Cours .Net
Workflow Foundation

Lemettre Arnaud
Arnaud.lemettre@gmail.com

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Microsoft® .NET

Sommaire

- Introduction
- Les types de workflows
- La théorie
- Création d'un Workflow
- Création de condition
- Utilisation dans un projet
- Passage de paramètres
- Utilisation avancée

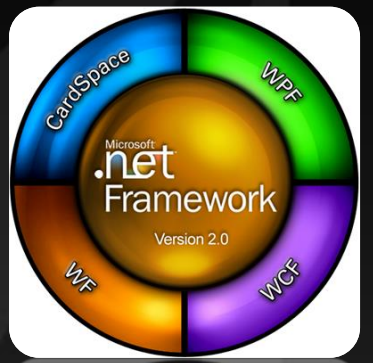
Introduction

- Dans ce cours, nous allons voir ce que sont les Workflows et comment les utiliser à travers des applications. Un workflow est un flux d'informations au sein d'une organisation.

- Définition : On appelle « workflow » (traduisez littéralement « flux de travail ») la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier

Introduction

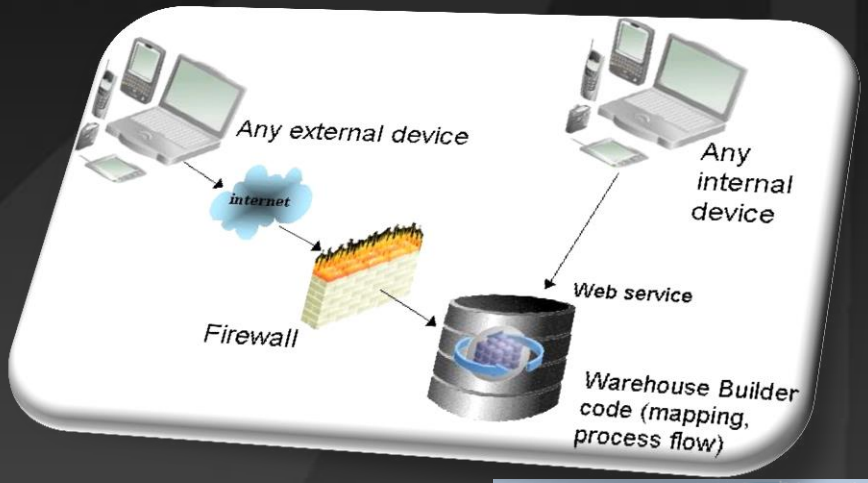
- Windows Workflow Foundation est une couche du framework .NET.
- Il est disponible à partir de la version 3.0 du Framework.
- Et peut s'exécuter sur des machines ayant Windows XP ou Vista. Une partie de ces fonctionnalités sont intégrées dans le compact Framework 3.5.
- Il permet donc la création et modélisation de workflow. C'est un moteur léger, mais auquel on peut ajouter beaucoup de services.



- Il s'intègre à Visual Studio, et on peut de ce fait disposer du designer

Introduction

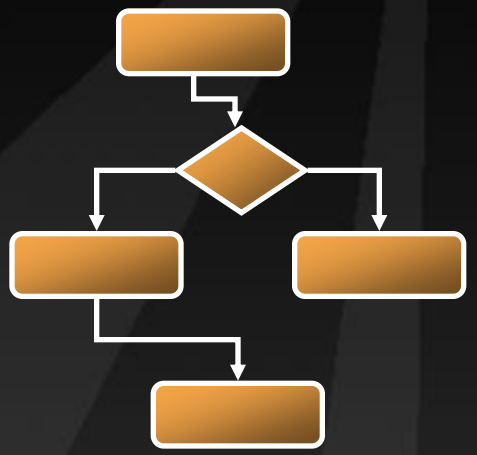
- Domaine d'application :
  - des applications consoles
  - Des applications Winforms
  - Des applications ASP.net
  - Des Web Services



Les types de Workflow

Les types de workflow

- Il existe deux types de Workflow.



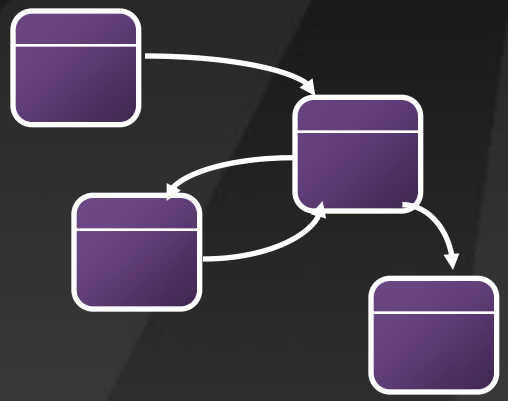
- Les Workflows séquentiels :

le fonctionnement est prédictible et les étapes s'exécutent séquentiellement (de manière ordonnée). Ce type de workflow fait intervenir le plus souvent des applications.

Les types de workflow

- Les machines d'Etats :

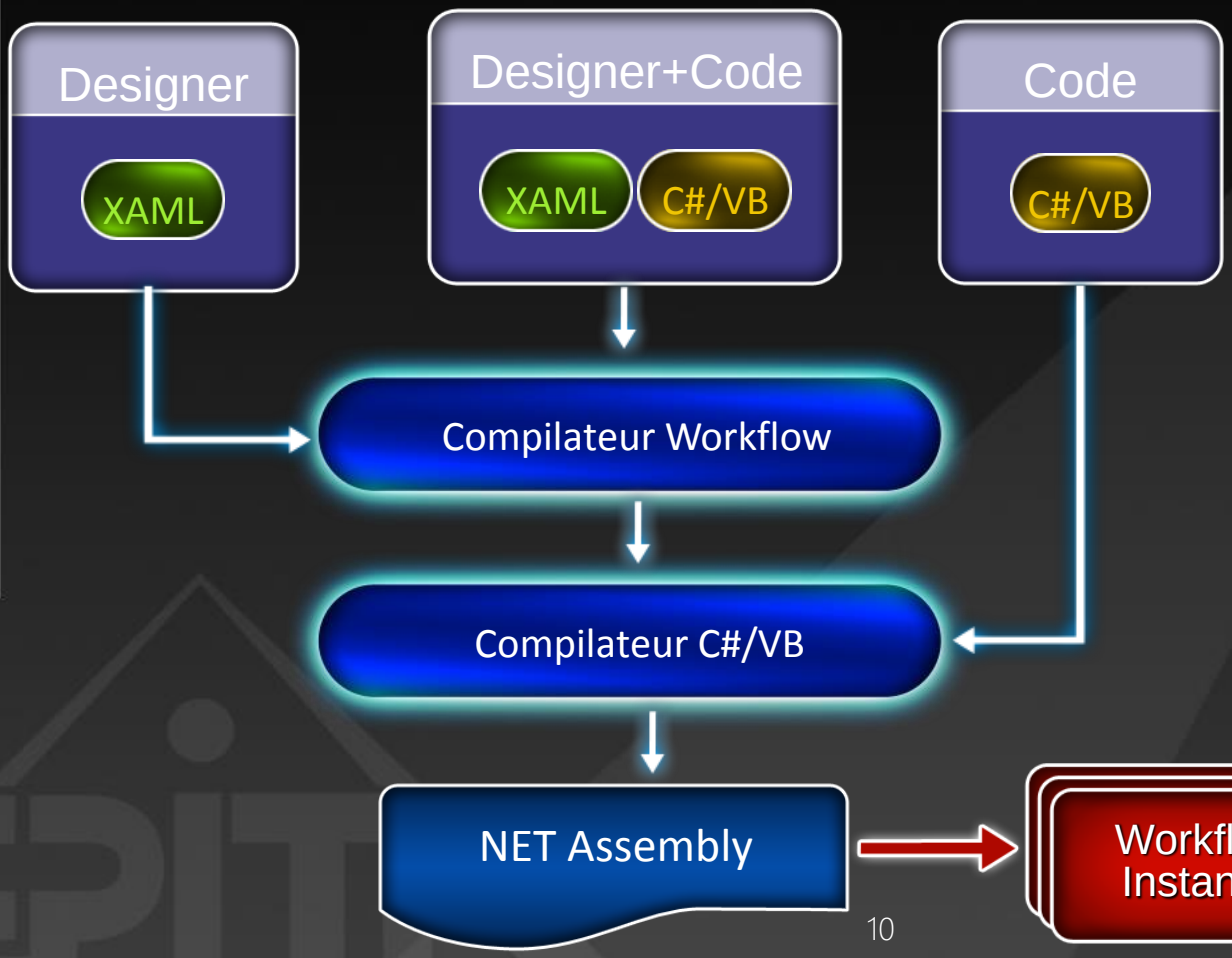
le fonctionnement est régi par le comportement et les actions de ses acteurs. Concrètement, ce type de workflow fait généralement intervenir des personnes qui modifient par leurs actions l'état du workflow.



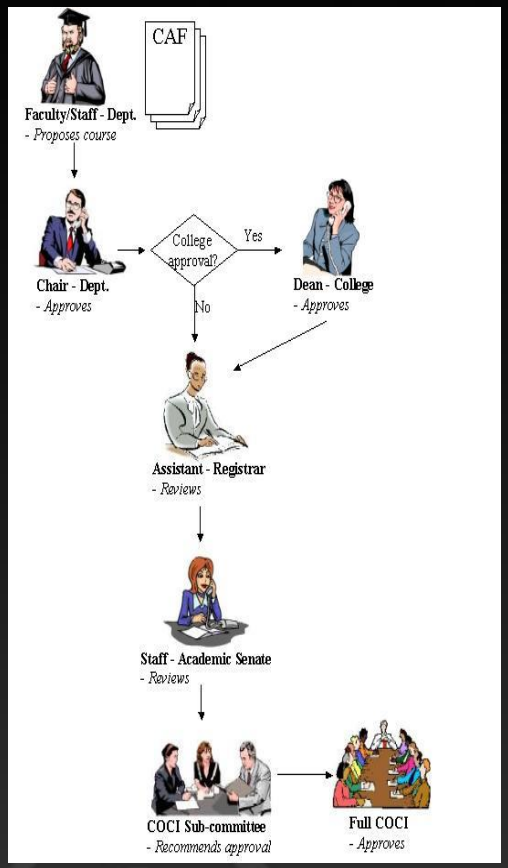
THÉORIE

Théorie

- Fonctionnement de Workflow Foundation



THÉORIE



- Nativement Workflow Foundation intègre un système de persistance de données. La sauvegarde se fait via un SQL Server 2005/2008.
- Une personnalisation de ce service est possible pour changer de base de données ou développer un système de persistance personnalisé avec par exemple un système de fichiers.

- Un service de tracking peut également être mis en place dans le but de pouvoir obtenir plus d'informations sur une instance pendant son exécution.

Théorie

- Les avantages de WF :
  - Simplicité de création et de mise en place.
  - Gain de temps car plus beaucoup de code à faire.

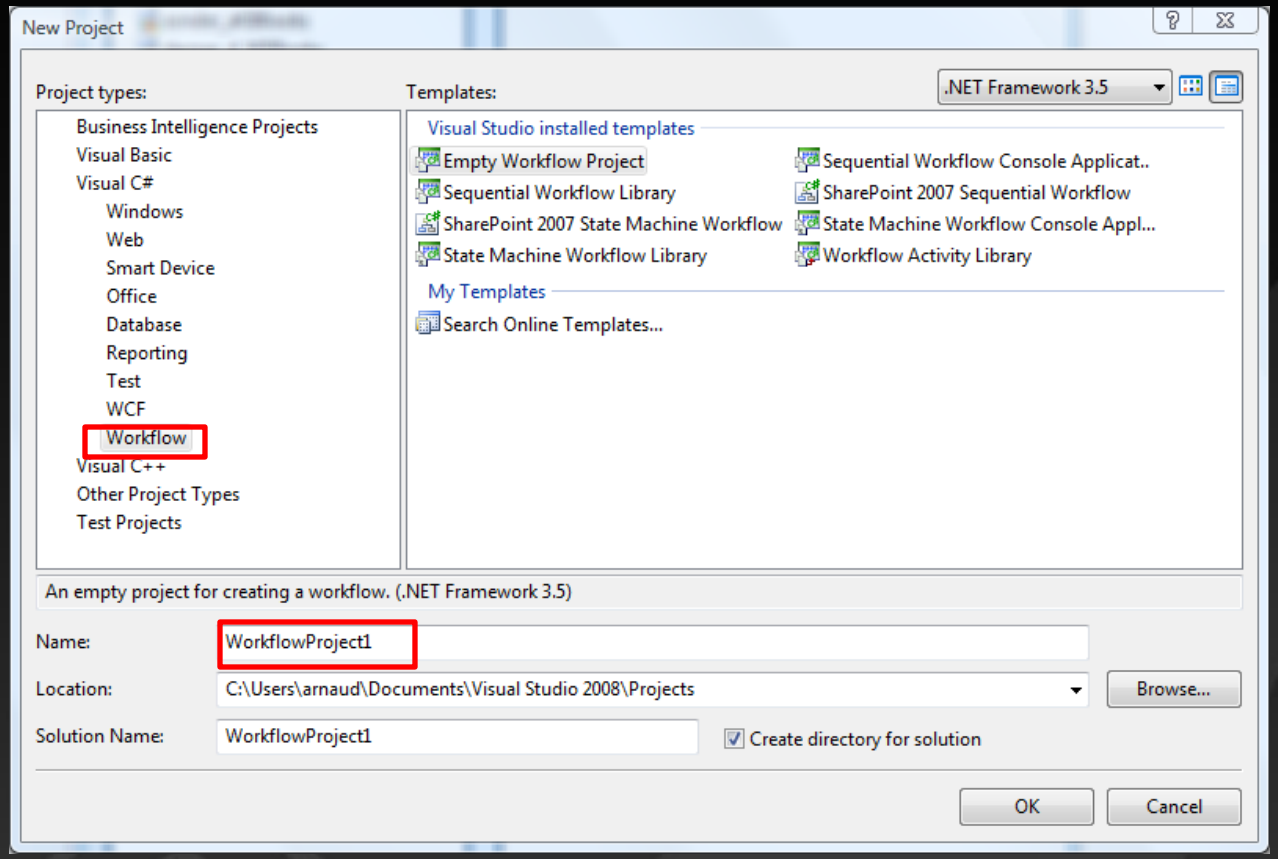
  - Supporte l'accès distant.
  - Réutilisabilité de workflows ayant déjà été programmés.
  - Possibilité de changer dynamiquement le code du workflow
    pendant son exécution.
  - On peut passer des paramètres.

La Pratique

Le designer

- Un workflow est un projet à par entière, il faut donc l'intégrer dans une solution. Pour cela, il faut ajouter un nouveau projet.

LE DESIGNER


New Project
Project types:
Templates:
.NET Framework 3.5
Business Intelligence Projects
Visual Basic
Visual C#
Windows
Web
Smart Device
Office
Database
Reporting
Test
WCF
Workflow
Visual C++
Other Project Types
Test Projects
Visual Studio installed templates
Empty Workflow Project
Sequential Workflow Console Applicat..
Sequential Workflow Library
SharePoint 2007 Sequential Workflow
SharePoint 2007 State Machine Workflow
State Machine Workflow Console Appl...
State Machine Workflow Library
Workflow Activity Library
My Templates
Search Online Templates...
An empty project for creating a workflow. (.NET Framework 3.5)
Name:
WorkflowProject1
Location:
C:\Users\arnaud\Documents\Visual Studio 2008\Projects
Browse...
Solution Name:
WorkflowProject1
Create directory for solution
OK
Cancel

Le designer

Nom projet	Description
Empty workflow Project	Projet vide rarement utilisé
Sequentioal workflow library	Crée une DLL avec le workflow séquentiel
Sharepoint 2007 State Machine workflow	Crée un projet pour intégrer directement dans sharepoint
State Machine Workflow library	Crée une DLL avec une machine d'état
Sequential Workflow Console application	Crée un projet avec une console comme hôte
Sharepoint 2007 Sequential Workflow	Crée un projet pour intégrer directement dans sharepoint
State Machine Workflow Console Application	Crée un projet avec une console comme hôte
Workflow Activity Library	Permet de faire des composants pour Workflow

Machine d'Etat

- Pour développer une machine d'état il faut :
  - Un objet pour communiquer
  - Une interface
  - Un service

Machine d'Etat

Le fichier d'échanges :

```
[Serializable]
public class TestChangeEventArgs : ExternalDataEventArgs
{
    private TestChangeType _testChangeType;
    public TestChangeType testChangeType
    {
        get { return _testChangeType; }
        set { _testChangeType = value; }
    }

    public TestChangeEventArgs(Guid instanceId, TestChangeType
                testChangeType)
        : base(instanceId)
    {
        _testChangeType = testChangeType;
        this.WaitForIdle = true;
    }
}
```

Un fichier d'enum

```
public enum
    TestChangeType
{
     Brake,
     Accelerate
}
```

Machine d'Etat

Le fichier d'interface :

```
[ExternalDataExchange]
public interface ITest
{
    event EventHandler<TestChangeEventArgs> TestChanged;
    event EventHandler<TestChangeEventArgs> TestChanged2;
}
```

Machine d'Etat

Déclaration du service :

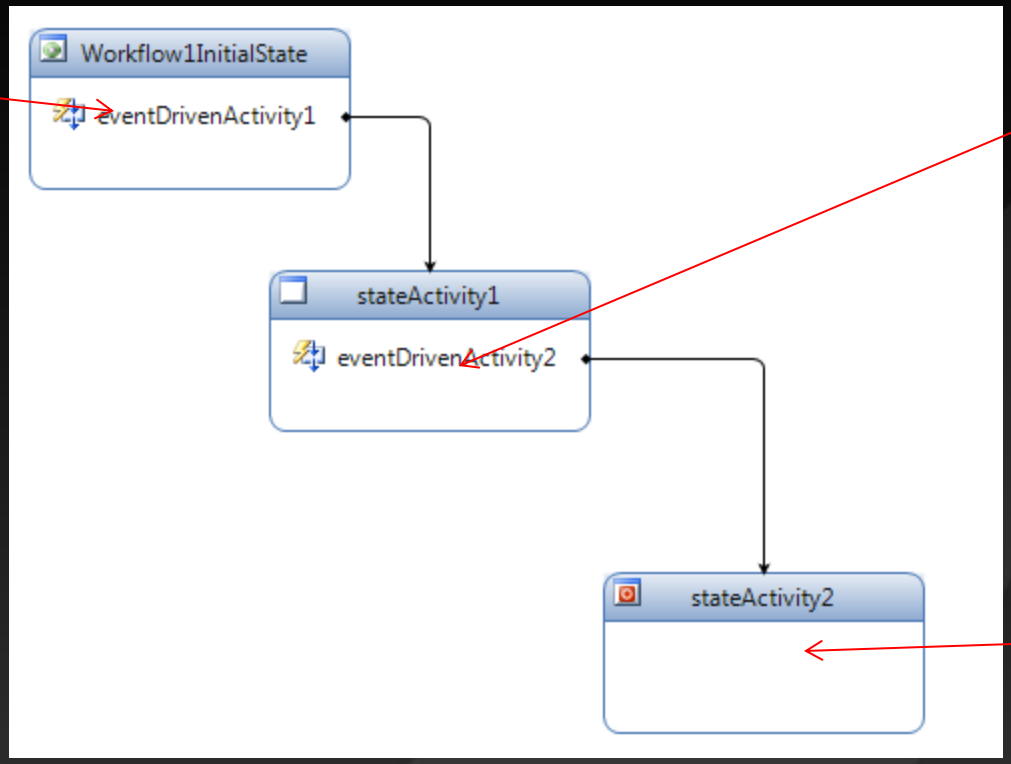
```
public class Service : ITest
{
    public event EventHandler<TestChangeEventArgs> TestChanged;
    public event EventHandler<TestChangeEventArgs> TestChanged2;
    public void RaiseEventTest(TestChangeEventArgs args)
    {
        EventHandler<TestChangeEventArgs> testChanged = TestChanged;
        if (testChanged != null)
        {
            testChanged(null, args);
        }
    }

    public void RaiseEventTest2(TestChangeEventArgs args)
    {
        EventHandler<TestChangeEventArgs> testChanged = TestChanged2;
        if (testChanged != null)
        {
            testChanged(null, args);
        }
    }
}
```

Machine d'Etat

Activité EventDriven

Autant de States …

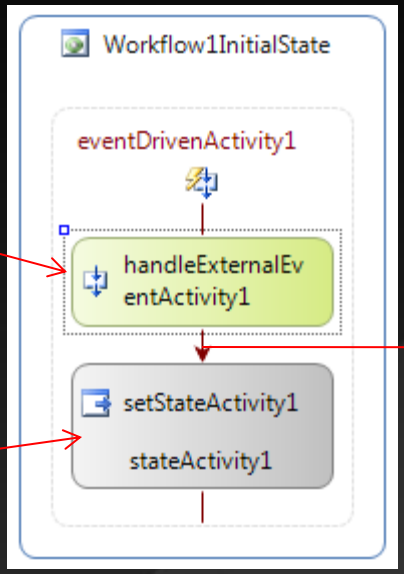


Double clic pour rentrer dans l'activité.

State avec un clic droit pour spécifier l'état final

Machine d'Etat

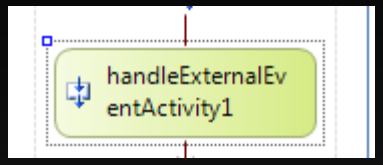
Permet de catcher un event



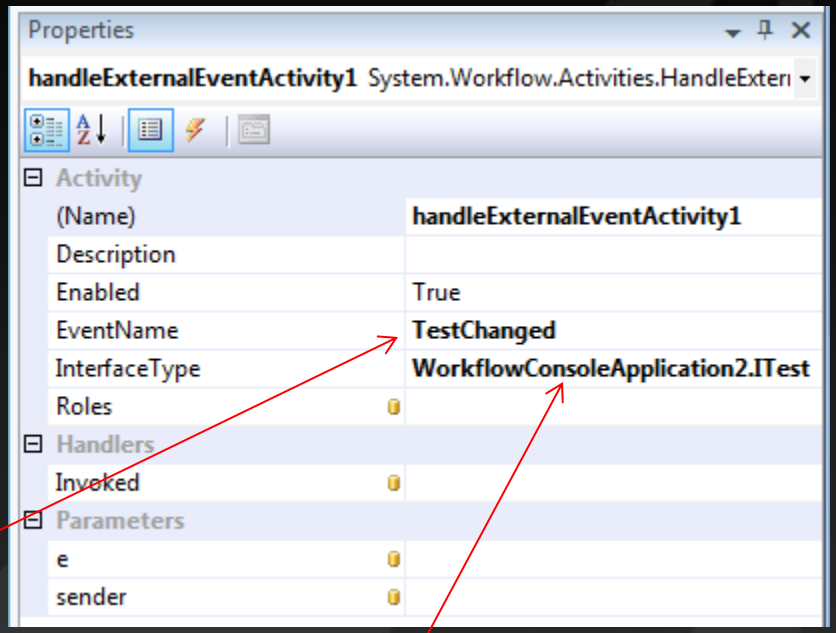
On peut ajouter toutes les activités que l'on veut. Fonctionne comme un workflow séquentiel

Obligatoire pour faire passer sur l'état suivant.

Machine d'Etat



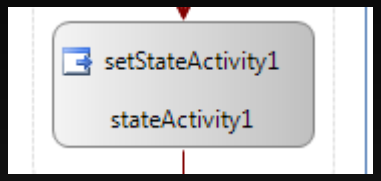
Pour compléter
l'activité, il faut mettre des propriétés



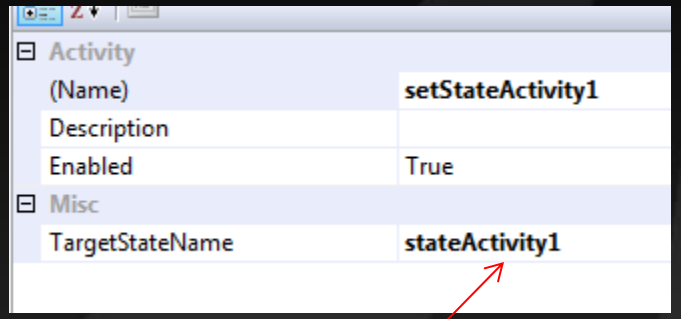
Nom de l'event qui
validera l'étape

Sélection de l'interface des events

Machine d'Etat



Pour compléter l'activité, il faut indiquer l'étape suivante

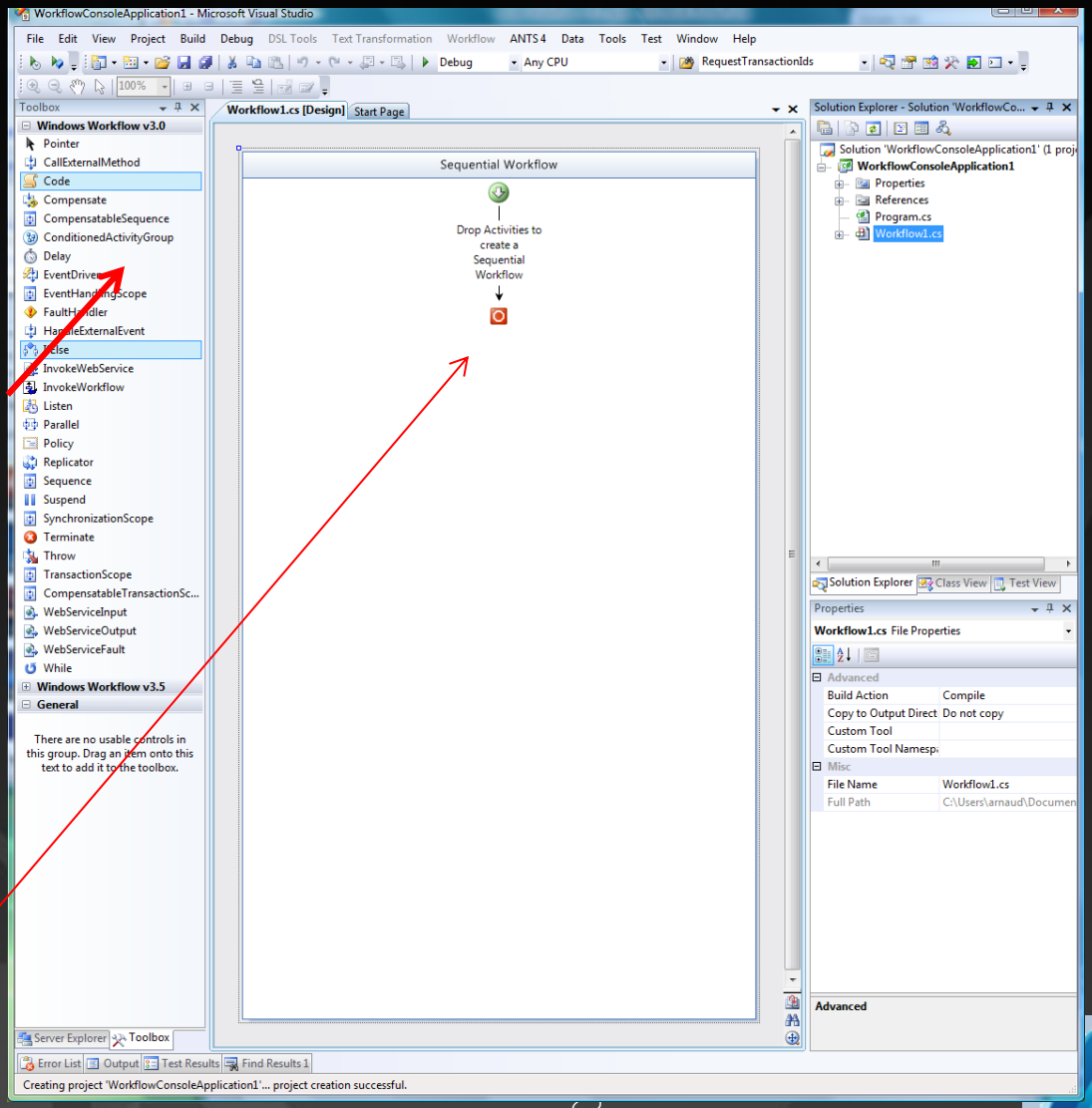


Nom de la prochaine étape

WF SEQUENTIEL-LE DESIGNER

Les contrôles

Le designer



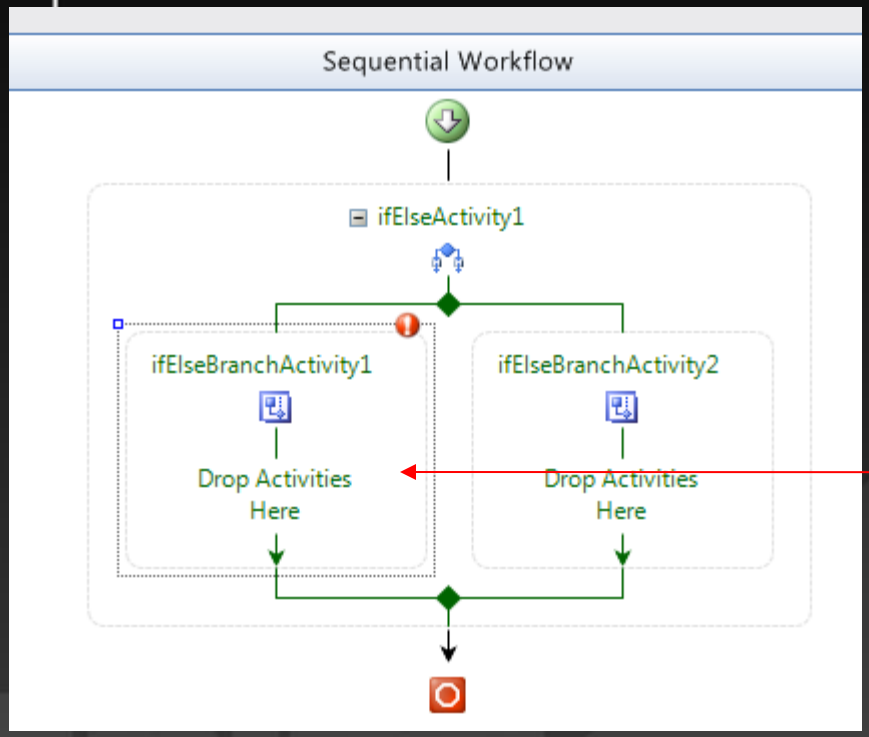
Le designer

- Les contrôles que l'on peut avoir au niveau du designer sont entre autre :

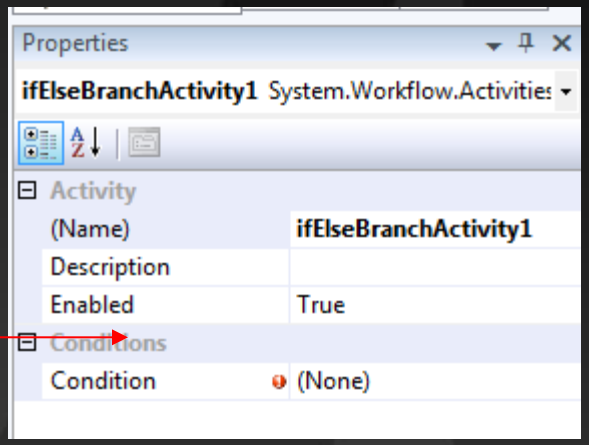
- Code
- Delay
- EventDriven
- IfElse
- InvokeWebService
- Parallel
- Sequence
- Suspend
- Terminate
- While
- …

Le designer

- Exemple avec ifElse, il faut prendre le composant et le drag & drop dans le designer



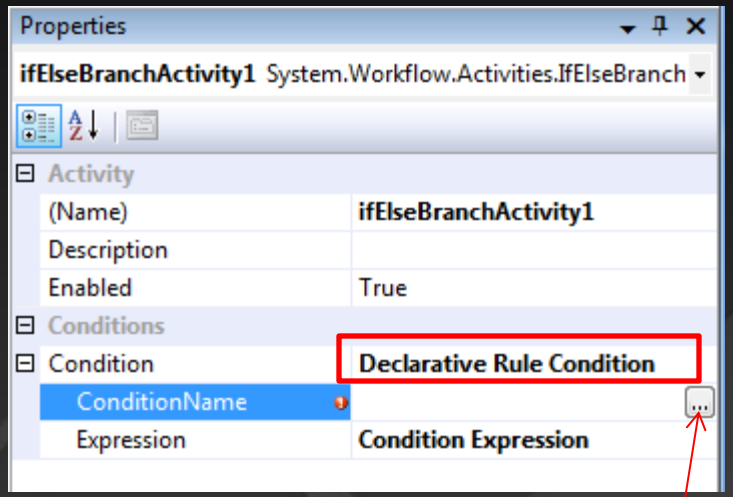
Il faut cliquer sur une branche du If, pour afficher les propriétés



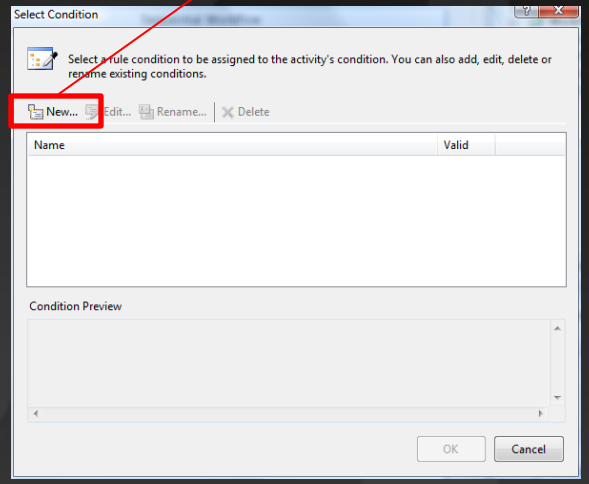
Les conditions

- Pour remplir les conditions, il y a 2 façons soit par le code ou par le designer.

Par le designer :



Création d'une nouvelle règle

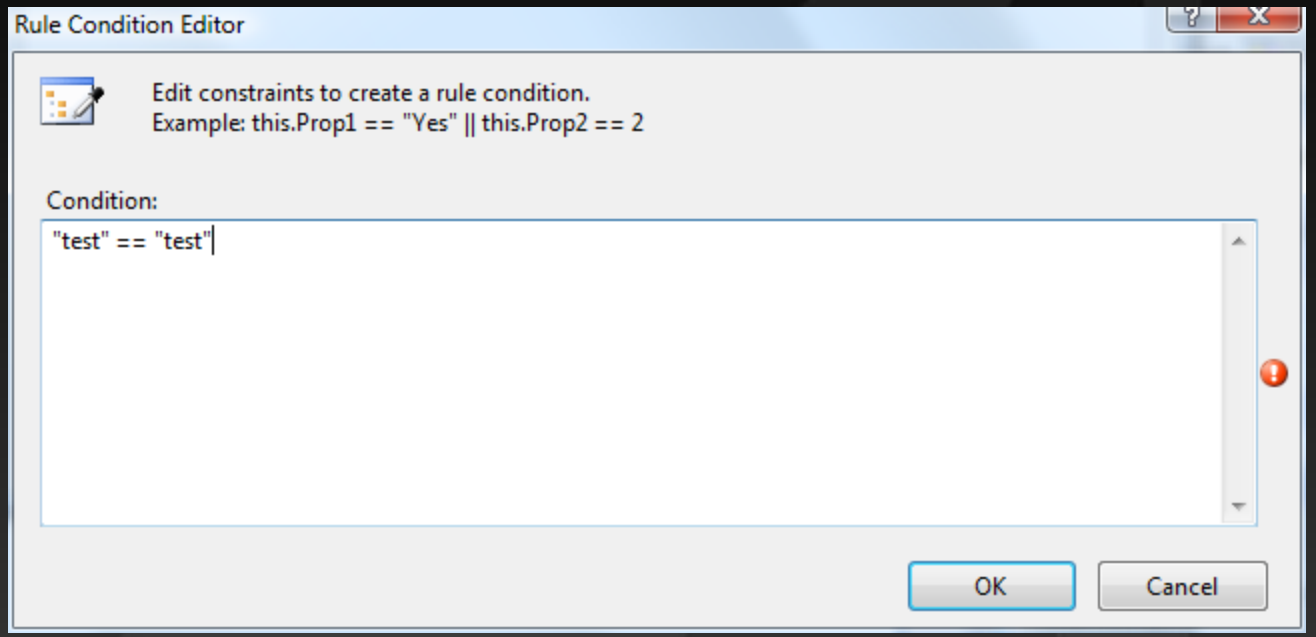


Accès au designer condition

Les conditions

- Dans cette interface il faut remplir la condition. Attention celle-ci doit renvoyer un booleen.

L'auto-complétion est disponible dans l'éditeur. On a accès sur toute la portée du projet designer

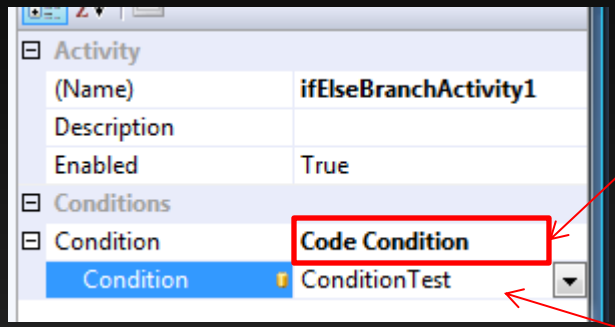


Les conditions

La 2ème méthode est de passer par le code :

```
private void ConditionTest(object sender, ConditionalEventArgs e)
{
    string test = "";
    if ("Test" == test)
    {
        e.Result = true;
    }
    else
    {
        e.Result = false;
    }
}
```

Les conditions



Pour sélectionner le fait que l'on veut utiliser une condition que l'on a codée.

Sélection du nom de la méthode

Utilisation dans un projet

- Maintenant il faut pouvoir l'utiliser dans un projet, la déclaration n'est pas la même selon que ca soit un workflow séquentiel ou une machine d'état

Utilisation dans un projet

Pour un workflow séquentiel

```
using (WorkflowRuntime workflowRuntime = new WorkflowRuntime())
{
    AutoResetEvent waitHandle = new AutoResetEvent(false);

    workflowRuntime.WorkflowCompleted += new
    EventHandler<WorkflowCompletedEventArgs>((sender, e) =>
    {
        waitHandle.Set();
    });

    workflowRuntime.WorkflowTerminated += new
    EventHandler<WorkflowTerminatedEventArgs>((sender, e) =>
    {
        Console.WriteLine(e.Exception.Message);
        waitHandle.Set();
    });

    WorkflowInstance instance =
    workflowRuntime.CreateWorkflow(typeof(WorkflowLibraryRadar.WorkflowRadar),
    instance.Start();
    waitHandle.WaitOne();
}
```


Utilisation dans un projet

Pour une machine d'état :

```
using (WorkflowRuntime workflowRuntime = new WorkflowRuntime())
{
    //rajout du systéme d'échange
    ExternalDataExchangeService dataExchange = new ExternalDataExchangeService();
    workflowRuntime.AddService(dataExchange);
    //rajout du service
    Service testService = new Service();
    dataExchange.AddService(testService);

    AutoResetEvent waitHandle = new AutoResetEvent(false);
    workflowRuntime.WorkflowCompleted += delegate(object sender, WorkflowCompletedEventArgs e) {
     waitHandle.Set(); };
    workflowRuntime.WorkflowTerminated += delegate(object sender, WorkflowTerminatedEventArgs e)
    {
        Console.WriteLine(e.Exception.Message);
        waitHandle.Set();
    };

    WorkflowInstance instance =
     workflowRuntime.CreateWorkflow(typeof(WorkflowConsoleApplication2.Workflow1));
    instance.Start();
    //déclenchement du premier état
    testService.RaiseEventTest(new TestChangeEventArgs(instance.InstanceId,
    TestChangeType.Brake));

    waitHandle.WaitOne();
}
```

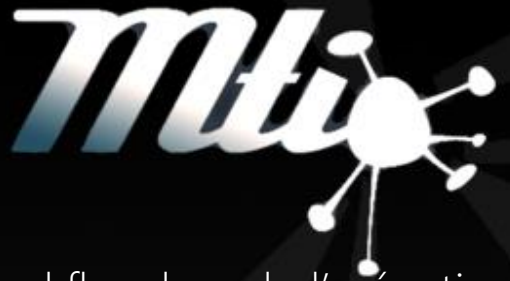
Passage de paramètres

Avec les workflows, on peut également leur passer des paramètres à l'initialisation, mais aussi des paramètres de sortie.
Pour cela dans le workflow.cs, il suffit juste de rajouter des Properties en public, peut importe le type.

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters.Add("Test", 12);
parameters.Add("Test2", "test");

WorkflowInstance instance =
    workflowRuntime.CreateWorkflow(typeof(WorkflowConsoleApplication2.Workflo
    w1), parameters);
```


Passage de paramètres

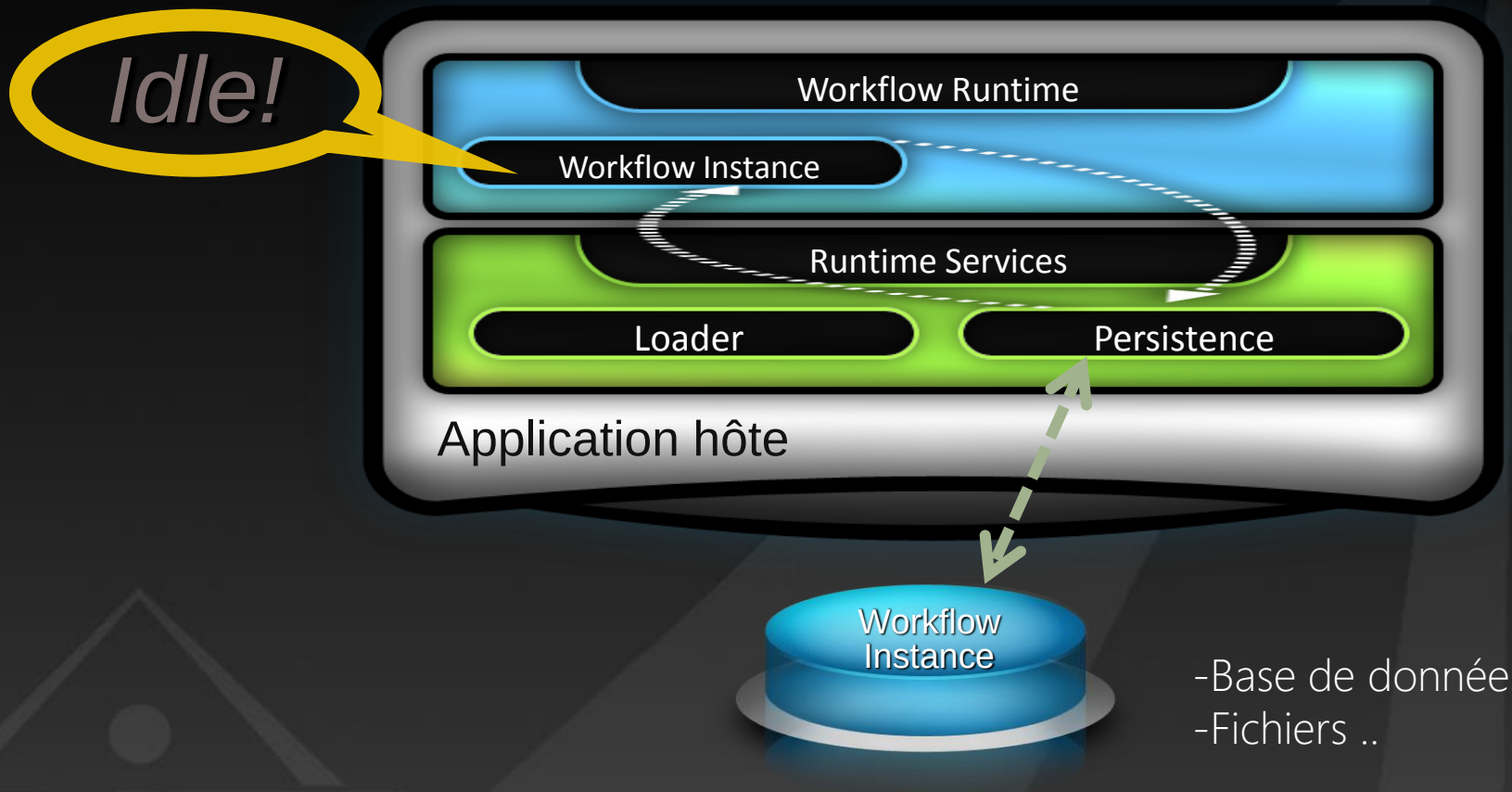


Pour récupérer des paramètres, il faut les compléter dans le workflow lors de l'exécution

```
workflowRuntime.WorkflowCompleted += new
    EventHandler<WorkflowCompletedEventArgs>((sender, e) =>
{
    bool supLimit =
    Boolean.Parse(e.OutputParameters["OutputMessage"].ToString());
    waitHandle.Set();
});
```


UTILISATION AVANCÉE

Mise en persistance



-Base de données
-Fichiers ..

Mise en persistance

Le système de persistance par défaut repose sur la base de données Sql Server.
Mais si celui-ci ne vous convient pas, vous pouvez implémenter votre propre système.

Pour le mettre en place, vous devez lancer les scripts pour construire les tables en base de données.

C:\Windows\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\

Il faut exécuter les scripts suivants :

- SqlPersistenceService_Schema.sql (pour les tables)
- SqlPersistenceService_Logic.sql (pour les procédures stockées)

Mise en persistance

Les tables qui apparaissent dans la base :

Mise en persistance

Il faut rajouter le service de persistance, comme pour ajouter le service des events :

```
SqlWorkflowPersistenceService stateservice =
    new SqlWorkflowPersistenceService(
        "Data Source=localhost;Initial
   Catalog=TestWorkflow;Integrated Security=True", true,
        new TimeSpan(0, 1, 0), new TimeSpan(0, 0, 30));
workflowRuntime.AddService(stateservice);
```

Mise en persistance

Pour récupérer le workflow une fois qu'il est persisté il suffit juste de faire :

```
workflowRuntime.GetWorkflow(idInstance).Load();
```

Guid du workflow

ceci fonctionne si nous ne coupons pas le service et que nous avons toujours le Moteur de workflow.

Mise en persistance

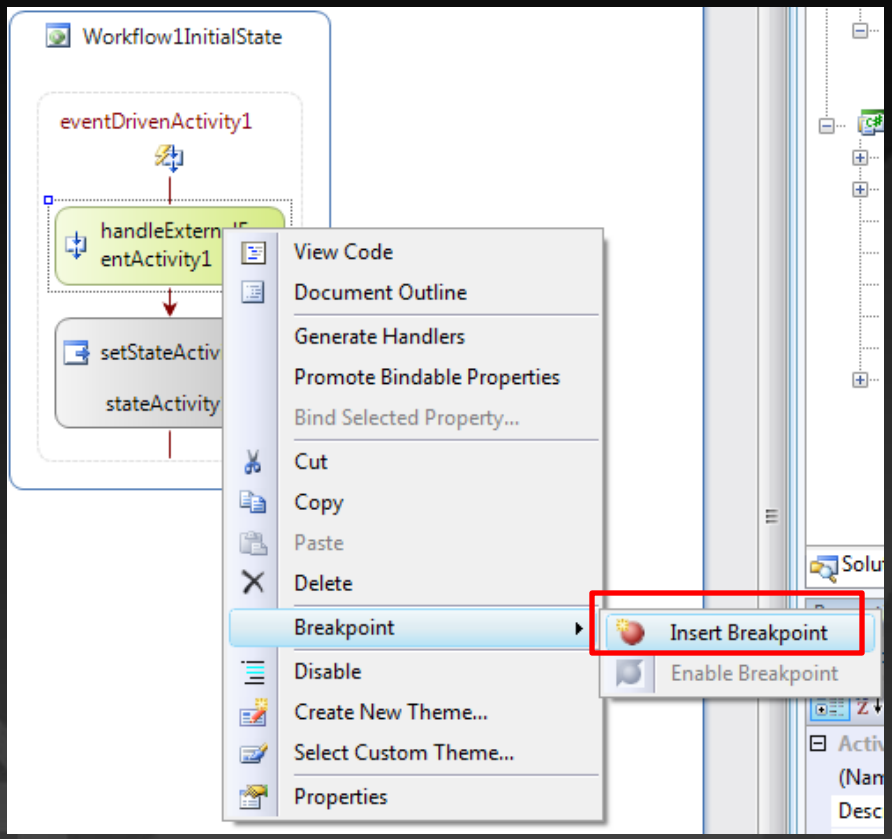
Dans le cas où on perd le runtime (crash application, fermeture …) :

```
SqlWorkflowPersistenceService persistance
    =(SqlWorkflowPersistenceService)

    workflowRuntime.GetService(typeof(SqlWorkflowPersistenceService
    ));
IEnumerable<SqlPersistenceWorkflowInstanceDescription> instances =
    persistance.GetAllWorkflows();
workflowRuntime.GetWorkflow(instances.First().WorkflowInstanceId).
    Load();
```

Debug

Les workflows peuvent également être debuggés il suffit de faire un clic droit sur une activité, et de placer un breakpoint.



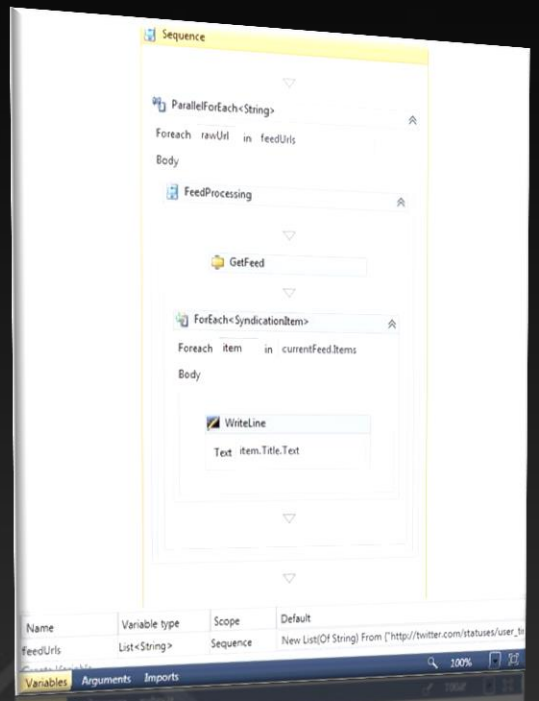
WF 4.0

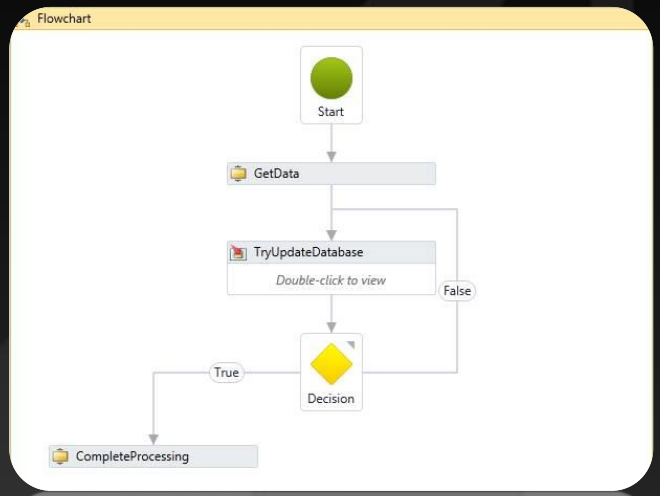
WF 4.0

- Réécriture du coeur du moteur ainsi que des services afin d'améliorer les performances
- Réécriture également du designer pour faciliter le développement
- Une meilleur intégration de WCF avec les modèles de WF
- Expression visual basic disponible au sein des activités, il n'y a plus de *.CS
- Des nouvelles activités de disponible
- De nouvelles façons de hoster les workflow (app fabric, hébergement automatique au sein d'un service, ...)

WF 4.0

Exemple des nouvelles interfaces :





WF 4.0

- Dans les 1eres versions de WF 4.0
  - Disparition des machines d'état
  - Apparition des dataFlow (moins flexibles que les 1er)
- Avec la CTP
  - Réapparition des machines d'état

WF 4.0

- Pour hoster les workflows :
  - Facon synchrone

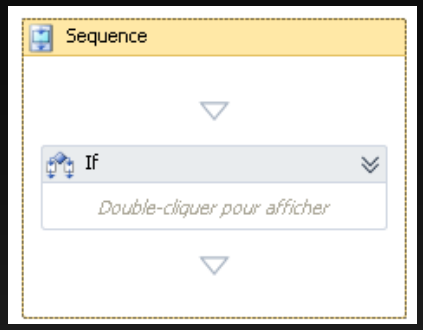
```
WorkflowInvoker.Invoke(new Workflow.WorkflowRadar());
```

  - Facon Asynchrone

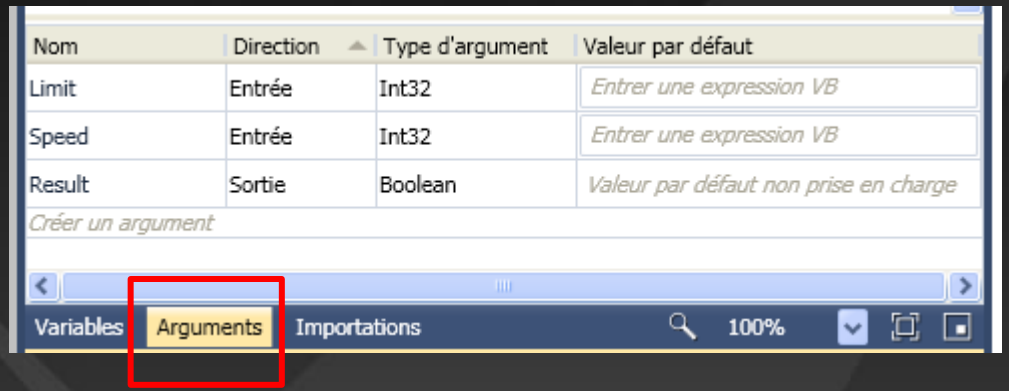
```
WorkflowApplication wfApp = new WorkflowApplication(new Workflow.
   WorkflowRadar());
wfApp.Run();
```


WF 4.0

- Le passage de paramètres, diffère un peu de la version précédente :
  - Déclaration des paramètres d'entré et de sortie :



En sélectionnant la séquence d'entré dans un 1er temps
Puis en passant sur l'onglet Argument :



WF 4.0

- L'initialisation se fait de la même manière à travers un IDictionnary

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters.Add("TestImmatriculation", 13256);

 IDictionary<string, object> output = WorkflowInvoker.Invoke(new Workflow.WorkflowRadar()
    , parameters);

 if (Convert.ToBoolean(output["Result"]))
    Console.WriteLine("GoodJob");
```

WF 4.5

WF 4.5

- Les expressions peuvent être écrites en C# (seulement VB auparavant)
- Retour des state machine

QUESTIONS ?