FlowTasks – How to create a workflow

Contents

[Introduction 1](#_Toc337713820)

[Coding the workflow 1](#_Toc337713821)

[Configuring the workflow 10](#_Toc337713822)

[Starting the workflow 11](#_Toc337713823)

[Complete the Workflow 11](#_Toc337713824)

[Conclusion 12](#_Toc337713825)

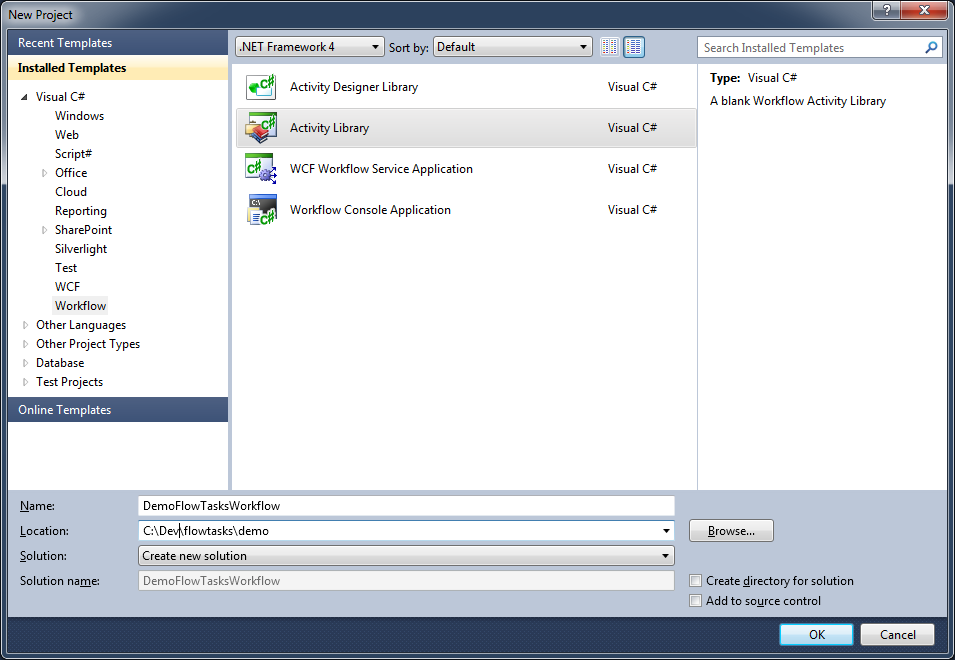
# Introduction

After you installed FlowTasks, using the steps described in the document FlowTasks Installation Guide.docs, you can then start to build your own workflows.

The code for the example can be found under Demo/DemoFlowTasksWorkflow folder.

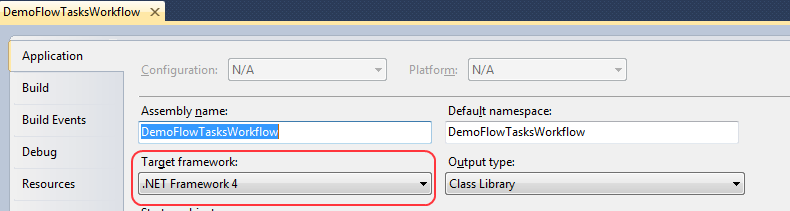
# Coding the workflow

1. Create a new workflow/Activity Library project



Rename the project DemoFlowTasksWorkflow.

Make sure you specify the correct target framework.



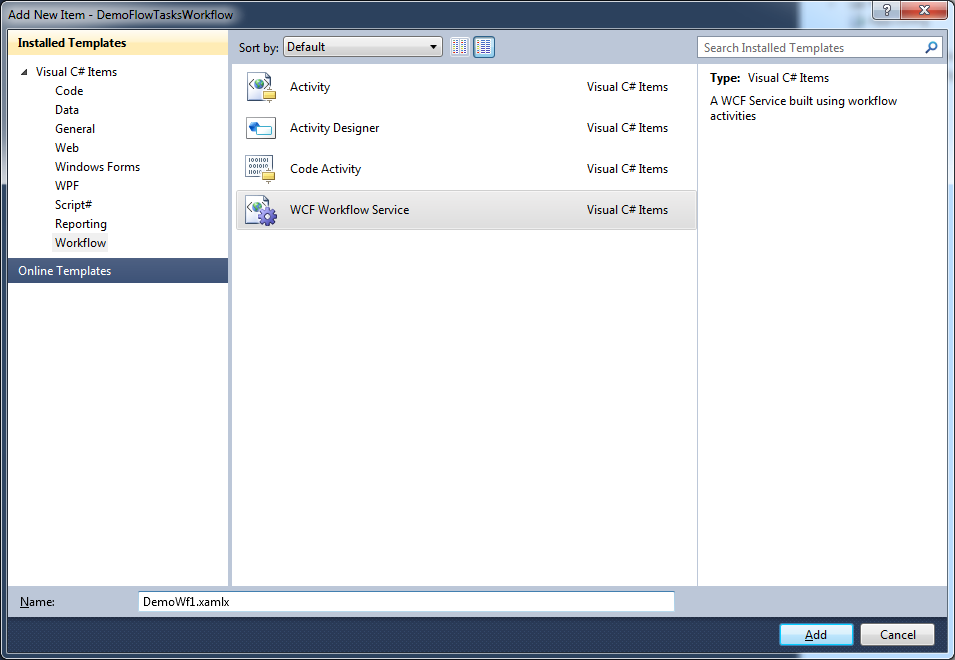
1. Reference all these libraries:

Flow.Tasks.Workflow.dll

log4net.dll

System.Activities.Core.Presentation.dll

1. Add a new WCF Workflow Service



Give it a meaningful name like DemoWf1.xamlx.

1. Add a new class file called DemoTask.cs and paste this code:

using System.Reflection;

using System.Activities;

using Flow.Tasks.Workflow.Activities;

using log4net;

namespace DemoFlowTasksWorkflow

{

public sealed class DemoTask : Activity

{

private readonly DelegateInArgument<TaskStatus> \_onInit = new DelegateInArgument<TaskStatus>();

private readonly DelegateInArgument<TaskStatus> \_onComplete = new DelegateInArgument<TaskStatus>();

private int \_correlationId;

public int CorrelationId

{

get

{

return \_correlationId;

}

set

{

\_correlationId = value;

}

}

private string \_taskCode;

public string TaskCode

{

get

{

return \_taskCode;

}

set

{

\_taskCode = value;

}

}

public DemoTask()

{

Implementation = () => new ApproveTask

{

AssignedToUsers = "{r.Dev}",

CorrelationId = CorrelationId,

DefaultResult = "Activity Expired",

Description = "This is just a demo task. It shows how simple it is to create a workflow.",

DisplayName = "Demo task",

TaskCode = TaskCode,

Title = "Demo task",

UiCode = "DemoTask",

ExpiresIn = "10d",

OnInit = new ActivityFunc<TaskStatus, TaskStatus>

{

Argument = \_onInit,

Handler = new CreateOnClientInit

{

DisplayName = "CreateOnClientInit",

Request = \_onInit

}

},

OnComplete = new ActivityFunc<TaskStatus, TaskStatus>

{

Argument = \_onComplete,

Handler = new CreateOnClientComplete

{

DisplayName = "CreateOnClientComplete",

Request = \_onComplete

}

}

};

}

}

internal sealed class CreateOnClientInit : CodeActivity<TaskStatus>

{

private static readonly ILog Log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

public InArgument<TaskStatus> Request { get; set; }

protected override TaskStatus Execute(CodeActivityContext context)

{

Log.Debug("CreateOnClientInit -> Start");

TaskStatus taskStatus = Request.Get(context);

/\*

\* User can do all the changes here before the activity is created.

\*/

taskStatus.AddParameter("TaskProp1", "TaskVal1");

//taskStatus.TaskInfo.ExpiresIn = "1m";

Log.Debug("CreateOnClientInit -> End");

return taskStatus;

}

}

internal sealed class CreateOnClientComplete : CodeActivity<TaskStatus>

{

private static readonly ILog Log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

public InArgument<TaskStatus> Request { get; set; }

protected override TaskStatus Execute(CodeActivityContext context)

{

Log.Debug("CreateOnClientComplete -> Start");

TaskStatus taskStatus = Request.Get(context);

taskStatus.SetWorkflowResult("OK");

Log.Debug("CreateOnClientComplete -> End");

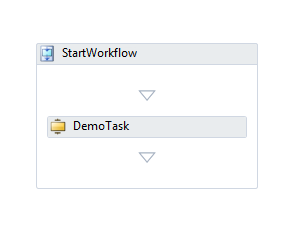
return taskStatus;

}

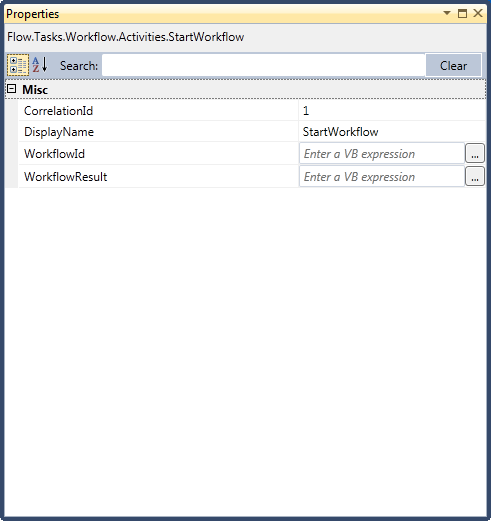
}

}

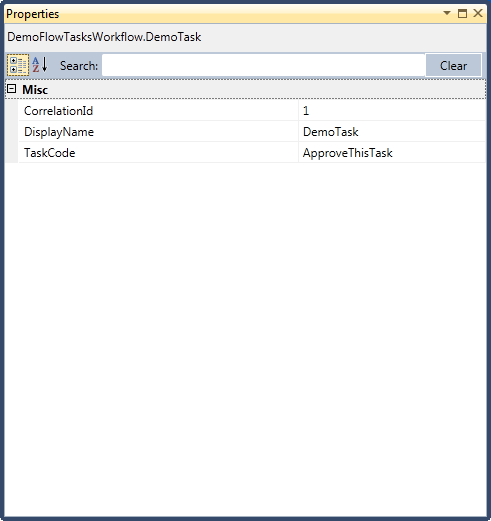
1. Open DemoWf1.xamlx remove the existing activities and add the following:



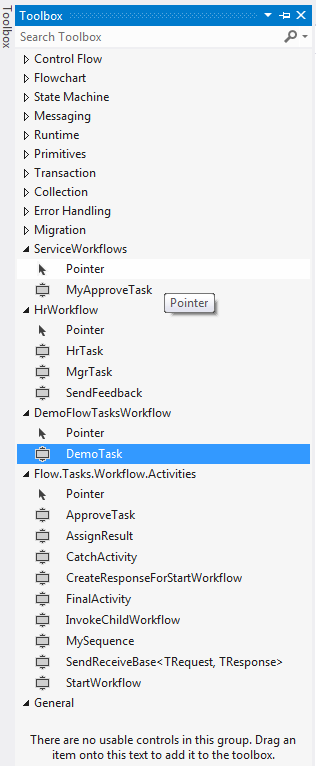
Properties for activity StartWorkflow:



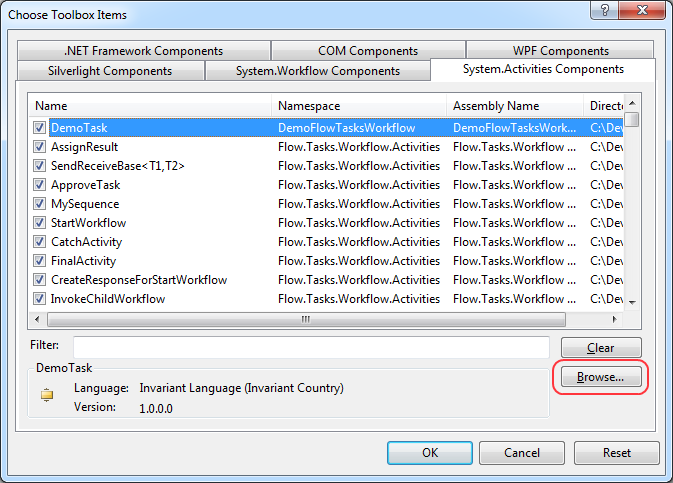
Property for DemoTask:



Note: in the toolbox you should see all this activities:



If this is not the case then from the menu select Tools/Choose Toolbox Items then browse to Flow.Tasks.Workflow.dll.



Then you should have the activities in your toolbox.

That’s it! This is all you need to create the workflow.

# Configuring the workflow

Now that you have the workflow you need to configure the framework to let it know how to find it.

1. Copy DemoWf1.xamlx under ServiceWorkflows folder (see FlowTasks Installation Guide.docs)
2. Copy DemoFlowTasksWorkflow.dll under the bin folder of ServiceWorkflows.
3. Add the following to FlowTasks.WorkflowCode and FlowTasks.WorkflowConfigutation tables





Use the script under Demo\DemoSql.

1. Add the following to the Web.config of the ServiceWorkflows web application under the serviceModel/client node:

Client

*<endpoint address="http://localhost/ServiceWorkflows/DemoWf1.xamlx/wce" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" contract="System.ServiceModel.Activities.IWorkflowInstanceManagement" name="BasicHttpBinding\_IFlowTasksOperationsControlDemo1" />*

*<endpoint address="http://localhost/ServiceWorkflows/DemoWf1.xamlx" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" contract="Flow.Tasks.Contract.IFlowTasksOperations" name="BasicHttpBinding\_IFlowTasksOperationsDemo1" />*

*<endpoint address="net.pipe://localhost/ServiceWorkflows/DemoWf1.xamlx" binding="netNamedPipeBinding" bindingConfiguration="NetNamedPipeBinding\_IFlowTasksOperations" contract="Flow.Tasks.Contract.IFlowTasksOperations" name="NetNamedPipeBinding\_IFlowTasksOperationsDemo1">*

*<identity>*

*<servicePrincipalName value="host/localhost" />*

*</identity>*

*</endpoint>*

And the following under the serviceModel/services node:

Server

<service name="DemoWf1" behaviorConfiguration="workflowServiceBehavior">

<endpoint address="" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" contract="IFlowTasksOperations" name="BasicHttpBinding\_IFlowTasksOperationsDemo1" />

<endpoint address="wce" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" kind="workflowControlEndpoint" name="BasicHttpBinding\_IFlowTasksOperationsDemo1" />

</service>

1. Add the following to the config of the DocsOnFolder connector (see FlowTasks Installation Guide.docs):

Client

<endpoint address="http://localhost/ServiceWorkflows/DemoWf1.xamlx/wce" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" contract="System.ServiceModel.Activities.IWorkflowInstanceManagement" name="BasicHttpBinding\_IFlowTasksOperationsControlDemo1" />

<endpoint address="http://localhost/ServiceWorkflows/DemoWf1.xamlx" binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IFlowTasksOperations" contract="Flow.Tasks.Contract.IFlowTasksOperations" name="BasicHttpBinding\_IFlowTasksOperationsDemo1" />

<endpoint address="net.pipe://localhost/ServiceWorkflows/DemoWf1.xamlx" binding="netNamedPipeBinding" bindingConfiguration="NetNamedPipeBinding\_IFlowTasksOperations" contract="Flow.Tasks.Contract.IFlowTasksOperations" name="NetNamedPipeBinding\_IFlowTasksOperationsDemo1">

<identity>

<servicePrincipalName value="host/localhost" />

</identity>

</endpoint>

The change this:

Change this:

<add key="WorkflowCode" value="DemoWf1" />

Done!

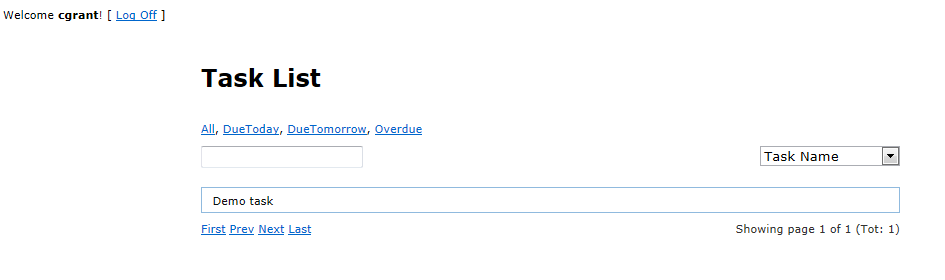
# Starting the workflow

Now if you followed FlowTasks Installation Guide.docs you should already have the DocsOnFolder windows service running. Restart the service so the changes on the config will take effect, the drop a file to the folder been monitored.

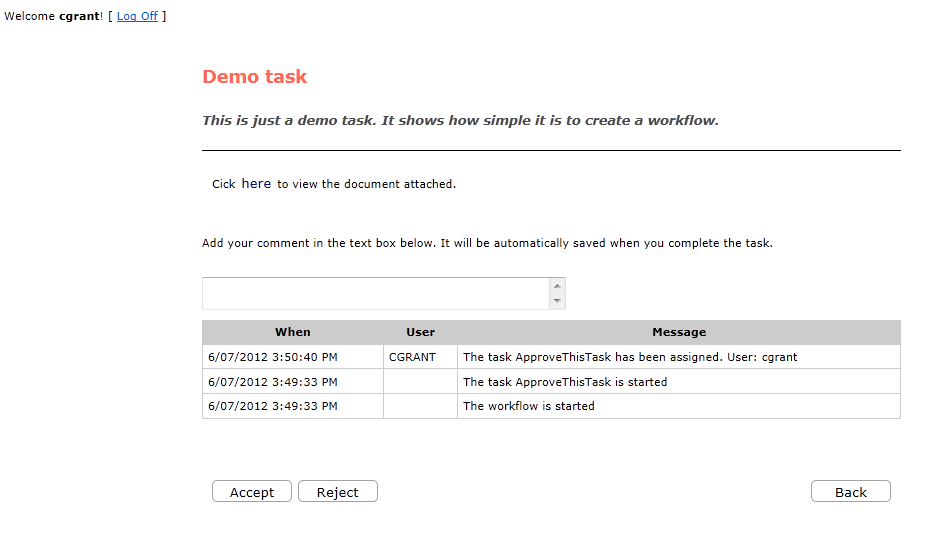
At this point a new instance of the workflow that you just created should start.

# Complete the Workflow

After the workflow is started you can see the activity in the Task list. When you login <cgrant, pwd> you will see the task in your task list.



Now you can open the activity and accept/reject it.



# Conclusion

As you can see is very simple to create a workflow using FlowTasks. This is a very simple example; however there is no limit on the complexity that your workflow can have.

Is up to you and your requirements how the workflow should start and what it should do!