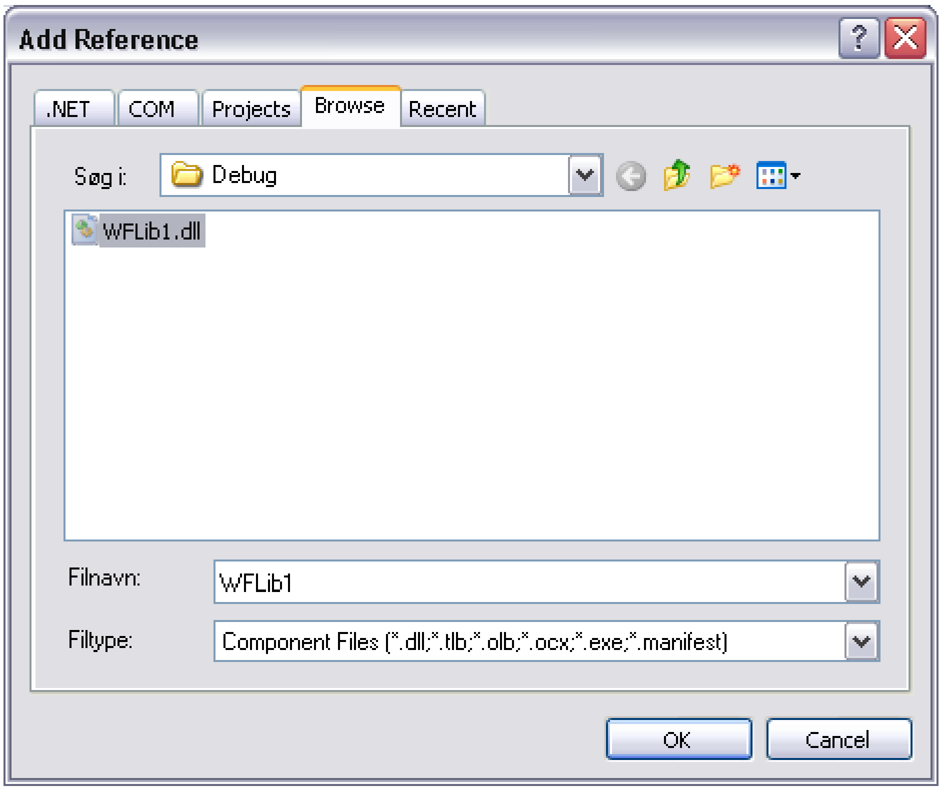
Text: message

Name	Direction	Argument type	Default value
message	In	String	"Hej fra WF"
Create Argument			

- Compile the DLL

Test AppliCation

- Create a simple console application
- Add Reference to the DLL


Add Reference

.NET | COM | Projects | Browse | Recent

Søg i: Debug

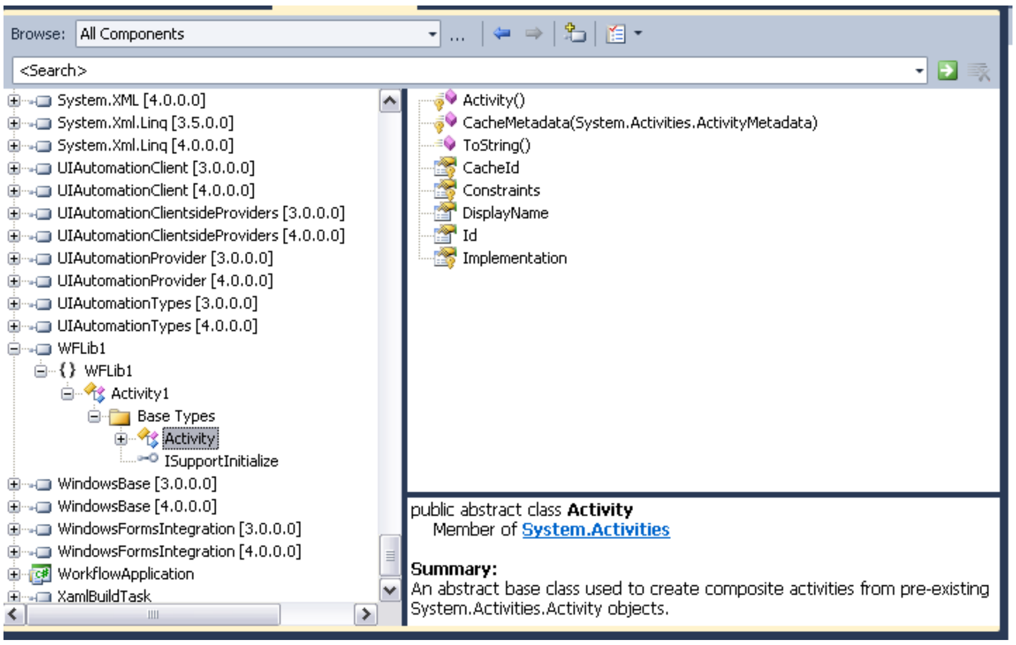
WFLib1.dll

Filnavn: WFLib1

Filtype: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

OK | Cancel

- Content can be seen


Browse: All Components

<Search>

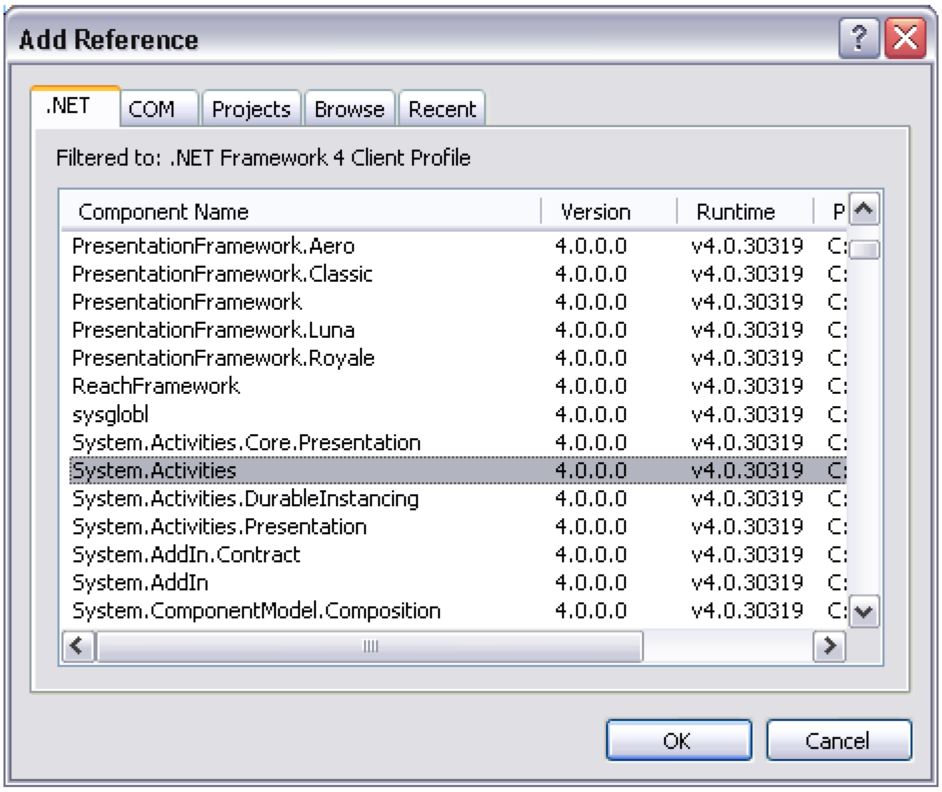
```
System.XML [4.0.0.0]
System.Xml.Linq [3.5.0.0]
System.Xml.Linq [4.0.0.0]
UIAutomationClient [3.0.0.0]
UIAutomationClient [4.0.0.0]
UIAutomationClientsideProviders [3.0.0.0]
UIAutomationClientsideProviders [4.0.0.0]
UIAutomationProvider [3.0.0.0]
UIAutomationProvider [4.0.0.0]
UIAutomationTypes [3.0.0.0]
UIAutomationTypes [4.0.0.0]
WFLib1
  {} WFLib1
    Activity1
      Base Types
        Activity
        ISupportInitialize
WindowsBase [3.0.0.0]
WindowsBase [4.0.0.0]
WindowsFormsIntegration [3.0.0.0]
WindowsFormsIntegration [4.0.0.0]
WorkflowApplication
XamlBuildTask
```

```
Activity()
CacheMetadata(System.Activities.ActivityMetadata)
ToString()
CacheId
Constraints
DisplayName
Id
Implementation
```

public abstract class **Activity**
Member of **System.Activities**

Summary:
An abstract base class used to create composite activities from pre-existing System.Activities.Activity objects.

- Set also **reference** to System.Activities

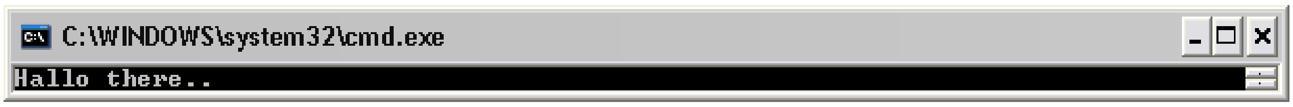


Filtered to: .NET Framework 4 Client Profile

Component Name	Version	Runtime	P
PresentationFramework.Aero	4.0.0.0	v4.0.30319	C:
PresentationFramework.Classic	4.0.0.0	v4.0.30319	C:
PresentationFramework	4.0.0.0	v4.0.30319	C:
PresentationFramework.Luna	4.0.0.0	v4.0.30319	C:
PresentationFramework.Royale	4.0.0.0	v4.0.30319	C:
ReachFramework	4.0.0.0	v4.0.30319	C:
sysglobl	4.0.0.0	v4.0.30319	C:
System.Activities.Core.Presentation	4.0.0.0	v4.0.30319	C:
System.Activities	4.0.0.0	v4.0.30319	C:
System.Activities.DurableInstancing	4.0.0.0	v4.0.30319	C:
System.Activities.Presentation	4.0.0.0	v4.0.30319	C:
System.AddIn.Contract	4.0.0.0	v4.0.30319	C:
System.AddIn	4.0.0.0	v4.0.30319	C:
System.ComponentModel.Composition	4.0.0.0	v4.0.30319	C:

- Code:

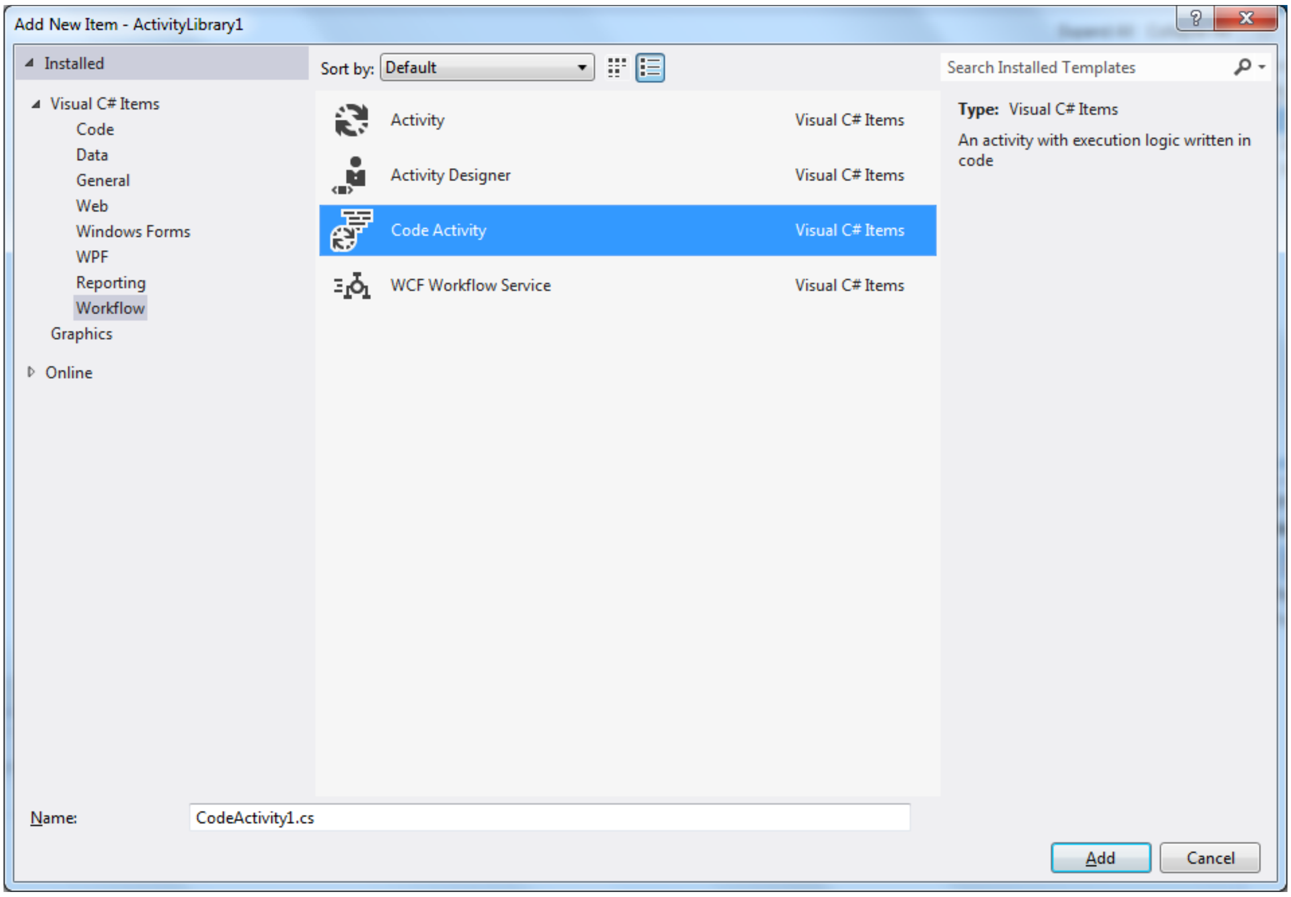
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Activities;
using WFLib1;

namespace TestWF
{
    class Program
    {
        static void Main(string[] args)
        {
            Dictionary<string, object> wfArgs = new Dictionary<string, object>();
            wfArgs.Add("message", "Hallo there..");

            WorkflowInvoker.Invoke(new WFLib1.Activity1(), wfArgs);

        }
    }
}
```



```
Hallo there..
```

Custom code activity

- Project type: Activity Library
- Do:
  - Add New Item
    - Vælg Code Activity



Activity	Visual C# Items
Activity Designer	Visual C# Items
Code Activity	Visual C# Items
WCF Workflow Service	Visual C# Items

Type: Visual C# Items

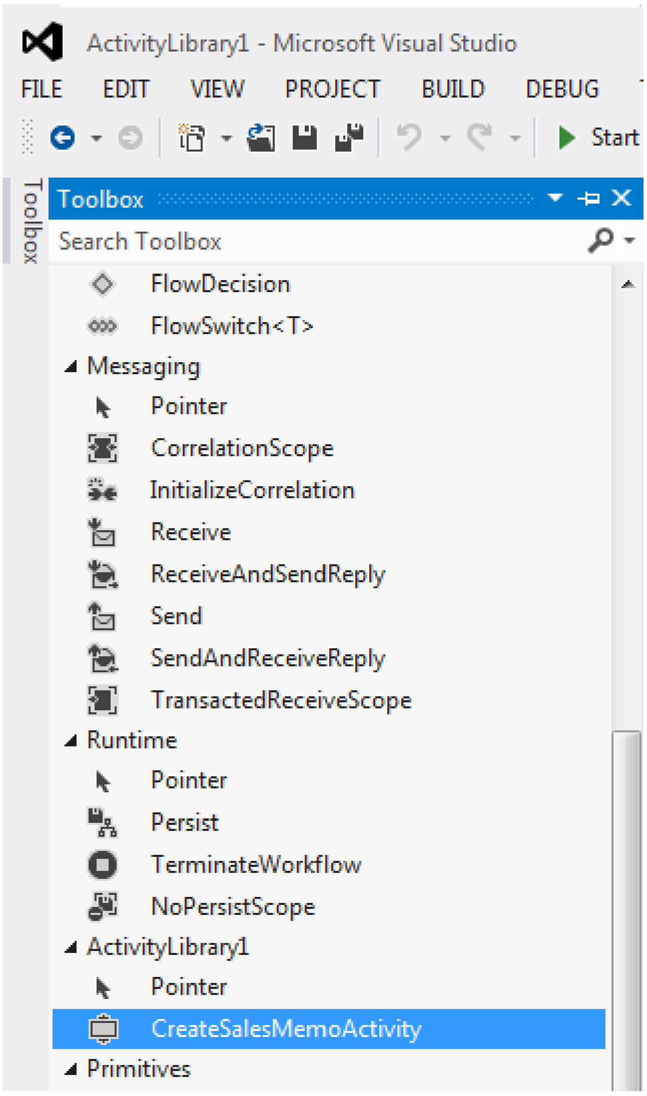
An activity with execution logic written in code

Name: CodeActivity1.cs

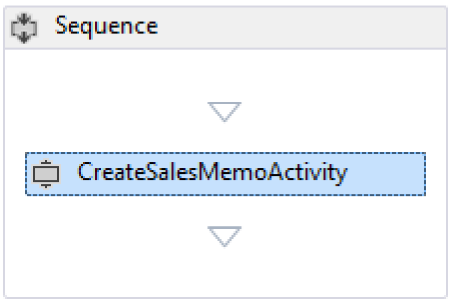
- Code:

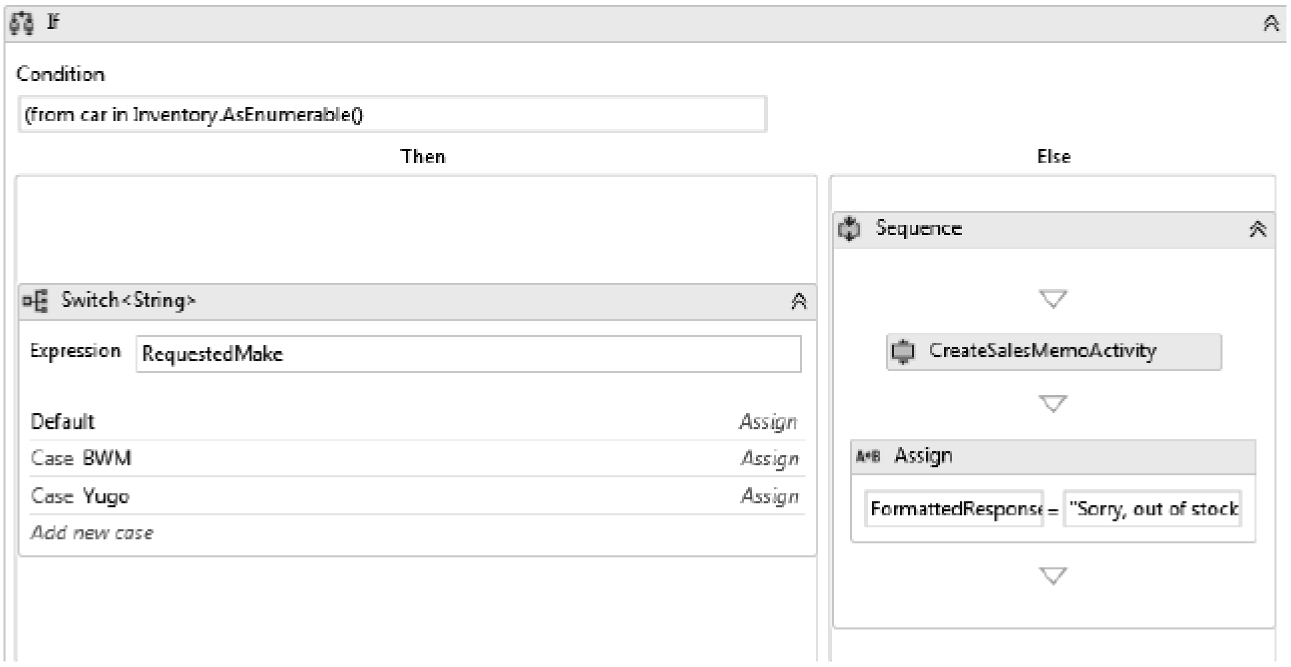
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Activities;

namespace ActivityLibrary1
{

    public sealed class CreateSalesMemoActivity : CodeActivity
{
// Two properties for the custom activity.
public InArgument<string> Make { get; set; }
public InArgument<string> Color { get; set; }
// If the activity returns a value, derive from CodeActivity<TResult>
// and return the value from the Execute method.
protected override void Execute(CodeActivityContext context)
{
// Dump a message to a local text file.
StringBuilder salesMessage = new StringBuilder();
salesMessage.AppendLine("***** Attention sales team! *****");
salesMessage.AppendLine("Please order the following ASAP!");
salesMessage.AppendFormat("1 {0} {1}\n",
context.GetValue(Color), context.GetValue(Make));
salesMessage.AppendLine("***********************************");
System.IO.File.WriteAllText("SalesMemo.txt", salesMessage.ToString());
}
}
}
```

- Now the activity can be found in toolbox



- Can for instance be placed in a sequence





WF Can invoke a WCF Service

- Communication using
  - SendActivity
  - ReceiveActivity

- Create the Service as normal with
  - [ServiceContract] and
  - [OperationContract]

- Create a SendActivity in the client application
  - A Service method can be associated with a SendActivity